tanulmányok

205/1988

MTA Számítástechnikai és Automatizálási Kutató Intézet Budapest

MAGYAR TUDOMÁNYOS AKADÉMIA
SZÁMITÁSTECHNIKAI ÉS AUTOMATIZÁLÁSI KUTATÓ INTÉZETE

ÚJ ESZKÖZÖK A FOGALMI MODELLEZÉSBEN

HERNÁDI ÁGNES

Tanulmányok 205/1988

A kiadásért felelős:

DR. KEVICZKY LÁSZLÓ

Jelen tanulmány a szerző kandidátusi értekezése.

Témavezető:

KNUTH ELŐD

ISBN 963 311 247 8
ISSN 0237-0131

Készült a
technografik ksz.
gondozásában
Dél-Pesti ÉSZKV Nyomda

"A számítógépekben már ma is több lehetőség rejlik, mint amennyivel élni tudunk. Folyamatosan üzemeltetjük a gépeket, de sok szakember úgy érzi, még nem tanultuk meg, hogyan használjuk ki igazán a gépek képességeit. Ezért gyors társadalmi és tudományos fejlődésre van szükség, ha az ember teljes értékű partner akar maradni. És ez, úgy vélem, az egyik legizgalmasabb kihívás, amivel az emberiségnek szembe kell néznie."

John G. Kemeny: Az ember és a számítógép
(Gondolat, Budapest, 1978.)

Tézis-összefoglaló

1. Kidolgoztam a számítógépes rendszerek környezet- és követelmény modellezésének új fogalmi taxonómiáját. Ez az ismert fogalmi modellező eszközöktől a következőkben tér el:

   a) nem csupán a hagyományos fogalmi tervezési szakaszt veszi célba, hanem kiterjed a korábbi módszerekkel egyáltalán nem vizsgált megismerési, környezet-elemzési és követelmény-specifikációs szakaszokra.

   b) a specifikáció-készítés folyamatát a gyakorlati tevékenységnek megfelelő szakaszokra explicit módon bontja fel, eltérő fogalmi alapokat nyújtva a természetükben is különböző szakaszok mindegyikéhez (probléma megismerés, fogalmi terv, logikai terv stb.).

2. Kidolgoztam a számítógéppel segített fogalmi modell tervezés egy olyan transzformáció-rendszerét, mely a modell integritásának sértetlenségét akkor is képes biztosítani, ha meglévő objektum példányokkal rendelkező fogalmi meghatározásokat utólag alakítunk át. Ilyen

eszközöket a korábban ismert modellek nem biztosítanak. Kidolgozását a modellezési technikáknak a valóságos megismerési folyamat iteratív és önrevideáló természetéhez való közelítése indokolta.

3. Kidolgoztam azokat az alapvető matematikai struktúrákat, amelyek az előző tézisekben összefoglalt konstrukciók szemantikájának egzakt formális ábrázolását teszik lehetővé. Mivel ennek alapján a különböző szintű specifikációkat many sorted logikába lehet fordítani, e specifikációkat olyan logikai elméletként is tekinthetjük, amelyek a csak logikán alapuló nyelvek által nem nyújtott leírás-strukturáló képességekkel rendelkeznek.

Köszönetnyilvánítás

Elôször és mindenekelôtt szeretnék köszönetet mondani témavezetômnek: Knuth Elôdnek, a matematikai tudományok doktorának mindazért a bátorításért, tanácsért és támogatásért, melyet az évek során tôle kaptam. Enélkül az értekezés nem készült volna el. Mélységesen hálás vagyok ezért.

Köszönettel tartozom Dettrich Árpádnak, aki mióta csak ismerem buzdított, és mindig szakított idôt arra, hogy segítsen, valahányszor hozzá fordultam. Köszönöm.

Hálás vagyok kollégáimnak a szakmai vitákért és konstruktív javaslatokért.

Szeretném megköszönni családom megértô türelmét, különösen kislányom feltétlen szeretetét, ami sokszor adott erôt a csüggedés perceiben. Köszönöm!

T A R T A L O M J E G Y Z É K

1. Fejezet

BEVEZETÉS

1.1 A háttér: a modellalkotás folyamata

Az emberi gondolkodásban alapvető a nagy mennyiségű tudás felhalmozódása és a kapcsolatok asszociatív felderítése. Ez lehetőség, de egyben korlát is.

Lehetőség, mert ily módon a megismert világ dolgait automatikusan "végtelen sok" kapcsolatba hozhatjuk, melyeknek egy része - ismereteink szerint - valódi, míg a többiről nem tudunk érdemlegeset mondani.

A korlátot pedig az jelenti, hogy ez a rendszer teljesen szubjektív mind rendszerszervezésében, mind bővíthetőségében. A számítógép is rendelkezhet rendkívül nagy mennyiségű tudással, de annak "begyűjtése" igen sok tudás módszeres és rendszeres egyesítésével történik. A gép esetén az információ visszanyerése tehát nem az asszociációk - látszólag - véletlen sorozatával történik, hanem többek megegyezésén alapuló rend szerint. Ez az adottság magában hordja - ezt különösen a modern gépek kapacitása és sebessége ismeretében állíthatjuk - egy rendkívül nagy tudáshalmaz sokféle - bonyolult rendezettségén alapuló - lekérdezését, vagy a részhalmazok gyors összegyűjtését, amelynek segítségével az ember újabb következtetésekre juthat.

A gép segítségével módszeresen összegyűjtött nagy mennyiségű és gyorsan hozzáférhető tudás így kiegészítheti az emberben lévő tudást, amelyet ezután az ember - asszociáción alapuló - intuitív készsége alapján újabb ismeretek felderítésére használhat. Az új ismeret hozzáadásával pedig tovább növelhető a "gép tudása".

Így az ember és a gép között olyan "szimbiózis" jöhet létre, amelynek hatása beláthatatlan.

Ebben a szimbiózisban (ma még!) az emberé a vezető szerep, és rajta múlik, hogy a számítógép milyen szintű segítőtársává válik.

A szimbiózis gondolatát - elsősorban - a párbeszédes üzemmódú (interaktív) rendszerek megjelenése vetette fel.

A fenti gondolatsor alapján vizsgáljuk meg, hogy mai szemléletünk szerint milyen színvonalon áll pl. egy gazdasági rendszer (egység) vezetéséhez szükséges információt szolgáltató rendszer kivitelezése, és az igények alapján milyen fejlesztési lehetőségeket kell megvizsgálnunk, ezekhez milyen eszközöket kellene biztosítanunk.

Egy jó információs rendszer célja nem az, hogy sok információval lássa el felhasználóit, hanem az, hogy a töméntelen tárolt információ alapján olyan összefoglaló jelentést vagy áttekintő tájékoztatást tudjon adni, ami a felhasználó munkájához éppen szükséges, és mindezt lehetőleg a felhasz-

náló által kért vagy hasznosnak vélt formában. Ezért az információs rendszerek tervezésénél nagy figyelmet kell fordítani arra, hogyan is akarjuk majd az információt visszakapni. Olyan, elég rugalmasan változtatható rendszert kell létrehozni, amely a mai szükségletek kielégítésén túl a jövő igényei szerint is átalakítható.

A hosszútávú tervezéshez és a döntéshozatalhoz elengedhetetlenül szükséges az intézmény vagy vállalat tevékenységének elméleti leírása, modellje. Ilyen modell alkalmazásának szükséges feltétele egy hatékony információs rendszer léte, de ez utóbbi önmagában még nem pótolja a modellt.

A modellalkotás rákényszerít a működésről kialakított elképzelések explicit megfogalmazására, ugyanakkor a modell használata folyamán tapasztalatokat szerzünk a hibás vagy hiányos feltevésekről, és a modell továbbfejleszthető.

Ilyen modellek birtokában érdemi (emberi) döntésekre csak az összes lényeges tényező, valamint a rövid- és hosszútávú következmények ismeretében kerülne sor a számítógép tájékoztatása alapján.

A modern párbeszédes üzemmódú számítógéprendszerek ideálisan alkalmasak ilyen modellek készítéséhez. A gépben fogalmi modell segítségével felépíthetjük a világ vizsgált tartományának "kicsinyített mását", gondosan leírva a különböző komponensek dinamikus kölcsönhatásait. Ugyanakkor nyilvánvaló, hogy ez a "kicsinyített más" sohasem lehet olyan való-

sághû, mint az elméleti modell. Az elméleti modell pl. állíthatja, hogy minden embernek van születési dátuma, mégis elôfordulhat, hogy a rendszer még nem jegyezte fel, pl. egy eszméletlen állapotban intenzív szobára vitt beteg esetén.

Ilyen modellek építéséhez olyan automatizált eszközre lenne szükség, amely nemcsak a modellek fogalomrendszeréhez igazodik, hanem támogatja a modellezés folyamatát is.

1.2 A modellezés általunk alkalmazott megközelítése

Ahhoz, hogy valóban megkönnyítsük a modellek létrehozását, kezelését, karbantartását és fejlesztését - különösen akkor, ha nagy és bonyolult modellekrôl van szó - nem elég az, ha lehetôvé tesszük a modelleknek osztályokba szervezett és tulajdonságokkal összefüggésbe hozott objektumokkal történô kifejezését. Meg kell engednünk, hogy a nagy modelleket strukturáló és könnyebben szerkeszthetôvé, módosíthatóvá tevô szervezési/absztrakciós eszközöket dinamikusan, lehetôleg párbeszédes üzemmódban át lehessen alakítani úgy, ahogy azt a hibás ábrázolások felismerése megköveteli. Biztosítani kell továbbá olyan eszközöket, amelyek a rendszer fokozatos fejlesztése, módosítása közben az információk közötti logikai összefüggések (ellentmondások) felderítésében segítenek, lehetôséget adva a dinamikus újraértékelésre. Ez merôben különbözik a többi megközelítéstôl.

A fogalmi modellezést támogató programozási nyelvekkel (pl.

TAXIS [MY'80a], [MY'80b] Galileo [ALB'85] szemben a mi célunk a követelmény-specifikációs tudás karbantartása és ellentmondás-mentességének biztosítása. Ugyanakkor különbözik felfogásunk az RML [GR'86] követelmény-specifikációs nyelv megközelítésétől is. Egy RML modell ugyanis idővel nem változik, nem adatmodell. RML-ben megfogalmazott követelmény-modell a világ apokalipszis utáni nézetéből írja le azokat a szemantikai feltételeket, melyek e világot jellemezték. Az RML kutatásainak talaján kifejlesztett CML [STAN'87] pedig egy tudás-reprezentációs nyelv, mellyel ellentétben mi a fogalmak és objektumok interaktív kezelését ajánljuk.

1.3 A modellező nyelvekkel szemben támasztott főbb követelmények

A specifikációs nyelvek központi célja tehát, hogy lehetőséget nyújtsanak a világra vonatkozó tudás természetes és kényelmes módon való összegyűjtésére és ábrázolására, ugyanakkor úgy szervezzék és strukturálják ezt a tudást, hogy azt a rendszerfejlesztők és a végső felhasználók könnyen megértsék.

E célok elérésében alapvetőeknek tartjuk a következő ismérveket, noha a felsorolás nyilván nem teljes. Először azokat a tudásfajtákat érintjük, melyek ábrázolására a modellező nyelvnek képesnek kell lennie. Ezután térünk csak rá a tudásszervezés vonatkozásaira.

1.3.1 Entitások, változások és kényszerek leírása

Először is, egy jó modellező nyelv megengedi, hogy a tervező leírhassa a tanulmányozott tartomány entitásait, a világ változásait, illetve eseményeit, valamint kijelenthessen kényszereket és feltételezéseket.

Néhány modellező nyelv - pl. az entitás-kapcsolat megközelítésen alapuló nyelvek - és módszer jó az entitások tulajdonságainak leírására, de gyenge a változások és kényszerek leírásának szempontjából. A legtöbb szemantikai adatmodell csak perifériálisan (pl. bizonyos átmeneti integritási kényszerek előírásával) modellezi a világ dinamikus oldalát. A fogalmi modellezést támogató programozási nyelvek (pl. TAXIS [MY'80b], Galileo [ALB'85] tranzakciókon keresztül veszik figyelembe a tevékenységek modellezését. A főként folyamatokhoz, eljárásokhoz vagy eseményekhez tervezett specifikációs módszerek alkalmasabbak a változás leírására. Kényszerek kijelentéséhez a logikák a legkényelmesebb nyelvek. Mivel egy modellező nyelvnek mindhárom dolog leírására alkalmasnak kell lennie, nélkülözhetetlen, hogy egyetlen nyelvben számos modellezési lehetőséget kombináljunk.

1.3.2 Kivételkezelés

Miután az információs rendszerben ábrázolt fogalmak legtöbbje a mindennapos emberi tapasztalatból ered, azaz úgynevezett "természetes fajta", nincs pontos definíciójuk. A természetes fajták leírásai, úgy tűnik, vagy reménytelenül határozatlanok, vagy ellentmondásra hajlamosak. Ez ellentétes - mondjuk - a matematika mesterségesen definiált fogalmaival, mint amilyenek pl. a halmazok vagy a sorozatok, melyek mindig "definíció szerint" kielégítik a rájuk vonatkozó kényszereket. A modellező nyelvek fejlesztőinek számolniuk kell a természetes fajták leírásából eredő problémákkal.

1.3.3 Időábrázolás

Mivel az idő múlása emberi tapasztalatunk sarkköve, és közvetlenül kötődik a világ dinamikus aspektusainak leírásához, a modellező nyelveknek is tudniuk kell kifejezetten beszélni az időről és a világ időbeli fejlődéséről*. Azok a nyelvek, melyekből hiányoznak ezek a képességek, arra kényszerítik a tervezőt, hogy vagy csak a világ statikus aspektusait vegye tekintetbe, vagy a dinamikát ne leírással, hanem eljárások sorozatával írja körül.

---

* Természetesen sok olyan alkalmazás van, melyeknek az időábrázolással szemben támasztott igénye ennél szerényebb, és jól megoldott.

A specifikációs nyelvek tervezôi rendszerint nem építenek be a nyelvbe kifejezett idôhivatkozásokat, hanem valamilyen erôsen egyszerüsítô feltételezést fogadnak el, pl. azt, hogy az események teljesen rendezettek. Néhány nyelv ugyan támogat maroknyi, idôvel kapcsolatos kifejezés-tipust, pl. olyanokat, hogy egy esemény "havonta egyszer következik be", de ennél jóvabb gazdagabb lehetôségekre van szükségünk ahhoz, hogy beszélhessünk az idôrôl és az idôre utaló viselkedésekrôl.

Ahogy szükség van szélsôségesen részletes eseményleírások megadására, ugyanúgy szükség van a relatív és az abszolut idô - "egy óra múlva" és "egy órakor" -, az ismétlôdô viselkedés - "a Föld mozgása minden 24 órában magában foglal egy tengely körüli fordulatot" -, és a bizonytalanság - "csütörtökön" - ábrázolására is.

1.3.4 Szemantika definiálása és következtetés

Általában azt szeretnénk, hogy a nyelveknek legyen ellentmondás-mentesség/ellentmondás fogalma, mellyel felismerhetjük a nyilvánvalóan helytelen specifikációkat.

Elônyösnek tünik, hogy a modellezô nyelvek jelentését valamilyen logikában fejezzük ki, mert

- a logikával készen kapjuk az "ellentmondás-mentesség", és esetleg többfajta "következtetés" fogalmát, valamint

- az automatizált következtetés területén tekintélyes munkát végeztek, különösen elsőrendű logikára épülő nyelvekkel, és ez a munka remélhetőleg beolvasztható a felhasználókat segítő számítógépes eszközökbe.

Logikai nyelv választására késztet az a meggondolás is, hogy a modellezés során olyan pontosak lehessünk, amilyenek csak akarunk, de ha a feladat úgy kívánja, lehessünk pontatlanok is. Ez ellentétes pl. a programozási nyelvek felfogásával, melyek viszonylag egységesen precíz specifikációkra rendezkedtek be.

Az alkalmazott logikai és leíró eszközök alapján a "szemantikaiság" fokára két általános osztályozási kritérium elfogadott:

- mi modellezhető, vagy mi fejezhető ki a szemantikai adatmodellben?, és

- mi vezethető le, vagy mire lehet következtetni a szemantikai adatmodellben?

Mivel a szabványos predikátum kalkulus kifejező erejét tekintve korlátozott, a szemantikai adatmodellek megfelelő formalizálása a szabványos predikátum kalkulus keretein belül - nyilván - szintén korlátozott.

Például, néhány jelenlegi szemantikai adatbázis rendszerben, így Abrial bináris modelljében és ennek Anderson-féle kiter-

jesztésében nem fejezhetô ki, hogy egy "i" egyed mind az "A", mind a "B" halmaznak eleme, ha "A" nem a "B" részhalmaza, és "B" sem az "A"-é. Azaz halmazbatartozás adott szintû term-re csak egyszer jelenthetô ki. Más rendszereknél problémát jelent a hamis és az ismeretlen igazság-értékek közti megkülönböztetés.

Egy szemantikai modell formalizmusának a következô, a szabványos predikátum kalkuluson felüli jellemzôkkel kell rendelkeznie:

- támogatnia kell a "sort"-okat vagy típusokat;

- legalább három igazság-értéket kell nyújtania: igaz, hamis és határozatlan;

- magasabb, legalább másodrendû formalizmusnak kell lennie, hogy a szabályos egyedek és predikátumok mellett predikátumok tulajdonságait és relációit is támogassa;

- támogatnia kell az olyan "modális" egyedeket, mint a "világok" és az "idôpontok", akkor is, amikor bevezetik a modellbe;

- támogatnia kell a létezô és a nem létezô egyedeket, az un. "null-objektumokat"; és végül

- támogatnia kell az ismert és az ismeretlen egyedeket.

Hogy a modellre épülő rendszer következtetéseket vonhasson le, azaz az adatbázisban kifejezetten nem tárolt információt vezethessen le, a modellhez egy következtető egység vagy szabály-alkotórész is tartozik.

Míg a tudásábrázoló rendszerekben általában nem tételezhetünk fel "apriori", mindenképpen igaz következtetési szabályokat, a szemantikai adatmodelleknél megelégszünk olyan módszerek kínálatával, amelyekkel a modellezési folyamatban felismert, biztosan igaz szabályok a modellbe építhetők úgy, hogy a szabályok megtalálása teljes mértékben a modellezőre hárul.

Annak, hogy a modellben következtetési szabályok adhatók meg, sok kézenfekvő előnye van. A legfontosabb talán az, hogy bizonyos adatok következtetési szabályként fogalmazhatók meg, és közvetlenül a modellbe építhetők. Ügyesen megválasztott szabályokkal jelentősen csökkenthetjük a fizikailag tárolt adatok mennyiségét, és gyors, kifejező lekérdező műveletekre nyílik lehetőség.

A szabályok olyan integritási kényszerek deklaratív kijelentésére is használhatók, melyeket az irodalomban tipikusan eljárásokkal jelentenek ki (pl. olyan n-esek törlése, melyek azonosítója nem létezik [DAT'83]).

