Írta:
LAKNER ROZÁLIA
HANGOS KATALIN
GERZSON MIKLÓS

INTELLIGENS IRÁNYÍTÓ RENDSZEREK

Egyetemi tananyag

TYPOTEX

2011

TÁMOGATÁS:
Készült a TÁMOP-4.1.2-08/1/A-2009-0008 számú, „Tananyagfejlesztés mérnök informatikus, programtervező informatikus és gazdaságinformatikus képzésekhez” című projekt keretében.

ISBN 978-963-279-511-9

KÉSZÜLT: a Typotex Kiadó gondozásában
FELELŐS VEZETŐ: Votisky Zsuzsa
AZ ELEKTRONIKUS KIADÁST ELŐKÉSZÍTETTE: Juhász Lehel

KULCSSZAVAK:
intelligens irányítás, real-time szakértői rendszerek, Petri hálók, kvalitatív modellek, szabályrendszerek, fuzzy következtetés

ÖSZEFOGLALÁS:
Az intelligens irányító rendszerek tankönyv anyaga a napjainkban dinamikusan fejlődő intelligens technikákat alkalmazó rendszer- és irányításelméleti feladatok (predikció, beavatkozás tervezés, rendszeranalízis, diagnosztika) megoldása során hasznosítható elméleti ismereteket és technikai megoldásokat tartalmazza műszaki informatikus hallgatók számára MSc szinten. A tankönyv fejezetei sorra veszik a folyamatirányító szakértői rendszer fogalmával és elemeivel, a tudásábrázolással, kvalitatív modellezéssel, Petri hálókkal, valamint a fuzzy irányítási rendszerekkel kapcsolatos tudnivalókat. A könyvben összefoglalt tananyagot a G2 folyamatirányító real-time szakértői rendszer rövid ismertetése egészíti ki. A fenti fejezeteket animált ábrák, valamint kidolgozott és megoldandó feladatok gazdagítják.

Tartalomjegyzék

1. fejezet

Bevezetés

Az „Intelligens irányító rendszerek” tárgy tematikáját még a Veszprémi Egyetem műszaki informatika szakára dolgoztuk ki választható tárgyként, amelyet az 1990-es évek elejétől kezdve folyamatosan, évente tartottunk nagy érdeklődés mellett. A tárgyhoz egy magyar nyelvű egyetemi jegyzet is készült [5], ám a tárgy tematikájának folyamatos frissítése miatt ez a 2000-es évek elejére elavulttá vált. Ekkor jelent meg egy kibővített angol nyelvű tankönyv [6], amelynek megvásárlását a kurzus hallgatói általában sajnos nem engedhették meg maguknak.

A kétfokozatú bologna-i rendszerű képzésben ez a tárgy a Pannon Egyetem mérnök infomatikus MSc képzésében kapott helyet kötelezően választható tantárgyként, és népszerűsége szerencsére nem csökkent. Közben a tárgy tematikája szintén kissé megújult, ezért szükségessé vált egy újabb segédanyag elkészítése. Nagy örömünkre szolgál, hogy ezúttal egy magyar nyelvű, a magyar hallgatók által szabadon hozzáférhető tankönyv készülhetett el a TÁMOP program támogatásával.

Mint a tárgy nevéből is látszik, az „Intelligens irányító rendszerek” épít a hallgatók BSc képzés során elsajátított mesterséges intelligencia és irányítástechnikai alapismereteire, de ezek tudását az új ismeretek fényében nagyban elmélyíti és megerősíti.

Az intelligens irányító rendszerek tématerülete egy viszonylag új, izgalmas, gyorsan fejlődő és intenzíven kutatott terület, ahol még nem alakult ki az ismeretek szintetizált, egységes rendszere. Ezért a tankönyv két alapozó, a folyamatirányító szakértői rendszerekkel és a tudáseprezentációval foglalkozó fejezet után egymással lazán kapcsolódó, tetszőleges sorrendben feldolgozható fejezetekből áll, amelyek egyike-másika egyes esetekben igény szerint elhagyható, vagy más modulokkal kicserélhető.

Az egyes fejezetekhez - a terjedelmi korlátok által megengedett mértékben - igyekeztünk kidolgozott példákat is biztosítani, ami a gyakoratok vagy laboratóriumi foglalkozások anyaga lehet. Egy teljes fejezetet szenteltünk a gyakorlati alkalmazásokban leginkább bevált, bár eleggé drága, és ezért nem széles körben hozzáférhető G2 folyamatirányító szakértői rendszer részletes ismertetésének.

Reméljük, hogy tankönyvünk nemcsak az érdeklődő hallgatókat segíti majd abban, hogy az intelligens irányító rendszerek érdekes és szép világával megismerkedjenek, hanem kollégáinknak is kedvet csinál hasonló kurzusok megtartásához, és ennek során jegyzetünk segítségükre lesz.

Veszprém, 2010. szeptember 30.

Lakner Rozália, Hangos Katalin és Gerzson Miklós
Pannon Egyetem
Műszaki Informatikai Kar

2. fejezet

Folyamatirányító szakértői rendszerek

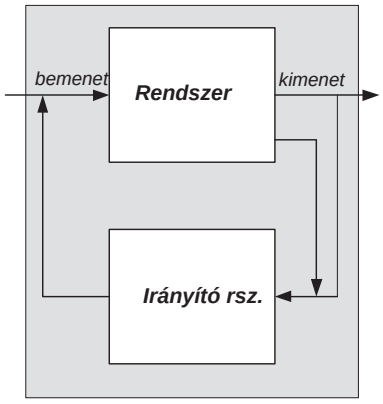
Az intelligens irányító rendszer fogalmának megértéséhez a (folyamat)irányító rendszer és az intelligens rendszer fogalmaiból indulunk ki, valamint megismerkedünk azokkal a technikákkal, amelyek e két különböző részrendszer együttes alkalmazását teszik lehetővé.

2.1. Intelligencia az irányító rendszerekben

A *rendszer*t a való világ egy részének tekinthetjük, amely környezetétől elhatárolt, s a környezethez határain keresztül kapcsolódik. A környezet rendszerre történő hatását input, a rendszer környezetére gyakorolt hatását pedig output jelek formájában írhatjuk le, ahol *jel*eknek az időben változó mennyiségeket tekintjük.

Az *irányító rendszer*ek feladata dinamikus, azaz időfüggő viselkedéssel bíró rendszerek meghatározott célú működtetése. Ennek eléréséhez a rendszer input és output jeleinek ismeretében olyan input megtervezése a feladat, amely az előre definiált irányítási célt kielégíti.

Az irányító rendszerek realizálása legtöbbször számítógép(ek) alkalmazásával történik, így ezek a rendszerek alapvetően valós-idejű szoftver rendszerekként értelmezhetők, amelynek fő elemei a következők:



2.1. ábra. Irányított rendszer

- adat fájlok, adatstruktúrák (nyers mért adatok, mért adatok, események, stb.),
- taszkok, algoritmusok (elsődleges feldolgozás, eseménykezelés, stb.),
- interfészek (taszk-adat, taszk-taszk, operátor-számítógép között).

A számítógéppel irányított rendszerek fő funkciói az adatgyűjtés (mérés), adatfeldolgozás, irányítás (szabályozás), rendszer analízis, identifikáció és diagnózis. Ezen feladatok többségénél szükséges *heurisztikus*, általában tapasztalati tervezési, üzemeltetési vagy karbantartási ismeretek leírására és kezelésére a mesterséges intelligencia módszerei alkalmazhatók.

Az intelligencia, ezen belül a számítógépes intelligencia fogalma számos vita tárgya a szakirodalomban. Általában azt mondhatjuk, hogy egy *intelligens rendszer* az emberhez hasonló módon old meg nagy bonyolultságú feladatokat. Az emberi gondolkodás menete általában kevésbé pontos, nem részletes, ugyan-akkor lényeges jellemzője a heurisztika, azaz a tapasztalat, intuíció által irányított problémamegoldás, amely során új körülmények között is jól alkalmazhatók a korábbi ismeretek. Ebben fontos szerepet játszik az összegyűjtött ismeretek rendszerezésével és manipulálásával megvalósított tanulás.