1.3.5 Egyöntetű formalizmus és redundancia

Végül szeretnénk, ha a nyelv könnyen megtanulható és olvasható, valamint kényelmesen használható lenne. Ezt nagyban elősegíti az egyöntetűség. Célszerűnek látszik az olyan objektum-középpontú nézet, melyben az összes információt tulajdonságokkal kölcsönös kapcsolatba hozott, és osztályokba csoportosított objektumok segítségével fejezhetjük ki (TAXIS).

A nyelvek a teljesebb és pontosabb leírások elérésében is segíthetik a modellezőt. Ennek egyik eszköze a redundancia. Ha a modellező több szempontból látja ugyanazt a helyzetet, akkor kevésbé valószínű, hogy elhagy egy lényeges tényt. Ráadásul, ha egyazon helyzet többszörös leírásai nincsenek összhangban, akkor nyilván legalább egyikük helytelen. Ez a tény nem fedezhető fel, ha csak egyetlen leírás adott.

1.3.6 Absztrakciós elvek támogatása

A legfőbb szervezési kérdés az absztrakció [HE'86a], [HE'88b]. Egy valóságos rendszer specifikációja nagyon-nagy és részletes. Az eddigi tapasztalatok azt sugallják, hogy a sok részletre kiterjedő és bonyolult leírások kezelésének célravezető módja a széles körben használt absztrakciós elveken - az osztályozáson, aggregáción, általánosításon, és a csak újabban megfogalmazott asszociáción [SM'77a], [SM'77b], [ML'81], [BRO'84] - alapuló strukturált szervezés

támogatása. A modellezô nyelveknek tehát egyrészt irányvonalakat kell adniuk a pillanatnyilag fontos részleteket illetôen, másrészt a finomító folyamatot támogató nyelvi jellegzetességeket kell nyújtaniuk.

1.4 A tudásábrázolás néhány, a modellezésben is alkalmazott vagy alkalmazható eredménye

A mesterséges intelligencia-kutatások utóbbi két évtizedének egyik kulcseredménye, hogy a gépek csak akkor tanusítanak intelligens viselkedést, azaz csak akkor képesek természetes nyelven kommunikálni, képeket megérteni stb., ha sokat "tudnak" a valós világról. Éppen ezért sok mesterséges intelligencia-kutatás foglalkozott a tudás számítógépes ábrázolásával és szervezésével [HE'87].

Miután a világra vonatkozó tudás megragadása a fogalmi/szemantikai modellezés lényeges és nélkülözhetetlen momentuma, természetesnek tûnik, hogy a mesterséges intelligenciában megtanultakat a modellezésben is alkalmazzuk.

A tudásábrázolás eredményeire támaszkodva legalább részleges választ adhatunk olyan kérdésekre, amelyek éppen csak felvetôdtek a szoftver-szerkesztésben. Ez a terület elképzeléseket és technikákat kínál:

- sokféle tudás (a természetes világból származó fogalmak, az alapértelmezés szerinti tudás, a kivételek, az időre vonatkozó információ, a bizonytalanság és a hiányos tudás) ábrázolásához, valamint

- e tudás oly módon történő szervezéséhez, mely kényelmesen kezelhető mind az emberek, mind a számítógépek számára.

A mesterséges intelligencia tudásábrázolással kapcsolatos kutatásai temérdek ötletet adnak a fogalmi modellezéshez [KN'88].

1.4.1 Objektum-orientált megközelítés

A szemantikai adatmodellek többsége [BRO'84]* a mesterséges intelligencia szemantikai hálózat kutatásában gyökerezik.

A szemantikai hálózatok a tudást objektum-gyűjteményként modellezik. Az objektumokat rendszerint háromféle összekötő hozza kölcsönös kapcsolatba, melyek attribútumokat, résztípusokat és objektumbeli tagságot ábrázolnak.

---

* [BRO'84] az eddig kifejlesztett szemantikai adatmodelleket négy világosan elkülönített csoportba sorolja: klasszikus modellek közvetlen kiterjesztései; matematikai modellek; irreducibilis modellek; és szemantikai hálózati modellek.

Az objektum-orientált ábrázolás közvetlen, természetes megfeleltetést tesz lehetővé a modell és a világ között. A modell megszerkesztésekor kiválasztjuk az alkalmazási tartomány leírásaiban említett fogalmakat, és a modellben definiáljuk a megfelelő objektum-osztályokat. Az egyes objektum-osztályok mint központok köré szervezzük a kérdéses fogalomhoz kapcsolódó információt.

A szemantikai hálózatokban azonban nem válik ketté a séma és az adat; a primitív adat-elemek a tagság (membership) kapcsolattal közvetlenül a séma-alkotórészekhez kapcsolódnak. Egy adatbázisban viszont különálló sémára van szükség a nagy mennyiségű, hasonlóan strukturált adat kényelmes kezeléséhez. További eltérés, hogy a szemantikai hálózatok általában nem foglalnak magukba adatkezelő nyelvet.

Az objektum-orientált megközelítés a szemantikai hálózatokon kívül olyan programozási nyelvekben gyökerezik, melyek - mint a Simula [DAH'70], a CLU [LI'78] és a Smalltalk [GO'83a], [GO'83b] - támogatják összetett adatobjektumok definiálását [HE'88b].

A szemantikai modellezés és az absztrakt adattípusok alkalmazása között az egyik fő különbség az, hogy míg egy absztrakt adattípus operátor rendszerint csak egyetlen objektum-típushoz tartozik, addig a szemantikai modell rendszerint olyan operátorokat támogat, melyekkel közvetlenül összefüggésbe hozhatók a különböző típusú objektumok.

1.4.2 Általánosító hierarchiák

A szemantikai hálózatokból került a szemantikai modellekbe az osztályokat összefüggésbe hozó is-a kapcsolat is [HE'86a].

Az is-a kapcsolat az objektum-osztályokat a fogalmak olyan természetes taxonómiájába rendezi, amely elősegíti a leírások kezelhetőségét: a hierarchiában egymáshoz közel helyezve az osztályokat, előtérbe hozza a köztük lévő hasonlóságot. Az is-a kapcsolat az általánosításnak nevezett absztrakciós mechanizmus alapja. Eszerint gyakran hasznos, ha először szándékosan figyelmen kívül hagyjuk a különféle, de mégis összefüggésben lévő osztályok közti különbségeket, és inkább valamilyen általánosabb szuper-osztály leírásaként adjuk meg közös aspektusaikat. Az osztályok közti különbségek akkor vezethetők be, amikor a modellező leírja, hogy az egyes osztályok - mint részosztályok - miképpen különböznek mindannyiuk közös szuper-osztályától.

Az általánosítás egyben sajátos fejlesztési módszertant is sugall, éspedig a specializálással történő lépésenkénti finomítást. Először a fizikai világban előforduló, odaillő entitások, tevékenységek és állítások legáltalánosabb osztályait kell leírni. A következő fázisban kiválasztjuk és leírjuk az egyes osztályok fontos részosztályait. A finomítási folyamat következő lépéseiben azután egyre kisebb és kisebb részosztályokat vezetünk be és írunk le, melyek egyre speciálisabb fogalmakat modelleznek.

A tulajdonságok öröklésének fogalmát látszólag mindig az is-a kapcsolat velejárójaként említik. Az öröklés azért hasznos, mert megengedi, hogy a modellező a specifikációit a részosztálynak a szuper-osztálytól való eltérésére korlátozza. Így az egyes lépéseknél csak a szinthez illő információt kell figyelembe venni.

Az általánosítás akkor igazán hasznos absztrakciós elv, amikor a modellezés nehézségeit nem annyira az algoritmikus bonyolultság, mint inkább a modellbe tartozó objektumok és részletek sokasága okozza. A rendszertani hierarchiákat végül is régóta használják más diszciplinákban, pl. a növénytanban és az állattanban sok megfigyelés szervezésére.

Egy tevékenység előírásának problémái a legtöbb esetben a tevékenységben résztvevő objektum vagy objektumok változataival függenek össze. Megfigyelhetjük, hogy sok szoftverfejlesztő projekt, de különösen az információs rendszer projektek, pontosan ilyen helyzetekkel foglalkoznak.

Ez a módszertan független a jólismert "dekompozicióval történő lépésenkénti finomítástól", és egyben kiegészíti azt. Specializációval ugyanis egy megoldás-osztály változatai vezethetők be a dekompozíció bármely szintjén, beleértve a végső szintet is, a programozási nyelv szintjét.

A specializációval történő finomítás érdekes példa arra, hogy milyen előnyökkel járhat a mesterséges intelligencia és a szoftver-szerkesztés fogalmainak - ezúttal az öröklődési

A kifogás-mechanizmus előnye egy további absztrakciós elv támogatása, nevezetesen, hogy először a szokásos vagy "normális" esetet írjuk le, később finomítva ezt a leírást a speciális vagy kivételes esetek feltüntetésével [BOR'84b].

1.4.4 Időábrázolás

Mivel szinte valamennyi kijelentésünk elmaradhatatlan része az idő - egy objektum bizonyos időben létezik; egy tevékenység bizonyos időpontban kezdődik illetve ér véget; egy állítás bizonyos időben igaz stb. -, egy világ-orientált modellező sémában is fontos szerepet játszik.

Az idő és az idővel kapcsolatos információ ábrázolása is olyan terület, amit lényegileg nem kutattak a szoftverszerkesztésben, és ahol ismét hasznot húzhatunk a mesterséges intelligencia területén végzett munkákból.

A bonyolultabb rendszerek, mint az időbeli (temporal) vagy feszített (tense) logikák (pl. [RES'71], [SER'80]), vagy az intenzionális logika (pl. [MON'73]) nem nyújtanak nagyobb kifejező erőt, csak hatékony jelölést. Ezek a logikák bonyolultságuk folytán - beleértve ezen tartományokban jelentős automatikus tételbizonyító képességek hiányát is - a modellezés céljára talán nem a legmegfelelőbbek.

A változások kezeléséhez az elsőrendű logikát hagyományosan

Másik megoldásként új részosztályok - pl. **repülő madarak** és **nem repülő madarak** - bevezetése jöhet szóba. Ez viszont a többnyire érdektelen osztályok számának kombinatorikus robbanását eredményezheti.

Végül is az a problémánk, hogy miközben megpróbálunk a lehető legáltalánosabbak lenni, nincs olyan elv, mellyel eldönthetnénk, hogy milyen messzire menjünk.

Tegyük fel, hogy a székeket akarjuk leírni egy kijelentéssel. Hogyan jellemezhetjük a székeket? A székeknek rendszerint négy lábuk van, de némelyiküknek csak három, és biztos, hogy készítettek már széket öt, hat, sőt több lábbal is.

Módszertani szempontból tekintve a kivételek kétségtelenül üröm az örömben, hiszen a finomítás egyik szintjén mondottak bármelyike végül is megcáfolható valamely alacsonyabb szinten.

Egy lehetséges megoldás, ha a tervezőnek megengedik kivételes osztályok vagy objektumok előírását. Ezekre a kivételes osztályokra vagy objektumokra kifejezetten elismertek az ellentmondások, és ezeket "kifogásokon" keresztül oldják fel. Természetesen a kifogásokhoz megfelelő szemantikát kell adni, hogy a végső specifikáció logikailag ellentmondásmentes legyen. Ennek megvalósításába pl. a nem-monoton logikával foglalkozó mesterséges intelligencia-rendszerek nyújthatnak bepillantást.

sémák és az absztrakció - házassága.

1.4.3 Kivételek kezelése

A specifikációk egyik legfontosabb jellemzője az ellentmondásmentesség. Sajnálatos módon a természetes világ leírásában gyakran csak nagyon nehezen kerülhetők el az ellentmondó kijelentések.

A kivételes helyzetek ábrázolását alaposan tanulmányozták a mesterséges intelligenciában [HE'87]. Klasszikus eset a következő probléma: "a madarak repülnek", "a pingvinek madarak", de "a pingvinek nem repülnek".

Az ilyen és ehhez hasonló példák a túlabsztrahálás eseteinek tűnnek. Ilyenkor kezdetben úgy hagyjuk figyelmen kívül a "jelentéktelen" részletek eltéréseit, hogy azok később az ellentmondásmentesség alááásása nélkül nem vezethetők be. Ennek az az oka, hogy a legtöbb fogalomnak nincs pontos definíciója, és csaknem minden szabály alól vannak kivételek.

Ellentmondással szembekerülve visszatérhetünk az eredeti leíráshoz és általánosabbá tehetjük - pl. nem mondunk semmit a madarak repüléséről -, így elkerüljük az ellentmondásokat. Ez számos hátránnyal jár. Az öröklés folytán egy ilyen változtatás sok olyan részosztályt is érinthet, melyre az eredeti állítás helyes és hasznos volt (a seregélyek, a verebek, a gólyák stb. mind repülnek).

a helyzeti kalkulussal (situational calculus) terjesztik ki [MCA'68]. Egy pótargumentummal bővül minden időfüggetlen függvény és predikátum, az eseményeket pedig az általuk "összekapcsolt" helyzetekre vonatkozó predikátumokkal írják elő. Ezen az általános kereten belül számos alternatíva adódik a "helyzetek" természete szerint.

Az időt vagy a változást alapvető tényként elfogadva még mindig számos kérdés marad nyitva a modellben előforduló objektumok és események természetével kapcsolatosan. Ilyen kérdés az, hogy az időt statikus pillanatfelvételek sorozataként tekinthetjük-e vagy sem. Ha igen, akkor mindig csak a "következő állapotról" kérdezhetünk. Ehhez rendszerint az a feltételezés vagy általános "frame axioma" rögzül, hogy az érdeklődési tartományban nincsenek változások, hacsak nem valamely előírt esemény okozza azt. Egy specifikációs nyelvben talán helyénvaló lenne megkövetelni, hogy egy ilyen feltételt a modellező kényszerként fogalmazzon meg, hiszen ez korlátozza a lehetséges megvalósításokat.

Az eseményeket "állapotváltozásokként" (állapot-párok halmazaiként) modellezve nem beszélhetünk arról, ami egy esemény előfordulása alatt történhet, és többek között nehezebbé válik az is, hogy az esemény-előfordulások időtartamáról beszéljünk.

Az idő metafizikusan lineáris nézetét elfogadva választhatjuk az időpontok mutatókként való használatának maradi, de viszonylag jól megértett módszerét (pl. [GR'86]), vagy az

intervallumok primitívekként történő alkalmazását, amint azt [ALLE'83] javasolja. Ez utóbbi megközelítés elkerül néhány kényes kérdést, melyek arra vonatkoznak, hogy mi történik azokban az időpontokban, melyekben az egyik esemény elkezdődik, mihelyst egy másik véget ér.

Az időt időpontok részben-rendezésének is tekinthetjük, ahogy [MD'81]-ben, [MCAW'81]-ben és a feszített (tense) logikában teszik, számos lehetséges jövővel, melyek mindegyike teljes rendezés. Ez akkor előnyös, ha olyan formális rendszerrel foglalkozunk, melyben bizonyos célok elérésérôl lehet vitatkozni (pl. tevékenységek tervezésekor, meggyőződések felülvizsgálatakor vagy jövőbeli lehetőségre utaló angol mondatok jelentésének ábrázolásakor).

1.4.5 Formális szemantika definiálása és következtetési szabályok beépítése

A mesterséges intelligencia kutatói fontos alapozó munkát végeztek a reprezentációs sémák formális szemantikájával kapcsolatosan is, valamint olyan tárgykörökben, mint a tudásbázison belüli következtetés és logikai ellentmondásmentesség.

A formalizálás folyamata először is azért jelentős, mert a kiokoskodott dolgok pontos jelentésének részletes végiggondolására kényszerít, és gyakran kétértelműségek, aszimmetriák felfedezéséhez vezet.

Másodszor, ha egy specifikációt viszonylag szabványos predikátum logikába lehet fordítani, akkor azt logikai elméletként is tekinthetjük, amely esetleg olyan leírás-strukturáló képességekkel rendelkezik, amilyet a csak logikán alapuló nyelvek nem nyújtanak. Az ilyen modell ellentmondásmentessége a logikai ellentmondásmentesség szabványos fogalma, és így a modellben levezetések és következtetések végezhetők. Ez lehetőséget ad arra, hogy számítógépes tételbizonyítót használhassunk a modell ellentmondásmentességének, vagy akár pl. annak eldöntésére is, hogy egy bizonyos helyzet bekövetkezhet-e a világban vagy sem.

Végül a nyelv felhasználói és olvasói végső döntőbíróként fordulhatnak a formális specifikációhoz azokban az esetekben, amelyekben valamely specifikáció pontos jelentését illetően nézeteltérés van.

A szabványos predikátum kalkulus kifejező ereje azonban túlságosan korlátozott ahhoz, hogy kifejezze

- az időpontok;
- az állított, tagadott és ismeretlen "tények";
- a létező, nem létező (null-) és ismeretlen objektumok;
- a közös, eltérő és ellentmondó adatu lehetséges világok fogalmait.

A szabványos predikátum kalkulus kifejező erejének hiánya hozta létre a "kiterjesztett rendszereket", melyek további jellegzetességek szabványos predikátum kalkulusba történő olvasztásának eredményei. E kiterjesztések túlnyomó része "many sorted" logikán alapuló szemantikai adatmodell. E logikákban az egyedekhez és a formalizmus predikátumaihoz velük együtt előforduló előírások ("sort"-ok vagy típusok) fűzhetők, hogy csökkenjen az "értelmetlen" kijelentések száma. A "many sorted" logikán alapuló szemantikai adatmodell kifejező ereje ennél fogva nem nő: egy "many sorted" predikátum kalkulus ekvivalens a szabványos predikátum kalkulussal.

Évekkel ezelőtt a tudás-alapu mechanikus fordításon és a természetes nyelvű intelligens válaszadáson dolgozva felismerték a logika bizonyos, a "szabványos" elemi logikától eltérő ágainak létezését - az intenzionális logikákat, a modális logikákat és a valószínűség logikákat -, melyek megengednek elemi (predikátum) logikában kifejezhetetlen, vagy csak "eljárásosan", az elemi logikával vegyítve kifejezhető kijelentéseket.

Így valójában a predikátum kalkulus hiányos kifejező erejének tulajdonítható a megközelítések "deklaratív", "eljárásos" és "hibrid" rendszerekre történő hármas felosztása a tudásábrázolásban és a szemantikai modellezésben.

Stachowitz [STA'85]-ben érvekkel támasztja alá, hogy a sze-

mantikai modell kifejezô és következtetô ereje egyaránt meghatározható csupán formális vagy szintaktikai kritériumok alapján. Míg a szemantikai adatmodell kifejezô erejét a benne alkalmazott logikai formalizmus azon termjei tükrözik, melyek egyszerû - univerzális vagy specifikus, állított vagy tagadott - mondatokkal használhatók; következtetô erejét a modell szabály-alkotórészében lévô szabályok szintaktikai struktúrája, és a szabályokban elôforduló logikai összekötôk jellemzik.