Az intelligens problémamegoldást igénylő feladatok közös vonásai az alábbiakban foglalhatók össze:

- általában *nehezek* (még az ember számára is!),
- *nem rendelkeznek* minden részletében tisztázott *fix megoldó mechanizmussal*,
- a megoldás *elemi tevékenységek sorozata*ként állítható elő (ez előre nem rögzített, s általában több, nem egyforma megoldási út létezik),
- a problémamegoldás *kereséssel* történik (minden választási helyzetben *szisztematikus próbálkozás*sal választjuk ki a következő „lépést”),
- emberi szakértelem, intuíció, gyakorlati tapasztalat, azaz *heurisztikus ismeret* szükséges a keresés irányításához/korlátozásához (a probléma tere nagy lehet, ezért az összes lehetőség kipróbálása szisztematikus úton a kombinatorikus robbanás problémája miatt nem lehetséges),
- *„elég kedvező”* megoldás elégséges,
- ma általában *az ember a jobb*.

A felsorolt tulajdonságokkal rendelkező feladatok közé tartoznak többek között a kirakós játékok, a sakk, a tételbizonyítás, a diagnózis és a szövegfordítás.

Általánosan megállapítható, hogy az intelligens rendszerek

- nehéz (nem-triviális, bonyolult, nagyméretű, összetett) feladatot oldanak meg,
- nem-triviális, az emberhez hasonló módon.

Az intelligens rendszerek legjellegzetesebb közös vonása a *heurisztikával vezérelt keresés*.

Az intelligencia megjelenésére (folyamat)irányító rendszerekben akkor van szükség, ha az irányítási feladatok legalább egyike intelligens problémamegoldást igényel. Ebben az esetben *intelligens irányító rendszer*ről beszélhetünk.

2.1.1. Tudásalapú rendszerek fogalma és felépítése

A *tudásalapú rendszerek (angolul knowledge based systems)* olyan Neumann elvű programstruktúrával rendelkező intelligens rendszerek, amelyekben az adatszerű passzív ismeretek külön vannak választva a végrehajtó aktív résztől. Egy tudásalapú rendszer tehát két fő részből áll, amelyek a következők:

- *tudásbázis*,
- *következtető gép*.

A *tudásbázis* tartalmazza egyrészt a problématerületet leíró specifikus ismereteket (tudást) általában valamilyen természetes nyelvhez közeli formalizmussal leírva, másrészt a konkrét feladat kiinduló és közbenső adatait egyszerű adatelemek formájában. Gyakran e kétféle komponenst fizikailag is kettéválasztják, és az adatokat *eset-specifikus adatbázis*ban vagy más néven *munka-memória*ban tárolják.

A *következtető gép* a feladatmegoldás „motorja”, amely általános problémamegoldó módszereket (beleértve a megoldáskereső módszereket) és egyéb szolgáltatásokat tartalmaz. Az adatok és tudás leírására alkalmas reprezentációs technikákról, valamint a tudásalapú rendszerek problémamegoldó módszereiről bővebben a 3. fejezetben lesz szó.

Az intelligens irányító rendszerek dinamikus rendszerek, ezért a tudásbázisukban szereplő adatok (pl. reaktor hőmérséklete 50 °C, „A” szelep zárva) nem csak a problémamegoldás során, hanem az időben is változhatnak. A tudásbázisukban szereplő bonyolultabb összefüggések tartalmazzák a heurisztikus információkat (pl. Ha a nyomás határérték feletti, akkor zárjuk a szelepet.), amelynek leírása leggyakrabban szabályok (lásd 3. fejezet) segítségével történik. A tudásbázis ezen része csak akkor változik, ha a szakterület ismereteinek módosítására van szükség.

2.1.2. Szakértői rendszerek

A *szakértői rendszerek* olyan tudásalapú rendszerek, amelyek szakértői szintű ismeretek felhasználásával egy szűk, de elég bonyolult, szakértelmet igénylő problématerület kezelésében nyújtanak kimagasló teljesítményt.

Egy szakértői rendszertől elvárt szolgáltatások a következők:

- az emberi szakértőhöz hasonlóan *javaslat*okat adjon egy probléma megoldá-sához,
- *kérdés-válasz* formájában kommunikáljon, s legyen „egyenrangú beszélgető partner”,
- feltett *kérdés*eihez szükség esetén adjon *magyarázat*ot,
- *javaslat*ait szükség esetén *indokol*ja,
- *bizonytalan* körülmények között is képes legyen *elfogadható javaslat*ot adni.

A szakértői rendszertől elvárt szolgáltatások biztosításához a tudásalapú rendszereknél megismert elemeken (tudásbázis, munkamemória, következtető gép) kívül további elemekre van szükség, amelyek a következők:

- *magyarázó alrendszer*, amely a rendszer akcióit magyarázza meg felhasználói kérésre feladatmegoldás közben és feladatmegoldás után,

- *tudásbázis kezelő/fejlesztő alrendszer*, amelynek feladata szolgáltatások nyúj-tása a tudásbázis építéséhez, teszteléséhez és módosításához; általában rendelkezik tudásbázis-fejlesztő eszközökkel, teszt-eseteket tartalmazó
könyvtárral és tudásszerzést támogató szolgáltatásokkal,

- *felhasználói felület*, amely a rendszer és felhasználója közötti kapcsolatot biztosítja természetes nyelvű párbeszéd formájában,

- *fejlesztői felület*, amely a tudásmérnök (illetve a vele szoros kapcsolatban együttműködő tárgyköri szakértő) számára biztosít lehetőleg felhasználóbarát felületet a tudásbázis kezeléséhez,

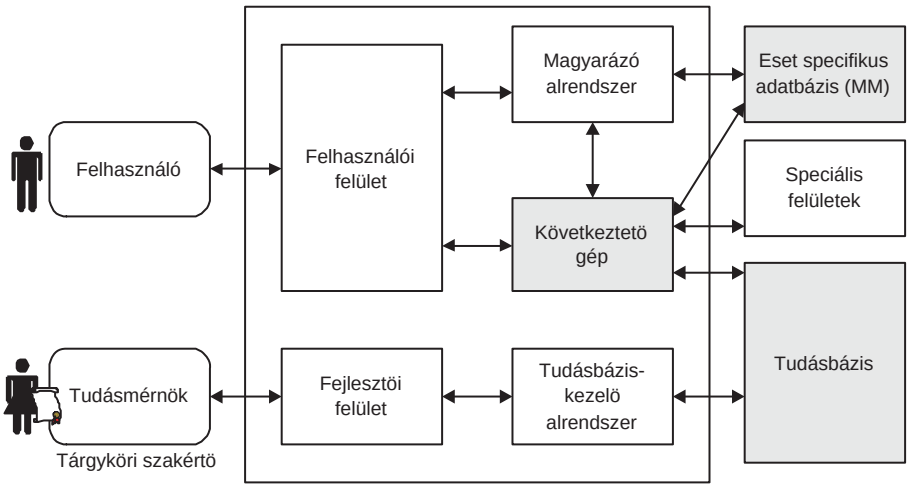
- *speciális felületek*, amelyek az adatbázis- és egyéb (például valós-idejű rendszerrel történő) kapcsolatokat biztosítják.

A szakértői rendszer *felhasználó*ja a hagyományos programok felhasználóinál nagyobb szerepet tölt be, hiszen a rendszerrel aktív párbeszédet folytatva egyenrangú partnerként vesz részt a feladat megoldásában, a rendszer javaslatait a magyarázatok figyelembe vételével értékeli, s ezek alapján maga dönt. A rendszerben megjelenő új szereplő, a *tudásmérnök* feladata a tudásbeszerzés és a tudás adott formába öntése (a tudásbázis tervezése és feltöltése, annak ellenőrzése, karbantartása). Ezen feladatokat az adott szakterület ismereteivel rendelkező *tárgyköri szakértő*vel együttműködve végzi. Az Intelligens irányító rendszerek című tantárgy egy tudásmérnök alapismereteként szükséges módszereket és eljárásokat foglalja össze.

A szakértői rendszerek alapvető elemei és felépítése a 2.2 ábrán látható. A szakértői rendszerek fejlesztéséhez alkalmas eszközök között kitüntetett szerepe van az üres tudásbázissal és erőteljes tudásbázis kezelő/fejlesztő alrendszerrel rendelkező *szakértői keretrendszer*eknek (vagy shell-eknek), amelyek a tárgyterülettől független szolgáltatásokat nyújtanak szakértői rendszerek létrehozásához és működtetéséhez, valamint támogatják a gyors prototípuskészítést és inkrementális rendszerépítést. A szakértői keretrendszerek elemei a 2.2. ábrán szaggatott keretben láthatók, a G2 szakértői keretrendszer bemutatásával a 7. fejezet foglalkozik.