Stachowitz azt is megmutatja, hogy a Tsichritzis és Lochovsky által felsorolt szemantikai jellegzetességek [TSI'82] e formális vagy szintaktikai kritériumokkal magyarázhatók.

1.4.6 A tudásbázis idôszerûségének megôrzése

A mesterséges intelligencia egyik központi kérdése a tanulás, a viselkedés tudás-szerzés következtében történô megváltoztatása.

Sok kutatás kapcsolódik a tudás-szerzés témaköréhez, olyan rendszerek építésével kísérletezve, amelyek automatikusan napra kész állapotra hozzák a formalizmusokat (pl. [WIN'75], [LA'83]).

A tudásábrázolás módja meghatározhatja, hogy egyáltalán elôfordulhat-e tanulás, és ha igen, akkor mit tanulhat a rendszer - kiértékelô függvényeket, új szabályokat vagy

keresési heurisztikákat. A tanulással kapcsolatosan kulcsfontosságú az ábrázolás két vonatkozása:

- a megtanultak modularitása, és

- a bemenő adatokhoz szükséges, kézzel végzendő munkák mennyisége.

A modularitás és a tanulás közti kapcsolat azért fontos, mert a tanulás tudás hozzáadásával/változásával jár, és ezt úgy kell végrehajtani, hogy az új tudás jó kölcsönhatásban legyen a meglévő tudással. A modularitás azt is megengedi, hogy a tudást különféle szakértők kezeljék.

Bár az un. production vagy szabályokon alapuló rendszerekkel való megközelítés nem az egyetlen modularitást adó formalizmus, mindenesetre azon kevés ábrázolási sémák egyike, melyet e vonatkozásban sikeresen kipróbáltak. Kevert ábrázolásmódra példa az RX [WIE'84].

1.5 A "modularizáció"

A mesterséges intelligencia kutatások korántsem fedik le a fogalmi modellek szerkesztésének minden lényeges problémáját.

Hogy a fogalmi séma könnyen olvasható és jól érthető legyen, hogy ki lehessen ragadni a felhasználót érdeklő viszonylag

kis részét - esetleg nem is teljes részletességgel -, "dekompozicióra", "modularizációra" van szükség.

A szoftver-szerkesztésnek ez a kulcsfontosságú területe a mesterséges intelligenciában nem merül fel. A programokat "top down" dekompozicióval ("strukturált programozással") vagy modularizációval ("absztrakt adattipusokkal") vagy ezek kombinációjával bontjuk kezelhető méretü részekre.

Fogalmi sémák esetén jóval bonyolultabb a helyzet, ezért az egyszerü dekompozició rendszerint nem kielégitő. A fogalmi sémákat ugyanis az egyes felhasználók érdeklődési területei szerint, tehát sok különböző szempontból lehet részekre bontani.

További eltérés, hogy míg a programozás lényegében konstruktív folyamat, a világ-orientált modellezés inkább leíró tevékenység. Bár egy program néhány szempontját előre kikötik, a programozó a legtöbb esetben szabadon tervezheti meg a program felépitését, az általa előre kiválasztott, jól strukturált módon. A modellezés eredményeként kapott fogalmi séma moduláris és hierarchikus szerkezetét viszont nem az információ-elemző választja. Ellenkezőleg, az információelemző feladata, hogy felfedezze a fogalmi séma hierarchikus és moduláris szerkezetét. Míg egy program tetszés szerint átstrukturálható, a helyes fogalmi sémán csak apró módositások végezhetők egy bizonyos struktura előirásához.

1.6 Az értekezés áttekintése

1.6.1 A kitűzött cél

Ezen értekezés célja, hogy az eddigi kutatások ([KN'86a], [KN'86b], [KN'86c], [KN'87]) eredményeit összefoglalva rögzítse egy olyan számítógéppel támogatott tervezési módszertan körvonalait, amely a modellezést az eddig kifejlesztett technikákkal szemben már a követelmény-elemzés első mozzanataitól segíti egy, a tervezési szinteknek megfelelő modell-hierarchia kezelésével.

E módszertan az integritás megőrzése mellett lehetővé teszi a "fogalom bázis" interaktív módon történő építését, akár pl. példával szemléltetett fogalmak megváltoztatását, sőt törlését is. Ugyanakkor a módszertan megfelelő leképező mechanizmussal alkalmassá tehető arra, hogy az egyes fogalmakat a különböző tervezési szinteken eltérő absztrakciós mértékben kezelje.

Egyúttal e módszertan elméleti megalapozását is szeretnénk megadni.

E módszertan gyakorlati használatának kulcsa a benne rejlő dinamika, hiszen a modellezés interaktív folyamat.

- Lehetővé teszi, hogy egy rendszer fogalom-hierarchiáit olyan fejlődő tudás-szerkezetnek tekinthessük, amely az emberi agyhoz hasonlóan folyamatosan finomítható és fejleszthető a rendszer élettartama alatt, interaktív módon. Ezt a jelenlegi modellek nem engedik meg.

- Alkalmassá tehető a különböző tervezési szinteken is megjelenő fogalmaknak a szintek szerint eltérő absztrakt kezelésére.

1.6.2 Az értekezés felépítése

A módszertan ismertetését - néhány szemléletes példa kíséretében - a 2.Fejezet tartalmazza. A különböző szintű modellek formális szemantikáját a 3. Fejezet tárgyalja, a 4. Fejezet pedig egy adatbáziskezelő rendszer tervezésének példáján szemlélteti a módszer alkalmazását. Végül az 5. Fejezet rövid összegzése a további munka lehetséges irányaira is utal.

2. Fejezet

ÚJ MÓDSZEREK A FOGALMI MODELLEZÉS SZÁMÍTÓGÉPES TÁMOGATÁSÁRA

Ebben a fejezetben bemutatjuk a fogalomalkotás folyamatát segítő (modellező) fogalmak top-down sémáját. A modellezési rétegek között olyan hierarchia kialakítását ajánljuk, amely megfelel a probléma-felvetés és a rendszerfejlesztés természetes folyamatának. Végül megvizsgáljuk, hogy e fogalomstruktúrák milyen feltételek mellett alakíthatók át úgy, hogy megőrizzék összes integritási tulajdonságukat - akár annak ellenére is, hogy valahány konkrét megvalósulással, példánnyal (instanciával) már rendelkeznek.

2.1 Az evolúciós rendszertervezés előnyei

Az adatmodellezés közismert paradoxonja, hogy

- alkalmazási szinten - szembekerülve a "valóság töredékeivel" - olyan részleteket érzékelünk, melyek általában ábrázolhatatlanok;

- reprezentációs szinten - szembekerülve a "gépek szintjeivel" - olyan részleteket ábrázolunk, melyek általában nem észlelhetők.

Az absztrakciós módszerek az adott összefüggésben szükségtelen részletek elnyomásával, és a lényeges információ formalizálásával, strukturálásával segítenek abban, hogy megbirkózhassunk ezzel a problémával.

Mivel az emberi észlelés tökéletlen, és csak bizonyos korlátok között vagyunk képesek a valóság teljes és pontos ábrázolására, bármely ábrázolás a valóság absztrakciója. Következésképp az absztrakció "jósága" alkalmazhatóságától függ.

Olyan modellt szeretnénk, amiben tetszőleges új információ asszimilálása nemcsak lehetséges, de természetes is. A legtöbb alkalmazásban, még ha a kezdeti tervet szigorúan top-down módon kidolgozva kapnánk is meg, a rákövetkező változások (pl. új adózási törvények következtében szükségessé váló elszámolási változások) megkívánják, hogy a rendszert rugalmasan lehessen módosítani.

Miután nincs elv, ami megmondaná, hogy az absztrakció adott szintjén mi az, ami valóban lényeges, és mi az, ami elhanyagolható, könnyen tévedhetünk. Tévedéseinket pedig sokszor csak a gyakorlati tapasztalatok alapján ismerjük fel, amikor fogalmaink már konkrét példányokkal rendelkeznek.

Evolúciós rendszertervezést megengedve ugyanolyan szigorúsági szabványokat érhetünk el, mint top-down tervezésnél, de anélkül, hogy megkövetelnénk a fogalmak előre meghatározott sorrendben való definiálását.

Az információs rendszerek tervezésének három modellezési állapotát vizsgáljuk, éspedig

- a feladatgyűjtés (problem acquisition) szintjén történő modellezést,
- a fogalmi szinten történő modellezést, és
- a logikai szinten történő modellezést.

2.2 Modellezés a megismerés szintjén

A ma közismert fogalmi tervezési módszertanok nem tűzik ki célul a követelmény-elemzés és követelmény-specifikáció kialakításának támogatását.

A tervezés e kezdeti állapotában még szinte minden bizonytalan, "cseppfolyós", napról napra változó. Ennek megfelelően az ebben a tervezési fázisban alkalmazható fogalmi eszközöknek

- kellőképpen általánosaknak (meghatározatlanoknak);
- meglehetősen hajlékonyaknak; és

- a fogalmi, valamint a logikai modell tervezésénél jelenleg javasolt és használt technikákkal jól összeegyeztethetőknek, illetve azokhoz jól csatlakoztathatóknak kell lenniük.

Az általánosság - azon túlmenően, hogy kezdetben sosem bölcs dolog a túlzott pontosság - azért fontos, hogy a fogalmak "bármely" irányba specializálhatók legyenek. A hajlékonyság pedig az ábrázolt modellek dinamikus változtatását (módosítását, átalakítását) teszi lehetővé [KN'86c].

2.2.1 Részosztály kontra példány

A jelenleg javasolt technikákat alkalmazva egy tipikus problémával kerülünk szembe az "is-a" és "instance-of" kapcsolatokat illetően. Tegyük fel, hogy egy olyan fogalmi nyelvet használunk, amely megengedi a következő állítások írását:

```
concept adó;
  terhelt_jövedelem_forrása: jövedelem;

concept általános_forgalmi_adó;
  instance-of adó;
  terhelt_jövedelem_forrása: fogyasztás;

concept személyi_jövedelemadó;
  instance-of adó;
  terhelt_jövedelem_forrása: egy_éven_belüli_jövedelem;
```

Bár ennél az egyszerû példánál látszólag minden rendben van, mégis meggyûlik vele a bajunk, ha definiciónkat további attributum hozzáadásával finomítani próbáljuk, mondjuk a következõképpen:

```
concept adó;
  terhelt_jövedelem_forrása: jövedelem;
  adókulcs: integer;
```

Ebben az esetben ugyanis az "általános_forgalmi_adó" és "személyi_jövedelemadó" fogalmakat nem tekinthetjük egyszerû példánynak, hiszen az absztrakció adott szintjén az "adókulcs" attributum értékét nem tudjuk adatszerûen megadni. Ezért elõnyösebb, ha ezeket úgy modellezzük, hogy

```
concept általános_forgalmi_adó;
  is-a adó;
  constraints
  terhelt_jövedelem: terhelt_jövedelem_forrása = fogyasztás;

concept személyi_jövedelemadó;
  is-a adó;
  constraints
  terhelt_jövedelem: terhelt_jövedelem_forrása =
                                    egy_éven_belüli_jövedelem;
```

Tovább is mehetünk. A "személyi_jövedelemadó" szintjén bevezethetünk olyan további attributumokat, mint pl. "adóalap" és "adómentes_jövedelem". Ekkor pl. az "500.000 Ft alatti

jövedelemmel rendelkező mezőgazdasági kistermelők adója" bizonyos aspektusból ismét modellezhető konkrét példányként, de ez ellen szólnak más megfontolások, melyek a résztípus ábrázolást sugallják.

Általában levonhatjuk azt a következtetést, hogy - legalább is a feladatgyűjtés szintjén történő modellezéskor - rendszerint nem dönthető el százszázalékos biztonsággal, vajon bizonyos fogalmak konkrét példányok-e, vagy éppen sajátos részosztályok. Tehát még a szemantikai hálózatok [QU'68], [BRA'79] fogalmainál is gyengébb fogalmakra van szükségünk.

Megoldásként egy általánosított, új - általunk case-of kapcsolatnak nevezett - kapcsolat bevezetését javasoljuk, amely a modellezés későbbi fázisaiban majd "is-a" vagy "instance-of" kapcsolattá finomítható.

2.2.2 Fogalom-azonosítás

Már a rendszerfejlesztés legelején, a kezdeti modell szerkesztésekor kifejezetten megkülönböztethetjük a név szerint és a kulcs szerint azonosított fogalmakat. Azaz [CH'76] értelmében különbséget tehetünk az entitások és a kapcsolatok között.

2.2.3 A tulajdonságok tipusai

Két fő tulajdonság-tipust különböztethetünk meg a relációs adatbázisbeli attributum és kényszer fogalmához hasonlóan, éspedig azokat a tulajdonságokat, melyekkel rendelkeznek tulajdonosaik, és azokat, amelyeket teljesítenek. Ezeket, a hagyományokat követve rendre "attributumoknak" és "kényszereknek" nevezzük [UL'82].

2.2.4 A modellszerkesztés kezdeti fázisában használt fogalmak

A legfelső szinten kiindulásként csupán néhány fogalom használatát ajánljuk, éspedig:

- a "kategóriát", mint a dolgok, jelenségek legfőbb jegyeiből kialakított fogalmat;

- az "entitást", amely név szerint azonosított "kategória" [CH'76];

- a "kapcsolatot", amely kulcs szerint azonosított "kategória" [CH'76];

- a "fogalmat" (nem téve még különbséget "absztrakt" és "konkrét" fogalmak között), mely "tulajdonságokkal" jellemzett "entitás";

- a "tulajdonságot", melynek szintjén még nem teszünk különbséget az "attributumok" és "kényszerek" között;

- az "attributumot", mint olyan tulajdonságtípust, mellyel a jellemzett "fogalom" rendelkezik [UL'82];

- a "kényszert", amely a jellemzett "fogalom" által teljesített tulajdonság [UL'82]; végül

- a "case-of" relációt, amely ezen a szinten a szokásos "is-a" és "instance-of" kapcsolatokat helyettesíti.

Értelmezés:

(i) A "kategóriát" az azonosítási kényszer és a hozzátársított (esetleg üres) tulajdonsághalmaz adja meg.

(ii) "Tulajdonság" bármi lehet, pl. egy természetes nyelvű mondat, amit a fogalom definiálásához szükségesnek tartunk megemlíteni. Nevük - a jellemzett fogalmat illetően - lokálisan azonosítja őket.

(iii) Minden attributumnak van egy értéke, amely feltételezés szerint egy fogalom, olyan előre definiált fogalmakat is megengedve, mint "szám", "szöveg", "név" stb. Az attributumoknak lehet módjuk is, pl. "individual", "set", "list", array", "n-tuple"

stb. A többi elérhetô technika nem vezet be attribútum-módokat. Ezeket a "másodlagos absztrakciós mechanizmusok" kifejezésére ajánljuk, amilyen az aggregáció, az asszociáció stb.

(iv) A "case-of" reláció fogalompárok összekapcsolására szolgál, valahányszor e pár elsô tagja valamilyen értelemben rendelkezik a második összes tulajdonságával. A lényeges az, hogy egy "case-of" kapcsolat esetén ne kelljen eldöntenünk mit is takar ez a kapcsolat valójában, mielôtt még eléggé megértenénk a modellezett jelenséget és a helyénvaló, alkalmazandó szempontot.

2.2.5 A kezdeti fázisban használt modellezô fogalmak szemiformális ábrázolása

A "kategóriát", "kapcsolatot", fogalmat", "tulajdonságot", "attribútumot", "kényszert" és a "case-of" relációt az itt bevezetett értelemben "fogalmaknak" is tekinthetjük, amit a következôképpen fejezhetünk ki:

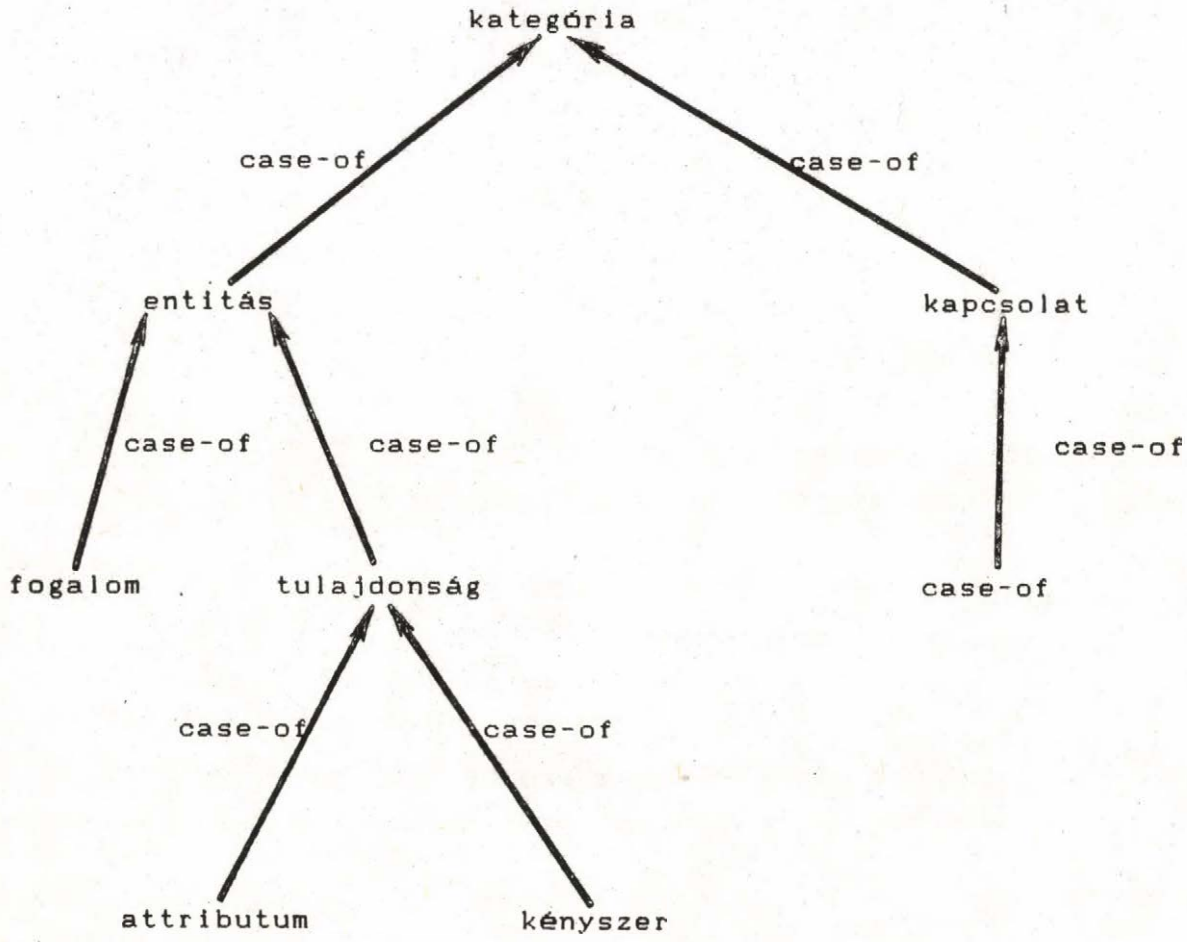
```
notion kategória;
  attributes
    azonosító: fogalom;
  constraints
    azonosítható:  az "azonosító" attribútum azonosítja
                   a kategóriát;
```

```
notion entitás;
  case-of kategória;
  attributes
    azonosító: név;

notion kapcsolat;
  case-of kategória;
  attributes
    azonosító: n-tuple-of fogalom;

notion fogalom;
  case-of entitás;
  constraints
    definiált: tulajdonságokkal jellemzett;

notion tulajdonság;
  case-of entitás;
  constraints
    definiál: van olyan fogalom, amit jellemez,
    lokálisan_azonosított: az "azonosító" attribútum a
                           jellemzett fogalmat illetően
                           lokálisan azonosít;

notion attributum;
  case-of tulajdonság;
  attributes
    érték: fogalom,
    mode: {individual, set-of, list-of, array-of,
           n-tuple-of};
```

notion kényszer;
  case-of tulajdonság;
  attributes
    érték: állítás;
  constraints
    teljesülnie_kell: az "értéknek" a jellemzett fogalom esetén teljesülnie kell;

notion case-of;
    case-of kapcsolat;
    attributes
      azonosító: 2-tuple-of fogalom;
    constraints
      olyan: fogalom$_1$ valamilyen értelemben rendelkezik fogalom$_2$ valamennyi tulajdonságával,
      körútmentes: ismételt alkalmazása körútmentes irányított gráfot eredményez;