2.2. Folyamatirányító szakértői rendszerek megvalósítása

Az intelligens technikát igénylő irányítási feladatok a probléma jellegéből adódóan többfélék lehetnek, s ennek megfelelően többféle módszert alkalmazhatnak a megoldás során. Például abban az esetben, ha a feladat megfogalmazása nem pontos (nem tudjuk a modellt) a bizonytalanság leírása történhet fuzzy modell vagy kvalitatív modell segítségével. A heurisztikus (jellemzően üzemeltetési) tudás leírására leggyakrabban a „minta → akció” hatásokat tartalmazó szabályokat alkalmazzák, s a feladatmegoldást logikai következéssel végzik. Az intelligens irányító rendszerek megvalósítása során legnagyobb problémát az okozza, hogy a mesterséges intelligencia klasszikus módszereit általában a statikus esetekre dolgozták ki, így az időbeli viselkedés leírásáról és kezeléséről külön kell gondoskodni.



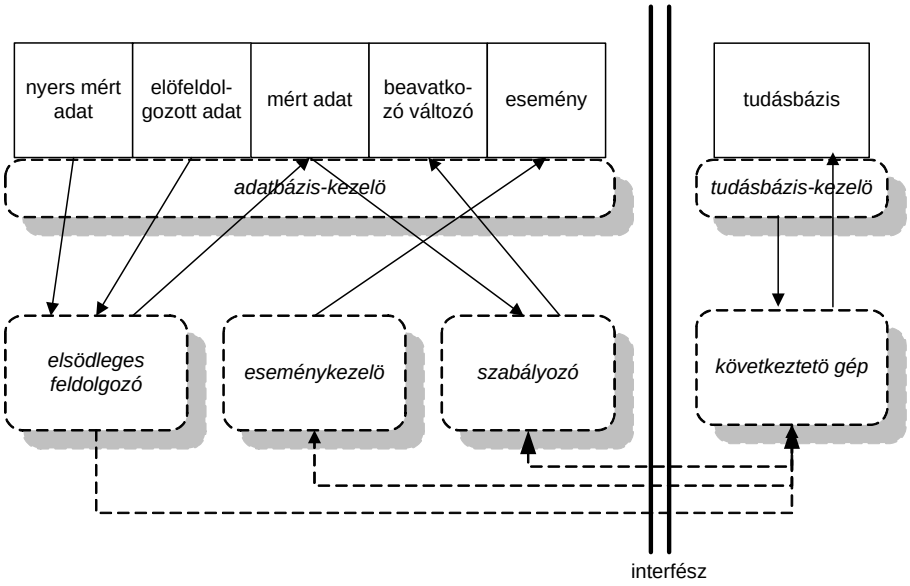
2.2. ábra. A szakértői rendszer elemei

2.2.1. Folyamatirányító szakértői rendszerek komponensei

Az irányítástechnikai célra alkalmas szakértői rendszerekben általában külön (gyakran két külön számítógépen futó) szoftver komponensként valósítják meg a szakértői rendszert, mint alrendszert és a real-time irányító rendszert, mint alrendszert, amelyek kapcsolatát egy dedikált interfész alkotja. Ezt az architektúrát szemlélteti a 2.3 ábra, ahol az interfészt kettős függőleges vonallal jelöltük, tőle jobbra a szakértői alrendszer, balra pedig az irányító alrendszer szokásos komponensei láthatóak. A komponensek közötti kapcsolatokat nyilak jelzik az ábrán, a szaggatott nyilakkal jelölt kapcsolatok szinkronizációs, a folytonos vonallal jelzettek adat-kapcsolatoknak felelnek meg.