Így az általunk javasolt fogalmak és a köztük lévő kapcsolat a következőképpen ábrázolható:


kategória
case-of
case-of
entitás
kapcsolat
case-of
case-of
case-of
fogalom
tulajdonság
case-of
case-of
case-of
attributum
kényszer

2.3 A fogalmi modellezés hagyományos szintje

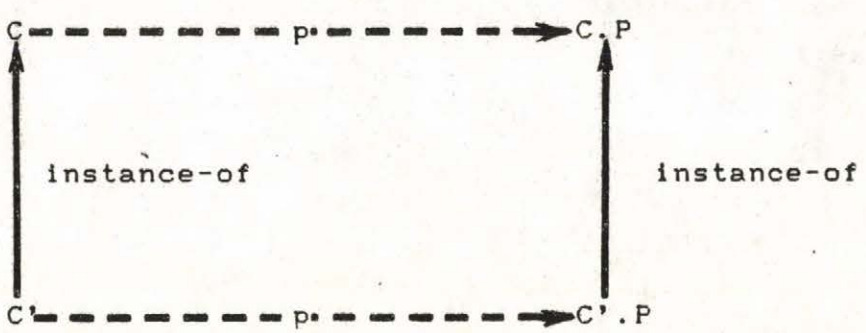
Ha már rendelkezünk egy jelenség (rendszer) felsô szintû, kezdeti modelljével, akkor ezt a modellt fokozatosan finomítanunk, gazdagítanunk kell. Ezek az átalakítások azonban távolról sem mechanikusak, mivel az absztrakciós szintek nemcsak részletességük mértékében, hanem természetüknél fogva fogalmi alapjaikban is eltérnek, ld. pl. [LUD'84]. Ennek következtében a finomítás nemcsak tervezési döntések meghozatalát, hanem új információ gyûjtését, és ennek modellbe történô integrálását is jelenti. Elengedhetetlen tehát, hogy a különféle rétegekhez tartozó fogalmi alapok összeegyeztethetôségével a modellezési szintek között "sima" átmenetet biztosítsunk.

A korábban körvonalazott felsô szintû modell - nyitottsága következtében - számos sajátos irányba gazdagítható és csiszolható. Egy ilyen finomítás során kapjuk a jól ismert fogalmi modellek (pl. TAXIS [MY'80b], DAPLEX [SH'81], Galileo [ALB'85]) egyikét. A következôkben ezt az utat vázoljuk fel.

2.3.1 Osztályok és példányok

Az osztályozás a legfontosabb és történetileg elôször természetesnek tartott absztrakciós mechanizmus. A SIMULA 67 [DAH'70] megszületése óta széles körben használják, és valamilyen formában az összes fogalmi nyelv alkalmazza. (Az osztály-definiciókat itt nem idézzük, ezek sokhelyütt, pl. az általunk hivatkozott mûvekben is megtalálhatók.)

Az osztályozás értelmében az egyik fogalom a másik példányának mondható, ahol az utóbbi az "absztrakt" az elôzôhöz viszonyítva. Következésképpen az "instance-of" kapcsolat az általunk bevezetett "case-of" kapcsolat speciális eseteként tekinthetô, amely egy fontos tulajdonsággal, az attributumkiválasztás "homomorfizmus szabályával" [KN'86b] rendelkezik. Nevezetesen, legyen p a C fogalom egy attributuma, és értékét jelölje C.P. Ha C' a C egy példánya, akkor C'.P-nek a C.P példányának kell lennie:



Az, hogy C'-nek is rendelkeznie kell a p attribútummal, a "case-of" kapcsolat definíciójából következik. Szemi-formális jelölésünkkel:

notion instance-of;
  case-of case-of;
  constraints
    homomorfizmus: fogalom$_1$ attribútumai fogalom$_2$ megfelelő attribútumainak példányai;

(Vegyük észre, hogy az "instance-of" fogalom azonosítója a (fogalom$_1$, fogalom$_2$) rendezett pár, ahogy ezt már a "case-of" esetén definiáltuk.)

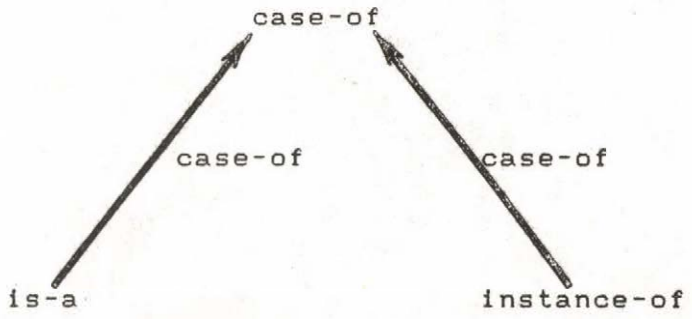
2.3.2 Az általánosítás

A második legfontosabb absztrakciós mechanizmus a rendszerint "is-a"-val jelölt általánosítás, amelyet az osztályozáshoz hasonlóan már a SIMULA 67-ben is megvalósítottak. Ennélfogva:

notion is-a;
  case-of case-of;
  constraints
    specializálás: fogalom$_1$ speciális esete fogalom$_2$-nek;

2.3.3 A "case-of" esetei

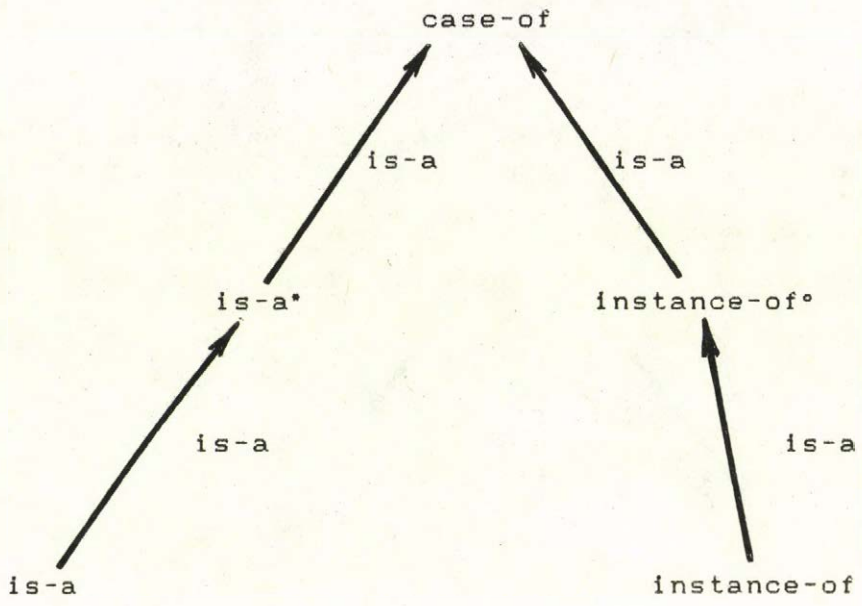
Az előzőek alapján a következő ábrát vázolhatjuk fel:



Jelölje "is-a*" az "is-a" tranzitív lezárását, és definiáljuk az "instance-of°" relációt a következőképpen:

$$\text{instance-of}^{\circ} = \text{instance-of} \circ \text{is-a}^{*}$$

ahol o a relációk "kör szorzatát" természetes join-ját jelöli [KN'86b]. A fogalmi modell kialakítása során természetesen sor kerülhet új fogalmak bevezetésére is, így előfordulhat, hogy egy case-of kapcsolatot is-a* vagy instance-of° fogalomlánccá finomítunk. A tranzitív kiterjesztést figyelembe véve, és a case-of relációt tranzitívnak feltételezve ábránk tovább finomítható:



Ennélfogva amikor úgy általában, azaz az instance-of° értelmében beszélünk a konkrét példányokról, "A is-a° B" esetén A példányainak halmaza szükségképpen részhalmaza a B példányai által alkotott halmaznak. Ez vitatható következményekhez vezethet, nevezetesen az "is-a" kapcsolat definiálható a példányok halmazára vonatkoztatott részhalmaz (subsetting) kritériummal. Ugyanakkor a világosság és helytállóság miatt fontos, hogy absztrakt szinten, azaz a később létrejövő, vagy talán sosem létező példányokra történő utalás nélkül beszélhessünk a fogalmak közötti "is-a" relációról.

2.3.4 Univerzumok

Még nem tettünk semmilyen megszorítást az "is-a" és "instance-of" kapcsolatok használatára - eltekintve a "case-of" szintjén deklarált általános körültmentességtől. Számos lehetőség nyílik olyan korlátozás bevezetésére, hogy értelmes és rendezett modellt kapjunk.

A példányként való deklarálás egyedisége:

Egy A fogalomról csak egyszer jelenthetjük ki, hogy "A instance-of B". Ezt a feltételezést minden ismert fogalmi modell alkalmazza. Tehát

$$\left.\begin{array}{l} A \text{ instance-of } B \\ \underline{\text{és}} \\ A \text{ instance-of}^{\circ} C \end{array}\right\} \Longrightarrow B \text{ is-a}^{*} C$$

beleértve a B = C lehetőséget is.

Több "absztrakciós szint" feltételezése:

Definiáljuk először az "absztrakciós szint" fogalmát a következő intuitív módon. Legyen L_1 azon fogalmak halmaza, melyek nem más fogalmak példányai. Legyen továbbá L_k az L_{k-1}-hez tartozó fogalmak példányainak halmaza.

Ekkor

(i) L_k az egész fogalomhalmaz feletti osztályozás;

(ii) Ha "A is-a B", akkor A és B ugyanazon absztrakciós szinthez tartozik;

(iii) Egy fogalom attributum-értékeinek ugyanahhoz az absztrakciós szinthez kell tartozniuk, mint amelyhez a fogalom is tartozik.

Ezzel a feltételezéssel is kivétel nélkül minden fogalmi módszertan él. Általános a meggyőződés, hogy legfeljebb három szint elegendő bármilyen modellezéshez. És csakugyan, néhány módszertan - pl. a TAXIS [MY'80b] - három, míg a többi csak két szintet alkalmaz.

Az "is-a" kapcsolatbeli absztrakt objektumok egyediségének feltételezése:

$$\left.\begin{array}{l} A \text{ is-a } B \\ \underline{\text{és}} \\ A \text{ is-a } C \end{array}\right\} \Longrightarrow \quad B = C$$

Néhány módszertan nem él ezzel a megszorítással. Mi nem sugalljuk sem elfogadását, sem elutasítását. Ez csupán

egy lehetőség. A feltételezés visszautasításának azonban természetesen ára van.

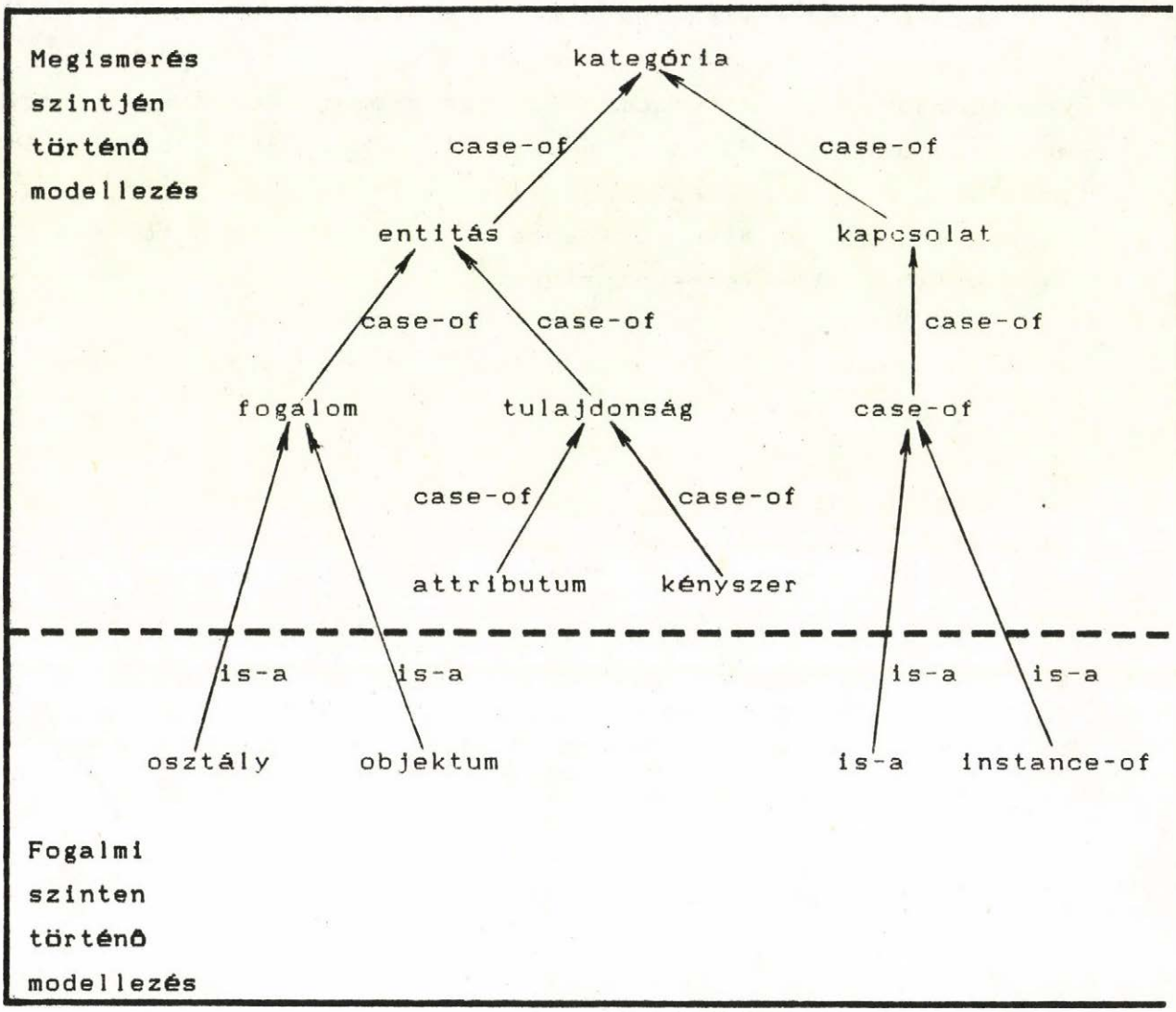
Korlátozzuk az absztrakciós szintek számát kettőre, ekkor megkülönböztethetjük az "absztrakt" fogalmak (osztályok) szintjét és a "konkrét" objektumok szintjét. Ennélfogva az objektumok az osztályok példányai. Emellett még a következő kiegészítő feltételezéssel élünk:

Az "is-a" reláció" csak az osztályok szintjén alkalmazható.

Az ismert módszertanok lényegileg ezt a szétválasztást alkalmazzák. A néhány modellben bevezetett un. "meta-szint" csupán "másodlagos" szerepet játszik. Érdemes megemlíteni, hogy a SIMULA 67 mindhárom általunk említett feltételezéssel él, sőt az utóbbi két korlátozást is alkalmazza.

Ezen a ponton a modellező fogalmak halmaza a következőképpen alakul:

absztrakt szint



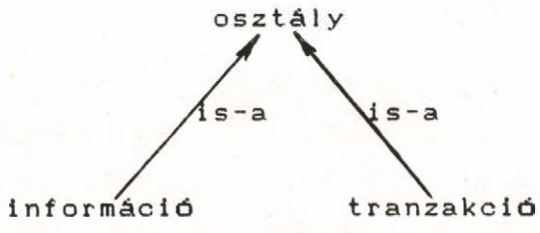
KONKRÉT PÉLDÁNYOK

konkrét szint

A modellezendő valós világi jelenség

2.4 "Logikai" szintű modellezés

Fogalom-renszerünket tovább finomíthatjuk bármelyik ismert, részletesebb modell irányába. Információs rendszerek esetén pl. nagyon fontos, hogy megkülönböztethessük az "információ-strukturákat" és a "tranzakciókat", és mindkettőt ajánlott eszközökkel támogathassuk. Ennélfogva az "osztály" fogalmát a következőképpen finomíthatjuk:



2.4.1 Információ-strukturák és tranzakciók

Az információ-strukturák logikai szintű leírására sajátos attribútumok szolgálnak. A TAXIS pl. "kulcsokat", "konstansokat", "változókat" vezet be (a neveket itt megváltoztattuk). Az "információ" fogalmát tehát így definiálhatjuk:

```
notion információ;
  is-a osztály;
  attributes
    kulcs: információ,
    konstans: attribútum,
    változó: attribútum;
```

Megjegyzések:

* A további megszorítás nélküli attribútumok általában több értékűek. Ez a fenti definicióra is vonatkozik, azaz egy "információ-struktúra" akárhány kulccsal, konstanssal, változóval rendelkezhet.

* A "kulcs" attribútum a fenti definició szerint tetszőlegesen összetett információ-struktúra lehet.

* Felvetődhet, hogy a "kulcs", "konstans" és "változó" attribútumokat az "attribútum" fogalom speciális (is-a) változataiként ábrázoljuk. Természetesebb azonban, ha nem ezt az utat választjuk.

Hasonlóképpen definiálhatjuk a TAXIS szellemében a tranzakció fogalmát is:

```
notion tranzakció;
  is-a osztály;
  attributes
    paraméter: információ,
    lokális: attribútum,
    funkció: tevékenység,
    válasz: információ;
  constraints
    előfeltétel: predikátum,
    eredmény: predikátum;
```

A fenti definicióban két definiálatlan fogalom is szerepel, éspedig a "tevékenység" és a "predikátum". Az algoritmikus és logikai kifejezések valójában nem tartoznak szervesen a modellezés logikai vagy fogalmi szintjéhez. E tervezési fázisban a modell építőjének és felhasználójának kényelme szempontjából jobb lenne a "tevékenységet" és a "predikátumot" inkább szavakban kifejezni, esetleg néhány jól kiválasztott specifikációs formalizmussal támogatva, de semmiképpen sem teljesen formálisan, azaz a jelenleg elérhető formális matematikai specifikációs eszközökkel.

Ezen a modellezési szinten a valóban szükséges formalizmusok helyes kiválasztása további kutatások feladata lehet.

2.4.2 Attributum-tulajdonságok

Információs rendszerek tervezésekor kényelmes, ha számos definiált attributum-tulajdonság áll a modellező rendelkezésére. Különösen hasznos a [KI'84]-ben javasolt választék, amely tartalmaz

- többértékű,
- egyértékű,
- egyedi,
- kimerítő,
- nem nulla,
- fordított stb.

attributumokat. Ezek az "attributum" résztipusaiként modellezhetők. Részletekbe itt nem bocsátkozunk.

2.4.3 Attributum-módok

Ahogy már korábban is említettük, az egyes attributumokhoz "mód" is társítható, pl. "set-of", "list-of", stb. Ezek a másodlagos absztrakciós mechanizmusok, nevezetesen az "aggregáció" és az "asszociáció" (ld. pl. [BRO'84], [GI'85]) kifejezésére szolgálhatnak.

Felmerülhet a kívánság, hogy az "aggregációt" és "asszociációt" önálló eszközökként alkalmazhassuk új fogalmak létrehozására, olyan fogalom-definiáló konstrukciós szerkezeteket használva, mint pl. "is-a set of":

```
notion gyermek;
  is-a set-of személy;
```

A modellezés tisztaságát és következetességét szem előtt tartva azonban más utat választunk. A fenti megközelítés helyett azt mondjuk, hogy a "tagokkal bírás" valójában tulajdonság, és pl. a következőképpen kell ábrázolni:

```
notion gyermek;
  attributes
    tagjai: set-of személy,
    ...
```

Számos más mód is javasolható, pl. "array-of", "tuple" stb. Ezt pillanatnyilag nem rögzítjük. Érdekes kérdés azonban, hogy a modellező személy vajon definiálhatja-e saját konstrukciós szerkezeteit mint új attributum-módokat? A választ csak további kutatások eredményei adhatják meg.

Néhány modell külön mechanizmusokat vezet be az objektum példányok halmazaihoz társított tulajdonságok kezelésére. (Ez az egyik oka pl. a TAXIS-beli metaosztályok bevezetésének is.) Pusztán egy objektum-halmaz azonban nem fogalmi szintű szerkezet. Ha mégis az, akkor viszont önálló fogalomnak kell lennie, melynek van neve, és amelyet a többi fogalomhoz hasonlóan definiálhatunk, mint pl.

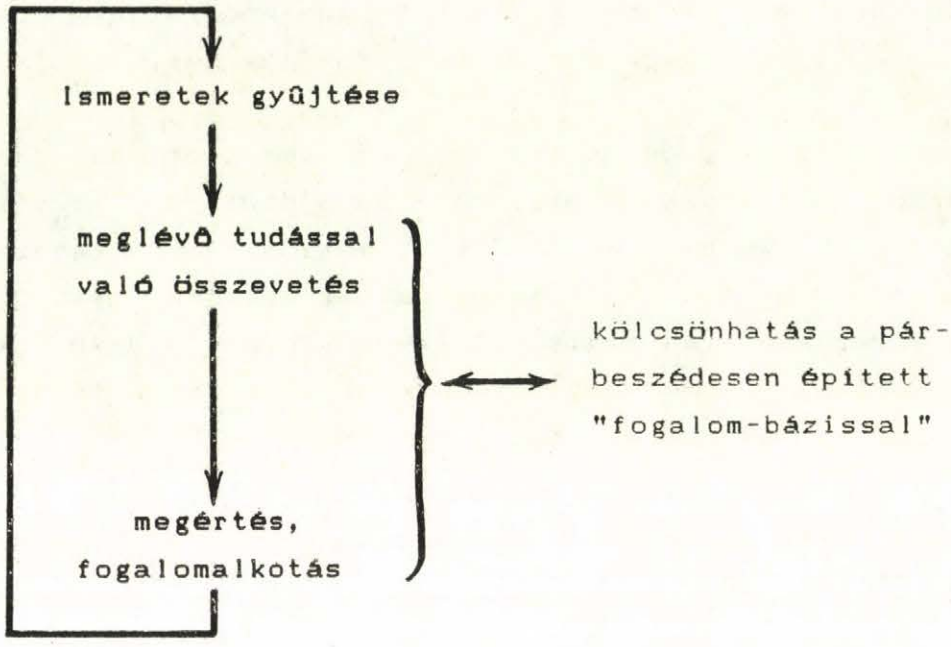
```
notion gyermek;
  attributes
    tagjai: set-of személy,
    átlagosan_kórházban_töltött_napok_száma: szám,
    ...
  constraints
    korhatár: 14.életévét nem töltötte be,
    ...
```

2.5 A modell-átalakítások kérdése

A számítógéppel támogatott fogalmi modellezés gyakorlati használatának kulcskérdése a dinamika.

4.5.1 Egy meglévő modell megváltoztatása

Az ismeretgyűjtés közismerten iteratív jellegű folyamat:



Az újonnan szerzett ismeret gyakorta a tudásbázis radikális változtatását igényli. A fogalmakat rendszerint ugyanis akkor találjuk tévesnek, amikor már sok különféle példányuk is van. Ez természetes is, hiszen először a konkrét objektumokra vonatkozó ismeretekre teszünk szert, és csak ezek után absztrahálhatunk.

Ugyanakkor az ábrázolás tradicionális megközelítése éppen a valóság ellentétének tűnik. Hiszen a jelenlegi modellek csaknem mindegyike (pl. [MY'84], [ALB'85], [TEI'79] [KN'82], [SH'81] stb.) megköveteli, hogy

előbb a fogalmakat definiálják a modellezők, és a konkrét példányok megadásakor már létező definíciókra hivatkozzanak; mi több

az egész fogalmi univerzumot rögzítsék, mielőtt még a konkrét példányok megadására sor kerülne.

Könnyen érthető, hogy ez utóbbi követelmény mögött az adathalmaz integritásának megőrzése áll.

Azt kellene tehát lehetővé tennünk, hogy a számítógéppel támogatott eszközök gondolkodásmódunkhoz illeszkedjenek, és a fogalmakat összes integritási tulajdonságuk megőrzésével manipulálni lehessen, annak dacára, hogy akárhány konkrét példánnyal rendelkeznek.

2.5.2 A modellezési szintek közti leképezés

Sokrétegű modellezési technikát használva minden szinten önálló modellünk van, melyeknek az adott szintre jellemző alkalmas kritérium értelmében "teljeseknek" kell lenniük.

Ennélfogva gyakran kell ugyanazokat a fogalmakat különböző szinteken, tehát eltérő absztrakciós mértékben kezelnünk.

Ezért további feltárandó terület lenne olyan alkalmas leképező mechanizmus meghatározása, amely úgy kapcsolná össze a szinteket, hogy egyúttal garantálná az egymástól függő fogalmak közti összeegyeztethetőséget is.

2.6 Fogalmi-szintű transzformációk

A fogalom-strukturákon manipuláló operációk három kategóriáját különböztethetjük meg, éspedig:

Szabványos, naprakész állapotra hozó operációk
Ezek a hagyományos információ-karbantartó operációk "insert", "delete", "modify" stb. megfelelői.

Hagyományos absztrakciós mechanizmusok
Idesoroljuk az osztályozást, az általánosítást, az aggregációt és az asszociációt.

Fogalmi átalakítások
Már környezetükbe ágyazott, és valahány konkrét példánnyal rendelkező fogalmakra ható operációk.

2.6.1 Szabványos operációk

Ezek a mi szempontunkból kevésbé érdekesek, noha pontos szemantikájuk bizonyos esetekben korántsem triviális. Ezért csak körvonalazunk néhány velük kapcsolatban felmerülő kérdést.

2.6.1.1 "Create" - fogalom létrehozása

Ez az operáció sosem veszélyezteti az adat-integritást. Az egyetlen probléma, amit gyakorlati okokból idéz elô az, hogy gyakran hivatkozhat más, még nem létezô fogalmakra. Ezt a következôképpen célszerû kezelni:

- bármely nem létezô fogalomra történô hivatkozás esetén a párbeszédes megerôsítés elkerülhetetlen;
- amennyiben a nemlétezô fogalomra történô hivatkozás megerôsítést nyer, egy üres, késôbb definiálandó (dummy), un. "nyílt fogalom" jön létre;
- "nyílt fogalmak" konkrét példányainak megadása nem megengedett, de "nyílt fogalmak" specializálhatók, és az így bevezetett speciális fogalmak szintén "nyílt fogalmak" lesznek.

2.6.1.2 "Cancel" - fogalom érvénytelenítése

A törlés a hivatkozást megengedô adat-univerzumok minden változatában kényes probléma (pl. [DE'86]). A fogalmakra ez még inkább igaz.

A fogalmakra rendszerint úgy hivatkozunk, mint

- objektum-osztályozásokra,
- általános fogalmakra, és
- attributumok értékkészletére.

Valahányszor törölni szándékozunk egy fogalmat, az összes fenti viszonyt úgy kell kezelnünk, hogy minden integritási tulajdonságot megőrizzünk. Ezt párbeszédes mechanizmusok támogatják. Amennyiben az integritási tulajdonságok megőrzése nem biztosított, a törlést megtagadjuk.

A fogalmak érvénytelenítésének számos változata van attól függően, hogy az érvénytelenítés mennyire érinti a tudásbázist. A következő fő variánsokhoz társítható természetes, gyakorlati jelentés:

(i) egyszerű törlés: töröld a fogalmat, ha nem sért integitási szabályt.

(ii) erőltetett törlés: töröld a fogalmat és az összes értelmét vesztő, közvetlenül e fogalomra hivatkozó objektumot, ha az integritás így biztosított.

(iii) tranzitív törlés: ugyanaz, mint az erőltetett törlés, csak éppen az objektumok tranzitív módon vett maximális halmazára vonatkoztatva.

2.6.1.3 "Change" - attribútum értékkészletének megváltoztatása

Az operáció a fogalom-specifikáción belül változtatja meg egy attribútum értékkészletét. Ez az átalakítás csak az eredeti értékkészletnél általánosabb vagy speciálisabb értékkészlet megjelölése esetén fogadható el.

(i) Ha az új értékkészlet az eredetinél általánosabb, az összes integritási tulajdonság érintetlen marad. Ezt az általánosított típus egybevágási törvény [KN'85] garantálja, azaz az aktuális példányok attribútum-értékei mindig összeegyeztethetők egy általánosított specifikációval.

(ii) Ha az új értékkészlet az eredetinél speciálisabb, az integritás nem garantált. Ekkor:

- valahányszor szükséges, a hivatkozott, hibás vagy téves speciális attribútumú objektumok újraminősítését kéri a rendszer;

- az értékkészlet-változtatás csak akkor megengedett, ha nem sért integritási szabályt.

2.6.1.4 "Modify" - fogalom módosítása

A fogalmat újabb tulajdonságokkal egészíti ki, vagy már meglévő tulajdonságokat távolít el. Ez tipikus relációs adatbázisbeli művelet. Fogalmi strukturák esetén azonban nem engedjük meg. Ennek az az oka, hogy ugyanaz a hatás más átalakítások sorozatával is elérhető, és így az integritási tulajdonságok teljes halmaza sokkal biztonságosabban és természetesebben kezelhető.

2.6.2 Absztrakciós mechanizmusok

Az absztrakciós mechanizmusokat már kielégítően tárgyalták az irodalomban. Az új az, hogy párbeszédes, interaktív módon nyújtjuk most ezeket a felhasználóknak. Így használatuk nem korlátozódik a "definiciós szakaszra", és a tudásbázis egész életciklusa alatt elérhetők maradnak. Nézzük meg, mit jelent ez a gyakorlatban.

2.6.2.1 Osztályozás (classification)

A következő szabályokat és mechanizmusokat alkalmazzuk:

(i) a rendszerben nem lehet osztályhoz, fogalomhoz nem társított objektum;

(ii) a rendszerbe bármikor bevihető objektum anélkül, hogy egy létező osztályra hivatkoznánk.

Ezt a látszólagos ellentmondást az oldja fel, hogy automatikusan új fogalom jön létre, valahányszor osztályozatlan objektummal kerül szembe a rendszer. Az új fogalomra, mint prototípussal való definícióra hivatkozunk. Pl.

object vádemelés prototype-of esemény;
  attributes
    előző (esemény): nyomozás,
    rákövetkező (esemény): tárgyalás;

Ily módon prototípusból való "tanulással" ellenőrizhetjük, hogy ugyanazon osztály bármely további objektumának attribútumai egybeesnek-e a prototípusból levezetett specifikációval. Ha nem, akkor természetesen az is előfordulhat, hogy a prototípus specifikációt adtuk meg rosszul. Ekkor azt a módosító operációk megfelelő halmazának alkalmazásával a kívánt módon megváltoztathatjuk.

A fordított irányú operáció, a konkrét példányok megadása (instantiation) semmi újat sem jelent ebben a környezetben, ezért nem foglalkozunk vele.

2.6.2.2 Általánosítás (generalization)

E mechanizmus párbeszédes változata lehetővé teszi, hogy néhány fogalom közös tulajdonságait kiválasztva egy új általános fogalmat hozzunk létre. Ily módon az eredeti fogalmak ugyan elveszítik a kiválasztott tulajdonságokat, de ezután öröklik azokat az új, általánosabb fogalomtól.

Nevezzük ezt az operációt "abstraction"-nek, elvonatkoztatásnak.

Az "abstraction" operáció, azaz a közös tulajdonságok kiválasztása nem érinti az objektum példányokat, és ennélfogva a tudásbázis integritását sem sérti meg soha. Hiszen a kiválasztott tulajdonság-definiciók csak az osztályozó fogalomból tűnnek el, de öröklődéssel mind definiáltak maradnak.

A közös tulajdonságok kiválasztásának megkisérlése előtt meg kell adnunk a tulajdonságok közti leképezést, hogy ennek alapján azonosíthassuk azokat.

Tekintsük pl. a következő fogalmakat:

```
notion mintaillesztő;
  attributes
    minta: reguláris kifejezés,
    teendő: tevékenység,
    input: list-of adatállomány-név;
```

```
notion szövegszerkesztő;
  attributes
    parancsok: szerkesztő_parancsok,
    szerkesztendő: list-of adatállomány-név;

notion rendező;
  attributes
    feltétel: rendezési_feltétel,
    adat: list-of adatállomány-név,
    rendezett: adatállomány;
```

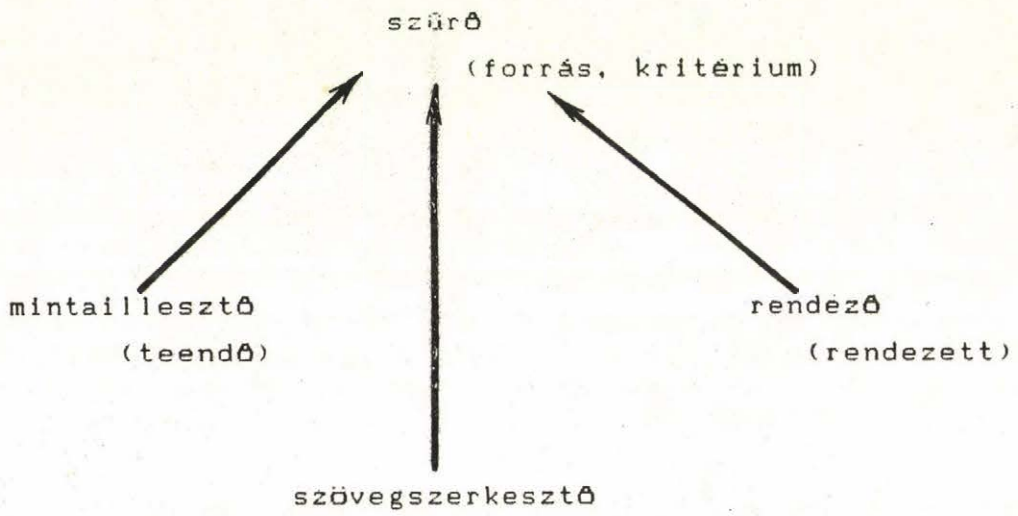
Könnyen felismerhető, hogy e fogalmak mindegyike valamiféle "információ-szűrő" (ld. UNIX [BOU'83]). Hogy e felismerésnek megfelelően bevezethessük a "szűrő" fogalmát, először az attributumokat kell megfeleltetnünk egymásnak, pl.

```
forrás := input in mintaillesztő,
          szerkesztendő in szövegszerkesztő,
          adat in rendező;

kritérium := minta in mintaillesztő,
             parancsok in szövegszerkesztő,
             feltétel in rendező;
```

E leképezések birtokában már előírhatunk egy olyan parancsot, hogy

notion szűrő abstraction-of mintaillesztő,
szövegszerkesztő,
rendező;
attributes extracted forrás,
kritérium;

Az átalakítás után a következő hierarchiát nyerjük:



Az "abstraction" operáció tehát nem sérti az integritást, viszont más problematikus kérdést vet fel, éspedig a következőt. Az "abstraction" operációban résztvevő fogalmak már rendelkezhetnek feléjük rendelt általános, un. szuperfogalommal. Így az "abstraction" öröklési hálózathoz vezethet, az általánosan elfogadott hierarchiákkal szemben. Egy konk-

rét megvalósításnak el kell döntenie, hogy megenged-e ilyen általánosított struktúrákat vagy sem.

A fordított operáció, a "specializáció" fogalmi szinten úgy valósítható meg, hogy a hierarchia megfelelő pozíciójában létrehozunk egy új fogalmat (ld. szabványos operációk).

Az "abstraction" operáció szélsőséges változata, amikor a kiválasztott tulajdonságok száma nulla, szintén modellezhető egy egyszerű "create" operációval.

2.6.2.3 Másodlagos absztrakciós mechanizmusok

Két, széles körben elterjedt absztrakciós mechanizmust kell még megemlítenünk, az "aggregációt" és a "csoportosítást". Ezeket "másodlagos" absztrakciós mechanizmusoknak nevezzük, mert

- nem hoznak létre új osztályokat (absztrakt szintű fogalmakat);
- ehelyett (konkrét) objektumokból álló szerkezeteket építenek; és
- e szerkezetekbe tartozás dinamikus.

2.6.2.4 Aggregáció (aggregation)

Az aggregáció heterogén objektumok időleges konglomerációja, melyhez tetszés szerint név is társítható, pl.:

```
rakomány = (vegyszer,
            UNIX PC,
            rózsaolaj);
```

Az aggregációnak időlegességéből adódóan nincs rögzített szerkezete, különben prototípus fogalommal modelleznénk. Az összetevőket a belső reprezentációban a "part-of" kapcsolat köti össze.