Az irányító alrendszer Egy mérés-adatgyűjtési, irányítási és diagnosztikai feladatokat is ellátó szoftver komponensnek valós idejű, azaz *real-time* viselkedést kell mutatnia, azaz az alábbi kulcsfontosságú tulajdonságokkal kell bírnia:

- *időfüggő viselkedés*, azaz az a képesség, hogy adott időpillanatokban, adott időtartam elteltével, vagy adott esemény bekövetkezése esetén előírt tevékenységeket elvégezzen,
- *véges válaszidő*vel kell rendelkeznie, azaz minden tevékenységnek egy adott időtartam elteltével biztosan be kell fejeződnie,
- képesnek kell lennie *time-out*-ra, azaz arra, hogy egy adott időtartam eltelte után az elkezdett tevékenységet megszakítsa és alapállapotba hozza (abortálja),
- működésének *nyers adatvesztés-mentes*nek kell lennie,
- *prioritás-kezelési képesség*gel kell rendelkeznie,
- viselkedésének „bájosan” kell elhalnia (*nice degradation*), azaz a terhelés növekedésével növekvő prioritási sorrendben kell e tevékenységeket elhagynia.



2.3. ábra. A folyamatirányító szakértői rendszer tipikus elemei

Az irányító alrendszernek a 2.3 ábra baloldalán látható, szaggatott vonallal határolt lekerekített téglalappal jelölt szokásos processzei közül az alábbiak szoktak a szakértői alrendszerrel kapcsolatban állni:

1. az *elsődleges feldolgozás (primary processing)*, amely a nyers mért adatokból előfeldolgozott, mérnöki egységre átszámított és státuszjelzéssel ellátott mért adatokat állít elő,

2. az *eseménykezelő (event handling)*, valamint a

3. tágabb értelemben vett *szabályozók (controller(s))*, amelyek között például diagnosztikai processzek is lehetnek.

A szakértői alrendszer A szakértői alrendszer a 2.1.2 alfejezetben leírt általános jellemzőkkel bíró szoftver komponens, amelynek a két alrendszer közötti interfész megvalósítása szempontjából az alábbi lényeges tulajdonságai vannak:

(A) A tudásbázisban tárolt elemek szintaktikai és szemantikai szempontból is erősen összefüggőek, így egy következtetés során a konzisztencia biztosítása érdekében a teljes tudásbázist le kell foglalni egy következtető gépnek.

(B) A következtetés maga NP-nehéz, azaz algoritmikusan nagy-bonyolultságú feladat, az eredmény előállításához a feladat méretével exponenciálisan növekvő számú műveleti lépés lehet szükséges. Ezért csak „laza” kapcsolat valósítható meg az irányító alrendszerrel, hogy annak véges válaszidejű tulajdonsága megmaradjon.

2.2.2. A szakértői és az irányító alrendszerek kapcsolata

A folyamatirányító szakértői rendszerek két alrendszerének összekapcsolására két út kínálkozik. A közönséges alrendszerek összekapcsolásánál elterjedten használatos *szoros összekapcsolás*, ahol közös adatbázis felett, az operációs rendszer és az adatbázis kezelő szokásos szinkronizációs szolgáltatásai révén valósul meg a kapcsolat. Ez a szakértői alrendszer fenti, a 2.2.1 alfejezetben leírt (A) és (B) tulajdonságai miatt nem megfelelő az irányító alrendszertől elvárt real-time viselkedés szempontjából.

Így a gyakorlatban csak a másik, úgynevezett *laza összekapcsolás*t alkalmazzák, ahol az alrendszerek külön rendszerként viselkednek, egy szinkronizált, adatkapcsolatokat is megvalósító *interfész* által összekapcsoltan.

Adatkapcsolatok A szakértői és az irányító alrendszerek adatkapcsolatai mindkét irányban fontosak a rendeltetésszerű működéshez.