Meg kell jegyeznünk azonban, hogy az aggregáció kifejezés használata korántsem egységes az irodalomban. Néhány szerző a fent említettől eltérően használja.

2.6.2.5 Csoportosítás (association)

A csoportosítás homogén objektumok időleges halmaza, melyet egy

(fogalom, kritérium)

pár jelöl ki, és a fogalom azon objektum-példányainak halmazát jelenti, melyek kielégítik a kritériumot. Ezeket az objektumokat a csoportosítás "elemeinek" (member-of) tekintjük. Ezen a ponton az irodalomban használatos terminológia ugyancsak nem egységes.

Végül megjegyezzük, hogy a másodlagos absztrakciós mechanizmusokat fogalmi szinten, azaz "nem időleges" értelemben is szükséges ábrázolni. Erre az un. "összetett attributumok" a legalkalmasabbak, pl. "set-of", "list-of" stb. - ezt itt nem részletezzük.

2.6.3 Az absztrakció néhány új mechanizmusa

Hozzáférhető fogalmi nyelvekben ilyen operációkat még nem valósítottak meg.

2.6.3.1 "Unify" - fogalmak azonosságának felülvizsgálata

Egy fogalom-halmazt használva gyakran kiderül, hogy néhány olyan fogalom, melyet azelőtt különböző fogalmakként kezeltünk, valójában fogalmilag megegyezik, és ennélfogva jobb lenne, ha egységesen modelleznénk ezeket. Rendelkezhetünk pl. a következő fogalom-párral:

```
notion üzenet;
  attributes
    címzett: ...,
    feladó: ...,
    dátum: ...,
    szöveg: ...;
```

```
notion: emlékeztető;
  attributes
    kinek: ...,
    kitől: ...,
    mikor: ...,
    miről: ...;
```

amelyeknek jelentése fogalmi szempontból ugyanaz. Jelölésük különbsége abból eredhet, hogy az első fogalom tipikusan egy rendszer tervezéséhez tartozik, míg a másodikat a környezet-elemző fogalmazhatja meg.

Érdekes matematikai példaként megemlíthetnénk az univerzális algebra úgynevezett "klónjait", és a többértékű logikák projekciókat tartalmazó zárt osztályait [KN'87]. Fogalmilag mindkettő ugyanazt ábrázolja, noha jelöléseik és terminológiáik eltérnek. (A zárt osztály egy függvényhalmaznak szuperpozicióra vonatkozó lezárása, míg a klón egy algebra term-függvényeinek halmaza.)

Ilyen esetekben lényeges, hogy a tulajdonság-halmazok, azaz az attributumok, és a rajtuk értelmezett kényszerek (amennyiben ilyenek egyáltalán léteznek) között kölcsönösen egyértelmű megfeleltetést találjunk. Ha ez megtörtént, akkor technikai szempontból két lehetőségünk van:

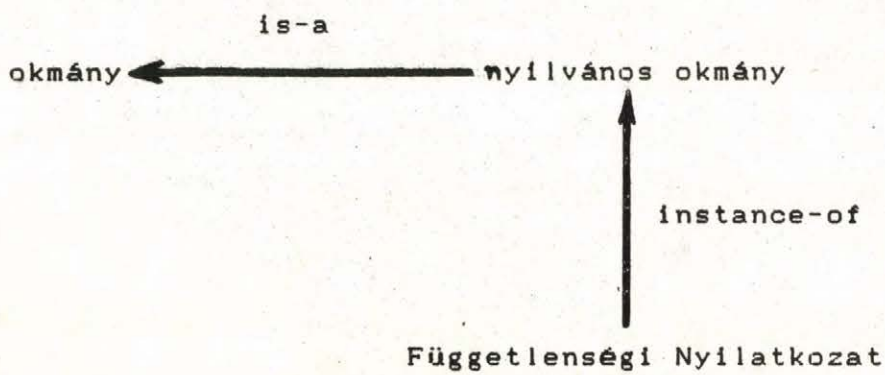
(i) Link, vagyis a fogalmak összekötése.
Ilyenkor egyszerűen, az eredeti fogalmak megváltoztatása nélkül, megadjuk az izomorf leképezést. Ennek következtében objektum példányaik halmaza megkülönböztethető marad, de uniójuk is visszakereshető.

(ii) Unite, azaz a fogalmak egyesítése egy új fogalomban (amely akár a komponensek egyike is lehet).
Ebben az esetben a meglévő objektum példányok

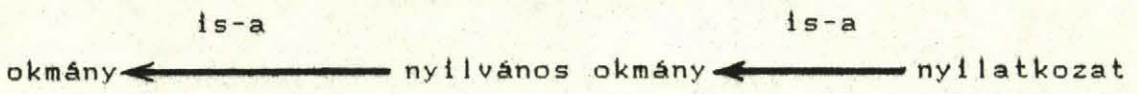
halmazai egyesülnek. Az unió-képzés további kérdést vet fel, nevezetesen az objektum azonosságának felülvizsgálatát. Két - azonos tipusú - objektumot akkor és csak akkor tekintünk azonosnak, ha megfelelő attributum- és kényszer-párjaik ugyanazokra az objektumokra és kényszerekre hivatkoznak.

2.6.3.2 "Requalify" - objektumok átminősítése

Ezen operáció célja, hogy megváltoztassa azt az osztályt, amelyhez egy konkrét példány tartozik. Ésszerű korlátozás, hogy az új és az eredeti osztály között is-a* kapcsolatnak kell lennie, azaz egy objektum csak általánosabb vagy speciálisabb osztály példányává minősíthető át. Tegyük fel pl., hogy tudásbázisunkban a következő helyzet alakult ki:



Később tapasztalatunk alapján meggyőződhetünk arról, hogy a leírt sajátos okmánynak vannak olyan általánosítható tulajdonságai, melyeket érdemes fogalmi szintre emelni:



Ekkor a "Függetlenségi Nyilatkozatot" kívánatos "nyilatkozatnak", és nem csupán "nyilvános okmánynak" minősíteni.

Általában az objektumok mindkét irányú átminősítése megengedett. Az általános irányba történő átminősítés, fenti példánknál maradva pl. egy "nyilatkozat" "nyilvános okmánnyá" minősítése, mindig megtehető, hiszen nem igényel új információt.

Ha azonban egy objektumot az eddigieknél speciálisabb fogalom példányává minősítjük át, akkor

(i) meg kell adnunk a specifikus fogalomra jellemző extra attributumok értékeit, hiszen ezekkel az eredeti osztály még nem bírt;

(ii) ellenőrizni kell a specifikus fogalomhoz társított további kényszereket.

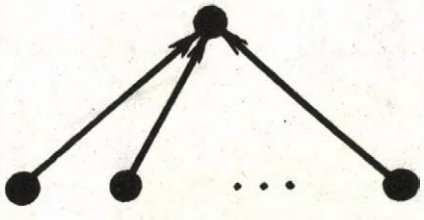
Ennek megfelelően a "requalify" operáció kéri az összes további hiányzó attributum specifikációját, és az átminősítés addig nem lép érvénybe, míg ezek mind nem adottak, és az összes különleges speciális igény nem teljesül.

2.6.3.3 "Refine" - az általánosító fogalom lánc finomítása

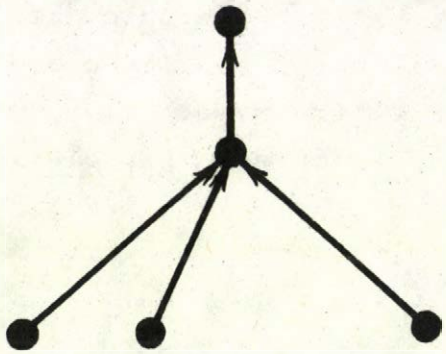
Ez az operáció új fogalmat szúr egy már létező általánosító/specializálódó láncba.

Két meglévő fogalom között létrehozott új fogalom a korábbi attributumok áthelyezését teszi szükségessé. Elvből csak az attributumok általános irányba történő mozgatását engedjük meg. Ennek megfelelően példánkban (következő oldal) a "középpontos nagyítás" bizonyos attributumai áthelyezhetők a "hasonlósági-transzformációhoz", de a "geometriai transzformáció" fogalma ezen a módon nem változtatható meg.

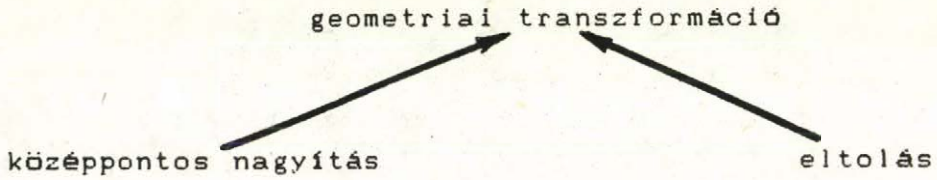
Szembetűnő az "abstraction" operáció (ld. 2.6.2.2) és a most bevezetett "refine" operáció közti szoros kapcsolat. A "refine" operáció általában egy



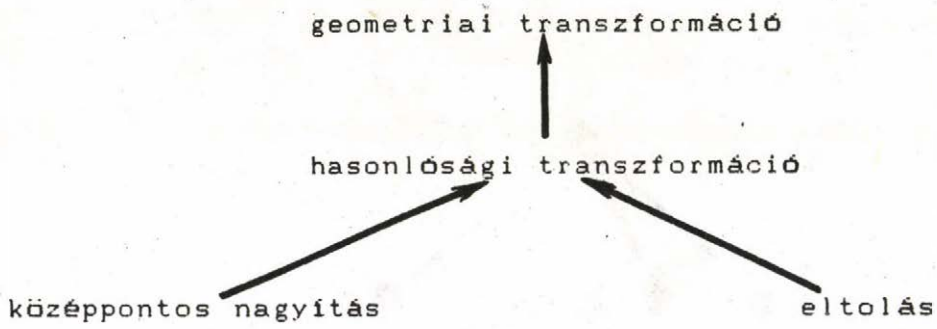
alakú strukturát alakít át



alakúvá. Pl. a



struktúra esetén végül is a



alakzatot szeretnénk megkapni.

A "refine" operáció tehát ugyanúgy működik, mint az "abstraction", azzal a kivétellel, hogy nem hozhat létre többszörös szülőket, hanem ehelyett a meglévő is-a kapcsolatokat változtatja meg. Következésképp a "refine" átalakítás tulajdonságai a konkrét példányokat illetően megegyeznek az "abstraction" operáció megfelelő tulajdonságaival.

2.6.3.4 "Contract" - az általánosító fogalom-lánc tömörítése

Ez az operáció két, is-a kapcsolattal összefüggésbe hozott fogalmat von össze egy egyszerű fogalommá. Tegyük fel pl., hogy definiáltuk a következő két fogalmat:

```
notion transzformáció;
  attributes
    név: ...,
    paraméterek: ...,
    tartomány: ...,
    változatok: ...;

notion t-transzformáció; is-a transzformáció;
  attributes
    időmérés: ...;
  constraints
    szinkronizálás: ...;
```

Nos, ha nem akarunk élni a fenti megkülönböztetéssel, kérhetjük hogy

contract (transzformáció, t-transzformáció)
as transzformáció;

ezzel egy olyan egyszerű fogalmat hozva létre, amely mind a hat fenti tulajdonsággal rendelkezik.

Ez az operáció csak akkor megengedett, amikor az általános fogalomnak egyáltalán nincs konkrét példánya.

3. Fejezet

A TARGYALT MODELLEK SZEMANTIKAI ALAPJAI

3.1 Bevezetés

Számos ok késztet minket arra, hogy megadjuk az előző fejezetben tárgyalt modellek jelentésének formálisabb leírását.

Először is a formalizálás folyamata a kigondolt dolgok pontos jelentésének részletes végiggondolására kényszerít, és gyakran vezet kétértelműségek és aszimmetriák felfedezéséhez.

Másodszor a formális szemantika olyan szabványt tűz ki, mellyel szemben megitélhető bármely számítógépes eszköz helyessége, amely egy modell ellentmondásmentességét kívánja meghatározni, vagy pl. arra akar válaszolni, hogy előadódhat-e egy bizonyos sajátos helyzet a világban az aktuális specifikációk alapján.

Végül a felhasználók végső döntőbíróként fordulhatnak a formális specifikációhoz azokban az esetekben, melyekben valamely specifikáció pontos jelentését illetően nézeteltérés merül fel.

A szemantika kifejezésére egy "many sorted" elsőrendű nyelvet használunk, melyben:

a "sortok" S halmaza nem üres, és a hagyományoktól eltérően nem tételezzük fel, hogy a "sortok" kölcsönösen kizárják egymást;

a változó-szimbólumok gyűjteménye potenciálisan végtelen, és mindegyikük a "sortok" valamelyikéhez tartozik;

minden egyes n-re adott olyan n-változós predikátum-szimbólumok gyűjteménye, amelyek mindegyikéhez az S-beli elemek egy, vagy több n-dimenziós vektora, un. "sort-szignatura" társul. Intuitív módon a szignaturák írják elő a megfelelő predikátumok argumentumainak "sortjait";

minden egyes n-re adott olyan n-változós függvény-szimbólumok gyűjteménye, melyek mindegyikéhez az S-beli elemek egy, vagy több (n+1)-dimenziós vektora, un. "sort-szignatura" társul. A szignaturák a függvény-argumentumok, valamint a függvényérték "sortjait" írják elő;

a

$$\vee \quad \wedge \quad \neg \quad \Rightarrow \quad \Leftrightarrow \quad \forall \quad \exists$$

jeleket a szokásos értelemben használjuk;

az egyenlőség fogalmának bevezetéséhez a nyelvet kiterjesztjük az = bináris predikátummal, és ¬= komplemensével.

Hagyományosan feltételezik, hogy a "sortok" kizáróak, azaz egy objektum csak egy "sorthoz" tartozik, és a predikátumok illetve a függvények csak egy sort-szignaturával rendelkeznek.

Ez a módosított definició tekintélyesen egyszerűsíti feladatunkat, és az [EN'72]-beli szerkezet, amely megmutatja, hogy a közönséges many sorted logika ekvivalens a szabványos logikával, triviálisan adaptálható nem diszjunkt sortok esetére.

Az állítások olyan elsőrendű formulák, melyek állításra nem hivatkoznak. Használhatók bennük az aritmetika és a karaktersorozatok kalkulusának szokásos jelölései, és szerepelhet bennük a "NIL" és "THIS" speciális szimbólum. A "NIL" speciális konstans minden sortnak eleme. Jelentése: "nincs érték". A "THIS" szimbólum a definiálandó osztály példányaira/"eseteire" utal, és olyan változónak tekintjük, melynek értékkészlete a definiálandó osztály.

Az állításokban feltételezzük, hogy az összes, "THIS"-től különböző szabad változóra univerzális kvantor vonatkozik, és valahányszor egy szimbólum konstans is, és változó is lehetne, változó - hacsak kifejezetten nem állítjuk, hogy konstans.

3.2 A megismerés szintjén szerkesztett modell jelentése

Ezen a szinten három sort-ot vezetünk be:

$$S = (C, P, A)$$

melyekbe rendre a fogalmakat, a tulajdonságokat és az állításokat soroljuk. Minden ξ formulához egy A^{ξ} állítás-fogalom tartozik, amely az állítás sort eleme. A^{ξ} tulajdonságai a ξ-ben szereplő szabad változók.

A különféle modellbeli szerkezetek jelentésének megmagyarázására a következő két alapvető függvényt illetve predikátumot használjuk:

PROPDEF kétváltozós tulajdonság-függvény, amely megadja egy "F" fogalom vagy egy ξ állítás "t" tulajdonságának értékkészletét, így szignaturái (C, P, C) és (C, P, A), valamint (A, P, C);

CASE kétváltozós predikátum, amely azt állítja, hogy az első paraméterként megadott "F_1" fogalom vagy ξ_1 állítás a második paraméterként megadott "F_2" fogalom vagy ξ_2 állítás "esete-e", azaz hogy F_1 <u>case-of</u> F_2 vagy A^{ξ_1} <u>case-of</u> A^{ξ_2} igaz-e. Szignaturái tehát (C, C) és (A, A). Nyilvánvaló, hogy

$$CASE(A^{\xi_1}, A^{\xi_2}) \Longrightarrow (\xi_1 \Longrightarrow \xi_2)$$

3.2.1 Alap-axiómák

Ezek az axiómák írják le, hogy milyen összefüggés van két fogalom tulajdonságai között, ha az egyik a másik "esete". Azt mondjuk, hogy egy fogalom valamely tulajdonsága létezik, ha e tulajdonság értéke nem "NIL".

* NIL-érték axiómák: {az alapvető predikátum értéke NIL-re hamis, az alapvető függvény értéke pedig NIL esetén NIL}

$$\neg CASE(NIL, x_C) \wedge \neg CASE(x_C, NIL) \wedge \neg CASE(NIL, x_A) \wedge \neg CASE(x_A, NIL)$$

$$PROPDEF(NIL, t_P) = NIL \wedge PROPDEF(x_C, NIL) = NIL$$

* Intenzionális <u>case-of</u> kényszer: {az öröklött attribútumok tartománya speciálisabb, az öröklött kényszerek erősebbek}

$$PROPDEF(x_C, t_P) = y_C \wedge y_C \neg= NIL \wedge CASE(v_C, x_C) \Longrightarrow (\exists z_C) PROPDEF(v_C, t_P) = z_C \wedge z_C \neg= NIL \wedge CASE(z_C, y_C)$$

$$PROPDEF(x_C, t_P) = y_A \wedge y_A \neg= NIL \wedge CASE(v_C, x_C) \Longrightarrow (\exists z_A) PROPDEF(v_C, t_P) = z_A \wedge z_A \neg= NIL \wedge CASE(z_A, y_A)$$

$$PROPDEF(x_A, t_P) = y_C \wedge y_C \neg= NIL \wedge CASE(v_A, x_A) \Longrightarrow (\exists z_C) PROPDEF(v_A, t_P) = z_C \wedge z_C \neg= NIL \wedge CASE(z_C, y_C)$$

3.2.2 Fogalom-definiciók axiómái

Egy fogalom-definició új szimbólumokat, azaz predikátumokat, függvényeket és konstansokat, valamint számos axiómát vezet be.

Egy tetszőleges fogalom-definició általános szintaxisa:

notion F; [case-of G;]

[attributes a_1 : D_1 [, a_k : D_k]$_2^n$;]

[constraints c_1 : ξ_1 [, c_j : ξ_j]$_2^m$;]

amelyben legalább egy tulajdonságot (attributumot vagy kényszert) meg kell adnunk.

E definició eredményeként a fogalom-konstansok listájához kell adnunk "F"-et, a tulajdonság-konstansok listájához pedig azon "a_k"-kat és "c_j"-ket, melyeket még nem tartalmaz.

Az attributum-definiciókban szereplő D_k "tartományok" fogalmak. A kényszer-definiciók , állításaira egy-egy új állítás-konstanst vezetünk be.

Meglévő axiómáinkat pedig kibővítjük a következő axiómákkal:

CASE(F,G) (amennyiben a definicióban szerepel case-of klauzula)

$PROPDEF(F,a_k) = D_k \qquad k=1,\ldots,n$

$PROPDEF(F,c_j) = A^{\xi_j} \qquad j=1,\ldots,m^*$

és e változókat nem tartalmazó axiómákon kívül még előírjuk, hogy

$$\xi_j \iff (\exists y)\, CASE(y, A^{\xi_j}) \qquad j=1,\ldots,m$$

$$(\forall x)CASE(x,F) \iff ((\exists y)CASE(y,A^{\xi_j}) \Longrightarrow PROPDEF(x,c_j) = y)$$

illetve, ha "ξ_j" szabad változóként tartalmazza a "THIS" speciális változót, akkor A^{ξ_j} -t egyetlen "v:ANY" tulajdonsággal kell definiálni, és a fenti utolsó axióma a következőképpen változik:

$$(\forall w)\, \xi_j(w/THIS) \iff ((\exists x)CASE(x,A^{\xi_j}) \wedge PROPDEF(x,v) = w)$$

$$(\forall x)CASE(x,F) \Longrightarrow ([(\exists y)CASE(y,A^{\xi_j}) \wedge PROPDEF(y,v) = x] \Longrightarrow PROPDEF(x,c_j) = y)$$

---

* A továbbiakban mindenütt elhagyjuk a sort-okra vonatkozó információt, ha az a szövegkörnyezetből könnyen kikövetkeztethető.

"ANY" tartománydefiníciót - mint neve is sugallja - akkor adunk meg, ha az attribútum értékkészletét nem kívánjuk megkötni.

3.3 A fogalmi szintű modell szemantikája

Fogalmi szintű modell jelentésének megadásakor is a már korábban bevezetett három sort-ot alkalmazzuk, de a fogalmak és az állítások sort-ját Object és Class rész-sort-okra osztjuk, így valójában öt alapvető sort van:

$$S = (C_c, C_o, P, A_c, A_o).$$

Egy kényszer értékeként megadott ξ formulát olyan A^{ξ} állítás-osztály "in-line" definíciójaként értelmezzük, amelynek attribútumai szabad változói. Az állítás-osztályok objektumai igaz predikátumokat ábrázolnak, ezért valamely feltétel teljesülésének megköveteléséhez ki kell jelentenünk, hogy annak a tulajdonságnak (kényszernek) az értéke, amely ezt a feltételt valamely objektumhoz kapcsolja, nem lehet NIL. Így a feltétel teljesülését az állítás-objektumok jelenléte helyettesíti.

A különféle modellbeli szerkezetek jelentésének megmagyarázására a következő alapvető predikátumokat és függvényeket vezetjük be:

INST kétváltozós predikátum, amely az állítja, hogy az "O" objektum a "C" osztály konkrét példánya, instanciája. Szignaturája: (C_o, C_c) és (A_o, A_c).

PROPDEF kétváltozós, un. definició szerinti tulajdonság-függvény, amely megadja, hogy a "C" osztálybeli objektumok "t" tulajdonságának mely osztályba kell tartoznia. Szignaturái: (C_c, P, C_c), (C_c, P, A_c) és (A_c, P, C_c), ahol egy állítás-osztály tulajdonságai az állítás szabad változóit tartalmazzák.

PROPVAL kétváltozós, un. konkrét tulajdonság-függvény, amely megadja egy "O" objektum "t" tulajdonságának értékét. Szignaturái: (C_o, P, C_o), (C_o, P, A_o) és (A_o, P, C_o).

IS kétváltozós predikátum, amely azt állítja, hogy a "C_1" osztály a "C_2" osztály részosztálya. Szignaturája: (C_c, C_c) és (A_c, A_c). Ha $IS(A^{\xi_1}, A^{\xi_2})$, akkor ξ_1 logikailag maga után vonja ξ_2-t:

$$IS(A^{\xi_1}, A^{\xi_2}) \Longrightarrow (\xi_1 \Rightarrow \xi_2).$$

3.3.1 Alap-axiómák

A következő axiómák azt írják le, hogy a definició szerinti, és a konkrét tulajdonságok miként függnek össze az osztályok konkrét példányaival és részosztályaival. Azt mondjuk, hogy egy osztály vagy objektum tulajdonsága létezik, ha értéke nem "NIL".

* NIL-érték axiómák: (az alapvető predikátumok értéke NIL-re hamis, az alapvető függvények értéke pedig NIL esetén NIL)

$$\neg INST(NIL, x_{Cc}) \wedge \neg INST(NIL, x_{Ac}) \wedge \neg INST(x_{Co}, NIL) \wedge \neg INST(x_{Ao}, NIL)$$

$$\neg IS(NIL, x_{Cc}) \wedge \neg IS(x_{Cc}, NIL) \wedge \neg IS(NIL, x_{Ac}) \wedge \neg IS(x_{Ac}, NIL)$$

$$PROPDEF(NIL, t_P) = NIL \wedge PROPVAL(NIL, t_P) = NIL$$

$$PROPDEF(x_{Cc}, NIL) = NIL \wedge PROPVAL(x_{Co}, NIL) = NIL$$

* Tulajdonság-érték kényszer: (a tulajdonság-értékeknek a definiált értéktartományhoz kell tartozniuk)

$$INST(O_{Co}, C_{Cc}) \wedge PROPVAL(O_{Co}, t_P) = y_{Co} \wedge y_{Co} \neg = NIL$$

$$\Rightarrow (\exists D_{Cc}) PROPDEF(C_{Cc}, t_P) = D_{Cc} \wedge INST(y_{Co}, D_{Cc})$$

$$INST(O_{Co}, C_{Cc}) \wedge PROPVAL(O_{Co}, t_P) = y_{Ao} \wedge y_{Ao} \neg = NIL$$

$$==> (\exists D_{Ac})PROPDEF(C_{Cc}, t_P) = D_{Ac} \wedge INST(y_{Ao}, D_{Ac})$$

$$INST(O_{Ao}, C_{Ac}) \wedge PROPVAL(O_{Ao}, t_P) = y_{Co} \wedge y_{Co} \neg= NIL$$

$$==> (\exists D_{Cc})PROPDEF(C_{Ac}, t_P) = D_{Cc} \wedge INST(y_{Co}, D_{Cc})$$

* Extenzionális <u>is-a</u> kényszer: (egy részosztály minden példánya egyben az általánosabb osztály példánya is)

$$INST(O_{Co}, C_{Cc}) \wedge IS(C_{Cc}, D_{Cc}) ==> INST(O_{Co}, D_{Cc})$$

$$INST(O_{Ao}, C_{Ac}) \wedge IS(C_{Ac}, D_{Ac}) ==> INST(O_{Ao}, D_{Ac})$$

* Intenzionális <u>is-a</u> kényszer: (az öröklött tulajdonságok tartománya speciálisabb)

$$PROPDEF(C_{Cc}, t_P) = E_{Cc} \wedge E_{Cc} \neg= NIL \wedge IS(D_{Cc}, C_{Cc})$$

$$==> (\exists F_{Cc})PROPDEF(D_{Cc}, t_P) = F_{Cc} \wedge (F_{Cc} = E_{Cc} \vee IS(F_{Cc}, E_{Cc}))$$

$$PROPDEF(C_{Cc}, t_P) = E_{Ac} \wedge E_{Ac} \neg= NIL \wedge IS(D_{Cc}, C_{Cc})$$

$$==> (\exists F_{Ac})PROPDEF(D_{Cc}, t_P) = F_{Ac} \wedge (F_{Ac} = E_{Ac} \vee IS(F_{Ac}, E_{Ac}))$$

$$PROPDEF(C_{Ac}, t_P) = E_{Cc} \wedge E_{Cc} \neg= NIL \wedge IS(D_{Ac}, C_{Cc})$$

$$==> (\exists F_{Cc})PROPDEF(D_{Ac}, t_P) = F_{Cc} \wedge (F_{Cc} = E_{Cc} \vee IS(F_{Cc}, E_{Cc}))$$

$$IS(D_{Cc}, C_{Cc}) ==> (\exists t_P)IS(PROPDEF(D_{Cc}, t_P), PROPDEF(C_{Cc}, t_P))$$

$$IS(D_{Ac}, C_{Ac}) \Longrightarrow (\exists\, t_P)\, IS(PROPDEF(D_{Ac}, t_P), PROPDEF(C_{Ac}, t_P))$$

A továbbiakban elhagyjuk a sort-információt, mivel az a szövegkörnyezetből könnyen kikövetkeztethető.

3.3.2 Fogalom-definiciók axiómái

Egy fogalom-definició új szimbólumokat, azaz predikátumokat, függvényeket és konstansokat, valamint megfelelő axiómákat vezet be.

Egy tetszőleges fogalom-definició általános szintaxisa:

concept F; [(is-a | instance-of) G;]

[attributes a_1: D_1 [, a_k: D_k]$_2^n$;]

[constraints c_1: ξ_1 [, c_j: ξ_j]$_2^m$;]

amelyben legalább egy tulajdonságot, attributumot vagy kényszert meg kell adnunk.

E definició eredményeként, amennyiben a definició nem tartalmaz instance-of klauzulát, a fogalom osztály-konstansok listájához kell adnunk "F"-et, és a tulajdonság-konstansok listájához pedig a benne még nem szereplő "a_k"-kat és "c_j"-ket.

Az attributum-definiciókban szereplő D_k tartományok fogalmak, felsorolások (enumeration), illetve "mód" prefixszel ellátott fogalmak, és így e két utóbbi "in-line" módon új fogalom osztály-konstansokat vezet be. A kényszer-definiciók állításaira is egy-egy állítás osztály-konstanst vezetünk be.

Meglévő axiómáinkat pedig a következő axiómákkal bővítjük ki:

IS(F,G) (amennyiben a definicióban szerepel is-a klauzula)

$$PROPDEF(F,a_k) = C^{D_k} \qquad k=1,\ldots,n$$

$$PROPDEF(F,c_j) = A^{\S j} \qquad j=1,\ldots,m$$

A fenti axiómák nem tartalmaznak változókat.

Ha a "tartomány" valójában egy {k1,k2,...,kn} alakú "in-line" fogalom-definició, akkor a fogalom osztály-konstanson kívül létre kell hoznunk a k1 ,..., kn fogalom objektum-konstans-szimbólumokat is, hacsak nem léteznek már, és be kell vezetnünk a következő axiómákat:

$$INST(k1,C^{D}) \wedge \ldots \wedge INST(kn,C^{D})$$

$$INST(k,C^{D}) \Longrightarrow k = k1 \vee \ldots \vee k = kn$$

$$((\forall k)\,INST(k,C^{\mathfrak{D}}) \;\Longrightarrow\; INST(k,F)) \;\Longrightarrow\; IS(C^{\mathfrak{D}},F)$$

$$((\forall k)\,INST(k,F) \;\Longrightarrow\; INST(k,C^{\mathfrak{D}})) \;\Longrightarrow\; IS(F,C^{\mathfrak{D}})$$

A különböző attribútum-módok esetén bevezetendő axiómákat itt nem részletezzük.

Amennyiben a definícióban szereplő állítás nem tartalmaza szabad változóként a "THIS" speciális változót, akkor előírjuk, hogy

$$\xi \Longleftrightarrow (\exists y)\; INST(y,A^{\xi})$$

$$(\forall x)\; INST(x,F) \Longrightarrow ((\exists y)\; INST(y,A^{\xi}) \Longrightarrow PROPVAL(x,c) = y)$$

azaz A^{ξ}-nek akkor és csak akkor van konkrét példánya, ha ξ igaz, és megköveteljük, hogy a c kényszernek ez legyen az értéke, valahányszor létezik.

Ha viszont "THIS" szabad változóként fordul elő ξ-ben, akkor az A^{ξ} osztályt egyetlen "v:ANY" argumentum-tulajdonsággal kell definiálni, és az előbbi két axióma helyett a következő axiómákat kell bevezetnünk:

$$(\forall w)\; \xi(w/THIS) \Longleftrightarrow ((\exists x)\; INST(x,A^{\xi}) \wedge PROPVAL(x,v) = w)$$

$$(\forall x)\; INST(x,F) \Longrightarrow ([(\exists y)\; INST(y,A^{\xi}) \wedge PROPVAL(y,v) = x] \Longrightarrow PROPVAL(x,c) = y)$$

Ha a fogalom-definíció instance-of klauzulát tartalmaz, akkor egyszerûen a fogalom objektum-szimbólumok listájához adjuk "F"-et, és D_k-kat, ha még eddig nem tartalmazta volna ezeket, majd létrehozzuk a megfelelô állítás objektum-szimbólumokat, és elôírjuk, hogy

INST(F,G)

$PROPVAL(F,a_k) = O^{D_k}$ $\quad k=1,\ldots,n$

$PROPVAL(F,c_j) = P^{\S_j}$ $\quad j=1,\ldots,m$

3.4 A "logikai" szintû modell jelentése

Négy sort-ot vezetünk be, éspedig rendre az információknak, a tranzakcióknak, az állításoknak és a tulajdonságoknak megfelelôen. Az elsô három sort mindegyike Object és Class rész-sort-okra oszlik, így valójában összesen hét alapvetô sort áll a rendelkezésünkre.

A modellbeli szerkezetek jelentésének megmagyarázására a fogalmi szintû modell jelentésének megadásakor bevezetett alapvetô predikátumokat és függvényeket használjuk, szignaturáikat értelemszerûen megváltoztatva; ezt itt nem részletezzük.

Az alap-axiómák köre sem bôvül, így azok a 3.3.1-ben megadott axiómák alapján rutinszerûen nyerhetôk.

Az egységesség kedvéért a következő általános szintaxist ajánljuk:

{information | transaction} F;
[{is-a | instance-of} G;]

[attributes a_1 : D_1 [, a_k : D_k]$_2^n$;]

[constraints c_1 : ξ_1 [, c_j : ξ_j]$_2^m$;]

Egy ilyen definició értelmezése a fogalom-definiciók értelmezésének mintájára könnyen nyerhető.

A definiált attributum-tulajdonságokat - amennyiben ilyenek vannak - axióma-sémaként adhatjuk meg. Pl. az az attributum-tulajdonság, hogy az attributum nem vehet fel NIL értéket, a következőképpen definiálható:

ELENGEDHETETLEN(a) ≡ [PROPDEF(F,a) ¬= NIL]

==> [INST(O,F) ∧ PROPVAL(O,a) ¬= NIL]

Jellegzetesen tranzakciókra alkalmazható attributum-tulajdonság, ha

- az attributum-érték olyan objektum, amely részt vesz a definiálandó tranzakcióban, és a tranzakció indulásakor érdekes, azaz a tranzakció "inputja". Egy ilyen objektumot a tranzakció-objektum vesz át a megfelelő att-

ributum által definiált osztályból;

az attributum-érték olyan objektum, ami részt vesz a definiálandó tranzakcióban, és a tranzakció befejeződésekor érdekes, azaz a tranzakció "outputja". Egy ilyen objektumot a tranzakció-objektum ad a megfelelő attributum által definiált osztályhoz.

Szükségünk van természetesen olyan axiómákra, melyek állítják az összes számnak, mint a SZAMOK információ-osztály konkrét példányainak, és az összes karaktersorozatnak, mint a KARAKTERSOROZATOK információ-osztály konkrét példányainak létezését, valamint definiálják a szokásos aritmetikai- és karaktersorozatokon értelmezett operációkat és predikátumokat. Ezektől itt eltekintünk, de megjegyezzük, hogy amennyiben egyesítenünk kell olyan halmazelméleti fogalmak teljes szemantikáját, mint számosság, átlag stb., akkor az axiómákban nem-elsőrendű szerkezetekre lesz szükségünk.

Az

$$INST(n,SZAMOK) \wedge INST(n,F) \Longrightarrow IS(F,SZAMOK) \vee IS(SZAMOK,F)$$

$$INST(z,KARAKTERSOROZATOK) \wedge INST(z,F) \Longrightarrow IS(F,KARAKTERSOROZATOK) \vee IS(KARAKTERSOROZATOK,F)$$

axiómák biztosítják, hogy a számok és a karaktersorozatok minden más információtól különböznek.

4. FEJEZET

ESETTANULMANY

Az előbbi fejezetekben ismertetett módszer alkalmazásának érzékeltetésére bemutatjuk egy újszerű adatbáziskezelő rendszer tervezésének kezdeti lépéseit.

4.1 A kezdeti elképzelés

Olyan adatbáziskezelő rendszert szeretnénk tervezni, amely hatékony eszköz lenne azon adatbázis-felhasználók kezében, akiknek gyakran változó (esetenként rosszul definiált) sémákon kell végrehajtaniuk többnyire ad hoc tranzakciókat.

Miután sosem az adatbázisra (vagy valamilyen formában ennek valamely részére) vagyunk kíváncsiak, hanem "csupán" hasznos információt szeretnénk kapni,

- az ember-gép párbeszédet teljesen elkülönítjük az adatbázis modellektől; és

- az adattársítás szabadságának biztosítása érdekében nem használjuk a "rekord", "reláció", "alak" stb. fogalmakat.

Az ikonok kiválasztásával és sablonok kitöltésével kezdeményezhető ember-gép párbeszédet un. "képek" közvetítik, amelyek rendszerint értelmesen összekapcsolt adathalmazt mutatnak. Így a felhasználóknak nem kell megtanulniuk szintaktikus fogalmakat. Már tárolt adat újrabegépelése is felesleges, mert az információ bármely értelmes szövegkörnyezetben felidézhető és kiválasztással átültethető.

Ezen a szinten elképzelésünket pl. a következő módon formalizálhatjuk (bizonyos fogalmakat, pl. azonosító, név, tárolási mód stb. ismertnek tételezve fel).

```
notion adathalmaz;
  attributes
    azonosító: azonosító,
    tárolás: tárolási_mód,
    hozzáférhetőség: set-of hozzáférési_mód;
  constraints
    hozzáférhető: THIS.hozzáférhetőség  = NIL,
    azonosítható: THIS.azonosító egyedi;
```

```
notion adatbázis;
  case-of adathalmaz;
  attributes
    azonosító: név,
    modell: adatmodell,
    fogalmi_séma: séma,
    nézet: alséma,
    sémadefiniálás: sémadefiniálás_módja,
    alsémadefiniálás: alsémadefiniálás_módja,
    lekérdezés: lekérdezés_módja;
  constraints
    leír: az adathalmaz információt és tipikus
          kapcsolatokat ábrázol,
    számítógépben_tárolt: az adathalmaz többé-kevésbé
          tartósan számítógépben tárolt;

notion CoDB;
  case-of adatbázis;
  attributes
    fogalmi_séma: kép,
    nézet: set-of kép,
    sémadefiniálás: képeken értelmezett mûveletek
          segítségével,
    alsémadefiniálás: képeken értelmezett mûveletek
          segítségével,
    lekérdezés: képeken értelmezett mûveletek
          segítségével,
    képtárai: set-of képtár,
    állandó_képek: set-of kép,
```

```
    CoDB-n_értelmezett_parancsok:
          {Create, Open, Close, Delete, Copy, Move,
           Rename};
  constraints
    van_állandó_kép: THIS.fogalmi_séma  THIS.állandó_képek

notion képtár;
  attributes
    azonosító: név,
    adatbázis: CoDB,
    tárlat: set-of kép,
    képtáron_értelmezett_parancsok: {Create, Open,
          Close, Delete, Copy, Move, Rename};
  constraints
    azonosítható: THIS.azonosító egyedi,
    adatbázishoz_tartozik: THIS.adatbázis  = NIL és
          THIS  THIS.adatbázis.képtárai,
    nem_üres: THIS.adatbázis.állandó_képek  THIS.tárlat;

notion kép;
  attributes
    azonosító: név,
    tárló: képtár,
    minősége: {vázlat, vagyon, védett, műkincs, állandó},
    képeken_értelmezett_tranzakciók: {Open, Create, Close,
          Delete, Copy, Move, Rename, Requalify, Print},
    képeken_értelmezett_műveletek: kép_műveletek;
  constraints
    azonosítható: THIS.név THIS.tárló-ban egyedi;
```

4.2 Egy finomított változat

Induljunk ki a "képekkel" kapcsolatos elképzeléseinkből.

Elterjedt szokás, hogy egy bizonyos feljegyzéshez adott összefüggésben kapcsolódó dolgokat a feljegyzést követő sorban bekezdéssel kezdjük. Ez jegyzeteinket könnyen áttekinthetővé teszi: világosan, ránézésre elkülöníthetők a főbb gondolatok és kifejtéseik.

Ezzel a módszerrel akarjuk megjeleníteni az adatok közti kapcsolatokat is. Adatmodellként a bináris-kapcsolat (binary-relationship) modellt választjuk.

Ekkor tehát, ha a t_1 és t_2 típusok között az r kapcsolat definiált, akkor ez egy képben

t_1 $\qquad$ t_2

$r: t_2$ $\qquad$ vagy $\qquad$ $r: t_1$

alakban jelenhet meg, attól függően, hogy a felhasználó céljának az adott kontextusban melyik felel meg.

A képeken értelmezett operációk halmazát pedig úgy kell majd megválasztanunk, hogy azok egyaránt alkalmasak legyenek az adatbázis módosítására, adatok módosítására és lekérdezésére, riport-generálásra stb. Ennélfogva a képek tartalma tranziens, és a bennük foglalt információ érvényét veszítheti. A képek által tartalmazott információ érvényességének

ellenőrzése az adatbáziskezelő rendszer feladata.

A képtárakban így lesznek kiállított és archivált képek, melyek érvényes információt nyújtanak a szemlélőnek vagy felhasználónak. Ugyanakkor az érvénytelenné váló információt tartalmazó képek az érintett képtár műhelyébe kerülnek, a kidolgozás alatt lévő képek közé. További sorsukról, esetleges restaurálásukról a felhasználó dönthet.

A képek védetté vagy műkinccsé nyilvánításával biztosítható, hogy az e képek által hordozott információ sose váljon érvénytelenné, sőt, műkincsek esetén ez az információ nem is módosítható. Ennek megfelelően az adatbáziskezelő rendszer visszautasít minden olyan képen értelmezett műveletet, melynek következtében valamely képtárban egy védetté vagy műkinccsé nyilvánított kép megsérülne.

Az állandó képek tartalmát az adatbáziskezelő rendszer az adatbázis változásának megfelelően újra és újra módosítja úgy, hogy tartalmuk bármely művelet végrehajtása után ismét érvényes információt hordozzon.

E szempontokat figyelembe véve a kezdeti elképzelésünknek megfelelő formális leírás a következőképpen módosul (csak a pontosított, finomított definiciókat ismételve meg):

```
notion CoDB;
  case-of adatbázis;
  attributes
    modell: bináris-kapcsolat modell,
    fogalmi_séma: kép,
    nézet: set-of kép,
    sémadefiniálás: képeken értelmezett mûveletek
        segítségével,
    alsémadefiniálás: képeken értelmezett mûveletek
        segítségével,
    lekérdezés: képeken értelmezett mûveletek
        segítségével,
    képtárai: set-of képtár,
    állandó_képek: set-of kép;
    CoDB-n_értelmezett_parancsok:
        {Create, Open, Close, Delete, Copy, Move,
         Rename};
  constraints
    van_állandó_kép: THIS.fogalmi_séma  THIS.állandó_képek,
    állandó_kép_karbantartás: THIS.állandó_képek által
        tartalmazott információ karbantartása az adat-
        báziskezelô rendszer feladata,
    állagmegôrzés: THIS.képtárai-ban tárolt és védetté
        vagy mûkinccsé nyilvánított képek tartalmának
        védelme az adatbáziskezelô rendszer feladata;
```

```
notion képtár;
     attributes
       azonosító: név,
       adatbázis: CoDB,
       kiállítás: set-of kép,
       archívum: set-of kép,
       mûhely: set-of kép,
       képtáron_értelmezett_parancsok:
          {Create, Open, Close, Delete, Copy, Move,
           Rename};

     constraints
       azonosítható: THIS.azonosító egyedi,
       adatbázishoz_tartozik: THIS.adatbázis  = NIL
          és THIS  THIS.adatbázis.képtárai,
       karbantartás: kép  THIS.kiállítás vagy
          kép  THIS.archívum => kép.nyújtott_információ =
          = érvényes,
       kiállítás_nem_üres: THIS.adatbázis.állandó_képek
            THIS.kiállítás;
```

```
notion kép;
  attributes
    azonosító: név,
    tárló: képtár,
    helye: {kiállítás, archívum, műhely},
    tartalma: forest-of képsor,
    nyújtott_információ: {érvényes, érvénytelen},
    minősége: {vázlat, vagyon, védett, műkincs, állandó},
    képeken_értelmezett_tranzakciók:
      {Open, Create, Close, Delete, Copy, Move, Rename,
       Requalify, Print},
    képeken _értelmezett_műveletek: kép_műveletek;
  constraints
    azonosítható: THIS.név THIS.tárló-ban egyedi,
    restaurálható: THIS.nyújtott_információ = érvény-
       telen => THIS.helye = THIS.tárló.műhely,
    felhasználóé: THIS.minősége = állandó => THIS.
       tartalma csak a felhasználó kezdeményezésére
       változtatható meg;

notion állandó_kép;
  case-of kép;
  attributes
    képeken_értelmezett_tranzakciók:
       {Open, Close, Copy, Print};
  constraints
    read_only: THIS.képeken_értelmezett_műveletek = NIL,
    kiállított: THIS.helye = THIS.tárló.kiállítás;
```

```
notion remekmű;
  case-of kép;
  attributes
    minősége: (védett, műkincs);
  constraints
    kiállított: THIS.helye = THIS.tárló.kiállítás,
    biztosítottt: THIS.nyújtott_információ = érvényes;


notion képsor;
  attributes
    bekezdés: (NIL, szóközök lánca),
    relációnév: NIL   név,
    típusnév: NIL  név,
    adatmező: NIL  szöveg;
  constraints
    üres_sor_nem_képsor_: THIS.relációnév  = NIL vagy
         THIS.típusnév  = NIL vagy THIS.adatmező  = NIL;
```

Ezt az eljárást a tervezés folyamán következetesen alkalmazva fokozatosan kialakíthatjuk adatbáziskezelő rendszerünk fogalmi modelljét, amely már tartalmazza a képeken értelmezett műveletek meghatározásait, az adatbázison és a képtáron értelmezett parancsok definicióit, és a képeken értelmezett tranzakciók leírásait.

5. Fejezet

ÖSSZEGZÉS

Célkitűzésünknek megfelelően az ismert fogalmi módszertanok megközelítéseinek korlátait próbáltuk átlépni. Két ilyen területet vizsgáltunk

- a rendszertervezés kezdeti szakaszait, a problémamegismerés, a környezetfeltárás és a specifikációkialakítás fázisait; valamint

- a modellek és reprezentációik dinamikus törvényszerűségeit, feltárva azokat a transzformációkat, melyek definíció szintű utólagos átalakítások esetén is biztosítják az ismeret-házis összhangját.

Ilyen típusú fogalomrendszerek gépi kezelésének megvalósítása céljából több irányban indítottunk kísérleti munkákat. Az egyik ilyen típusú újabb kutatási elképzelésünk a [HE'88a], [BOD'88], [HE'88b] dokumentumokban található.

IRODALOMJEGYZÉK

AB'74 Abrial, J.R. Data semantics. In: Data Base Management, J.W. Klimbie, K.L. Koffeman, eds., North-Holland, 1974, pp.1-60.

ALB'85 Albano, A., Cardelli, L., Orsini, R. Galileo: A strongly typed, interactive conceptual language. ACM Trans. on Database Systems, Vol.10, No.2, July 1985, pp.230-250.

ALLE'83 Allen, J. Maintaining knowledge about temporal intervals. Communications of the ACM, 26(11), Nov. 1983, pp.832-843.

AND'81 Anderson, T.L. The Database Semantics of Time. PhD Thesis, University of Washington, 1981.

BOD'88 Bodó, Z., Hernádi, A., Knuth, E. "coDB" Common Base Definition. Reference Manual, Part I, Version 5. Computer and Automation Institute, Hungarian Academy of Sciences, Budapest 1988.

BOR'81 Borgida, A.T., Greenspan, S. Data and Activities: Exploiting Hierarchies of Classes. Proc. of the Workshop on Data Abstraction, Databases and Conceptual Modelling, Pingree Park, Colorado, June 23-26, 1980. (SIGPLAN Notices, Vol.16, No.1, January 1981, pp.98-100.)

BOR'84a Borgida, A., Mylopoulos, J., Wong, H.K.T. Generalization/Specialization as a Basis for Software Specification. In: On Conceptual Modelling, Brodie, M.L., Mylopoulos, J., Schmidt, J.W., eds., Springer Verlag, New York, 1984, pp.87-114.

BOR'84b Borgida, A. Intelligent Handling of Exceptions in Information Systems - An Overview. Proc. of the First International Workshop on Expert Database Systems, L. Kerscheberg, ed., Institute for Information Management, Technology and Policy, University of South Carolina, Columbia, South Carolina, Oct.1984, pp.643-651.

BOU'83 Bourne, S.R. The UNIX System. Addison-Wesley, 1983.

BRA'79 Brachman, R.J. On the Epistemological Status of Semantic Networks. Associative Networks: Representation and Use of Knowledge by Computers, N.V. Findler, ed., Academic Press, New York, 1979, pp.3-50.

BRO'84 Brodie, M.L., Mylopoulos, J., Schmidt, J.W. On Conceptual Modelling. Springer Verlag, New York, 1984.

CH'76 Chen, P. The Entity-Relationship Model - Toward a Unified View of Data. ACM Transactions on Database Systems, Vol.1, No.1, pp.9-36, March 1976.

DAH'70 Dahl, O.J., Myhrhaug, B., Nygaard, K. SIMULA 67 Common Base Language. Publication S-22, Oct.1970. Norwegian Computing Center. Forskningsvien 1B Oslo 3, Norway.

DAT'83 Date, C.J. An Introduction to Database Systems, Vol.II. Addison-Wesley, 1983.

DE'86 Demetrovics, J., Knuth, E., Radó, P. Computer aided specification techniques. World Scientific Press, Series in Computer Science, Vol.1,pp.1-114, 1986.

EN'72 Enderton, H.B. A mathematical introduction to logic. Academic Press, 1972.

GI'85 Gibbs, S.J. Conceptual Modelling and Office Information Systems. In: Tzichritzis, D., ed., Office Automation, pp.194-224, Springer Verlag, 1985.

GO'83a Goldberg, A., Robson, D. Smalltalk-80. The Language and its Implementation. Addison-Wesley, 1983.

GO'83b Goldberg, A. Smalltalk-80. The Interactive Programming Environment. Addison-Wesley, 1983.

GR'86 Greenspan, S.J., Borgida, A., Mylopoulos, J. A Requirements Modelling Language and Its Logic. Information Systems, Vol.11, No.1, 1986, pp.9-23.

HE'86a Hernádi, Á. Abstraction Mechanisms in Data Modelling; Proc. of the 9th Int. Seminar on Database Management Systems, Reinhardsbrunn, GDR, November 30-December 5, 1986, pp.177-188.

HE'86b Hernádi, Á. A típus fogalma és szerepe a modellezésben - absztrakt adattípusok alkalmazásának új elveiről. MTA SZTAKI Tanulmányok 190/1986, Budapest 1986, p.60.

HE'87 Hernádi, A., Bodó, Z., Knuth, E. A tudásábrázolás technikái és gépi eszközei (felmérő tanulmány). MTA SZTAKI Tanulmányok 197/87, Budapest 1987. p.213.

HE'88a Hernadi, A., Bodó, Z., Knuth, E. A Different Interactive Intreface for Database Management Systems. Seregelyes, 1988 (megjelenés alatt).

HE'88b Hernádi, A., Bodó, Z., Knuth, E., Radó, P. Pictures of a Data Exhibition. 4th IFAC/IFIP/IFORS/IEA Conference on Man-Machine Systems '89 (megjelenés alatt).

KI'84 King, R., McLeod, D. A Unified Model and Methodology for Conceptual Database Design. In: On Conceptual Modelling, Brodie, M.L., Mylopoulos, J., Schmidt, J.W., eds., Springer Verlag, New York, 1984, pp. 313-327.

KN'82 Knuth, E., Halász, F., Radó, P. SDLA, System Descriptor and Logical Analyzer. In: Information Systems Design Methodologies: A Comparative Review; Olle, Sol and Veerijn-Stuart, eds., North Holland, pp.143-147, 1982.

KN'85 Knuth, E:, Rónyai, L. Closed Convex Reference Schemes. Proc. of the IFIP TC-2 Conference on System Description Methodologies, Kecskemét. System Description Methodologies, North Holland, 1985, pp.435-453.

KN'86a Knuth, E., Hannák, L., Hernádi, A.: Notes on Conceptual Representations. In: Proc. of the ACM SIGART International Symposium on Methodologies for Intelligent Systems; Knoxwille, Tennessee, October 22-24, 1986: Ras, Z.W., Zemankova, M. eds., pp.390-398.

KN'86b Knuth, E., Hannák, L., Radó, P. A Taxonomy of Conceptual Foundations. IFIP DS-2 Conference on Knowledge and Data, 1986.

KN'86c Knuth, E., Demetrovics, J., Hernádi, A. Information System Design: On Conceptual Foundations. Proc. of the IFIP 10th World Computer Congress, Dublin, Ireland, September 1-5, 1986, Information Processing 86, H.J. Kugler, ed., Elsevier Science Publishers B.V., North-Holland, 1986, pp.635-640.

KN'87 Knuth, E., Demetrovics, J., Hernádi, A. On Transformation Properties of Conceptual Structures; Methodologies for Intelligent Systems, Ras, Z.W., Zemankova, M., eds., Elsevier Science Publishing Co. Inc., North-Holland, 1987, pp. 217-223.

KN'88 Knuth, E., Kiss, O., Hernadi, A. Key Issues in Manufacturing Information Management. Proc. of the 54th World Congress of IACTF, Delhi, India, 1987 (megjelenés alatt).

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

LI'78 Liskov, B., Moss, E., Schaffert, C., Scheifler, B., Snyder, A. CLU Reference Manual. Massachusetts Institute of Technology, Laboratory for Computer Science, Computation Structures Group, Memo 161, July 1978.

LUD'84 Ludewig, J., Mitchell, R. Incompleteness and Abstraction in Program Description. In: International Workshop on Models and the Languages for Software Specifications and Design; Orlando, Florida, 1984.

MCA'68 McCarthy, J. Programs with common sense. In: Minsky, M., ed., Semantic information processing, The MIT Press, 1968.

MCAW'81 McCawley, J. Everything linguists ever wanted to know about logic but were afraid to ask. University of Chicago Press, 1981.

MD'81 McDermott, D. A temporal logic for reasoning about processes and plans. Technical Report 196, Dept. of Computer Science, Yale University, March 1981.

ML'81 McLeod, D., Smith, J.M. Abstraction in Databases. Proc. of the Workshop on Data Abstraction, Database and Conceptual Modelling, Pingree Park, Colorado, June 23-26, 1980; SIGPLAN Nptices, Vol.16, No.1, January 1981, pp.19-25.

MON'73 Montague, R. The proper treatment of quantification in ordinary English. In: Hintikka, J., Moravics, J., Suppes, P., eds., Approaches to natural languages, Reidel, 1973, pp.221-242.

MY'80a Mylopoulos, J., Bernstein, P.A., Wong, H.K.T. A language facility for designing database-intensive applications. ACM Transactions on Database Systems, Vol.5, No.2, June 1980, pp.185-207.

MY'80b Mylopoulos, J., Wong, H. Some Features of the TAXIS Data Model. Proc. of the 6th International Conference on Very Large Databases, Montreal, Canada, October 1980.

MY'84 Mylopoulos, J., Levesque, H. An Overview of Knowledge Representation on Conceptual Modelling. In: Brodie, M., Mylopoulos, J. and Schmidt, J., eds., On Conceptual Modelling, Springer Verlag, New York, 1984, pp.3-18.

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

RES'71 Rescher, N., Urquhart, A. Temporal Logic. Springer Verlag, 1971.

SER'80 Sernadas, A. Temporal aspects of logical procedure definition. Information Systems 5, No.3, 1980, pp,167-187.

SH'81 Shipman, D. The Functional Data Model and the Data Language DAPLEX. ACM Transactions on Database Systems, Vol.6, No.1, March 1981, pp.140-173.

SM'77a Smith, J.M., Smith, D.C.P. Database Abstraction: Aggregation. Communications of the ACM, Vol.20, No.6, June 1977, pp.405-413.

SM'77b Smith, J.M., Smith, D.C.P. Database Abstractions: Aggregation and Generalization. ACM Transactions on Database Systems, Vol.2, No.2, June 1977, pp.105-133.

SM'78 Smith, J.M., Smith, D.C.P. Principles of Conceptual Database Design. Proc. of NYU Symposium on Database Design, 1978; New York, NY, 18-19 May 1978.

SM'79 Smith, J.M., Smith, D.C.P. A Database Approach to Software Specification. Technical Report CCA-79-17, Computer Corporation of America, Cambridge MA, April 1979.

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

STAN'87 Stanley, M. CML. Master Thesis. University of Toronto, Department of Computer Science, 1987.

TEI'79 Teichroew, D. et al. Application of the Entity-Relationship Approach to Information Processing System Modelling. Proc. of the International Conference on Entity-Relationship Approach to Systems Analysis and Design, Los Angeles, CAL, December 10-12, 1979.

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

UL'82 Ullmann, J.D. Principles of Database Systems. Second Ed., Computer Science Press, 1982.

WIE'84 Wiederhold, G. Knowledge and Database Management. IEEE Software, Vol.1, No.1, pp. 63-73, January 1984.

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