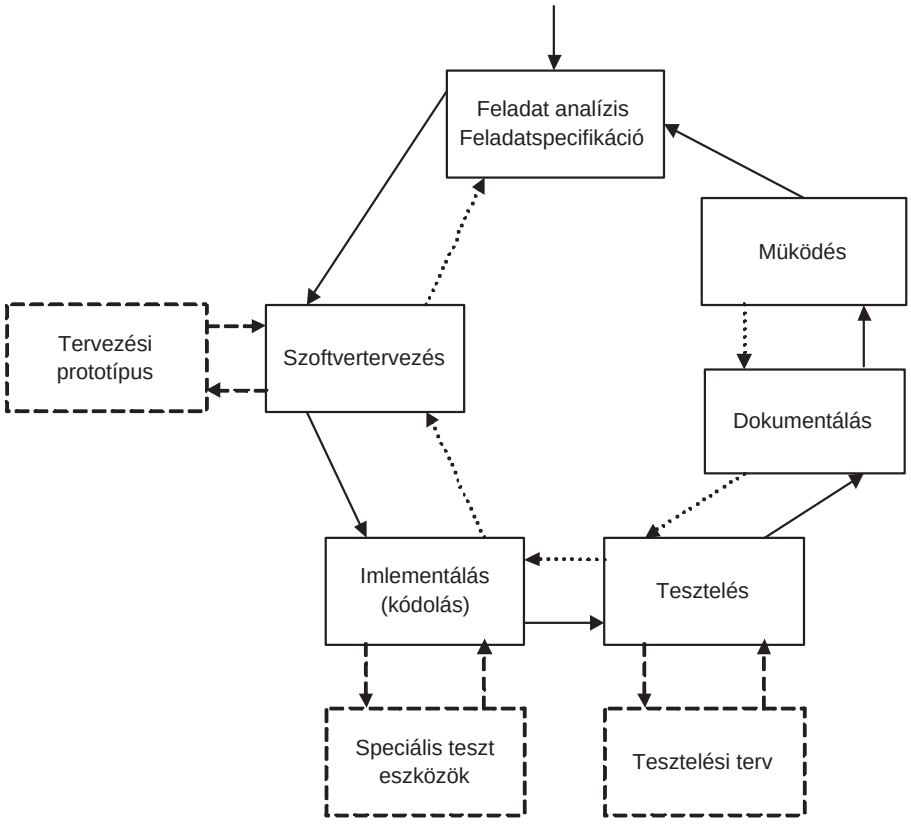
- Az irányító alrendszerből a szakértői alrendszerbe az alábbi adatok kerülnek átadásra:
  - a *ténybázis gyökérpredikátumainak (lásd 3.1.3. fejezet) értéke*, ez egy összefüggő nagyobb adathalmaz, amelyet a konzisztencia biztosítása érdekében tükörmásolással, a szinkronizáció biztosítása céljából pedig órára vezérelten hajtunk végre (lásd szinkronizációs kapcsolatok alább),
  - *események* (tipikusan jelváltozások).
- A szakértői alrendszerből az irányító alrendszerbe az alábbi adatokat kell eljuttatni:
  - a következtetés eredményét (pl. az operátori kérdésre adott választ) események formájában, diagnosztikai jellegű adatokat,
  - a rendszeres időközönkénti vagy eseti diagnosztikai feldolgozások eredményeit események vagy származtatott adatok formájában,
  - beavatkozójel változást (néha) esemény formájában.

Szinkronizációs kapcsolatok A szinkronizációs kapcsolat többnyire az irányító alrendszerből érkező kezdeményezés hatására valósul meg az alábbi esetekben:

1. kezelői kérésre (a kezelő valamire kíváncsi a következtető géptől),
2. az eseménykezelő kérésére (eseményre indított feldolgozás, tipikusan diagnosztikai kérések),
3. órára vezérelt, ciklikus feldolgozások (előrebecslés).

2.2.3. A folyamatirányító szakértői rendszerek tervezése

A folyamatirányító szakértői rendszer tervezése és megvalósítása jól meghatározott szakaszokra bontható az intelligens szoftver rendszerek 2.4 ábrán bemutatott szoftver életciklusa mentén. Az ábrán szaggatott vonalakkal határolt téglalapok formájában tüntettük fel azokat a kiegészítő tevékenységeket illetve elemeket, amelyekre a folyamatirányító szakértői rendszerek tervezése és megvalósítása során kiemelt figyelmet kell fordítani. Ezek az alábbiak:



2.4. ábra. Intelligens szoftverrendszerek életciklusa

1. *Tervezési prototípus* (design prototype)
   A tervezési prototípusokat elterjedten alkalmazzák a tudásbázisú rendszerek megvalósításakor arra, hogy a kidolgozott heurisztikus algoritmusok működését és tulajdonságait „kísérletileg” állapítsák meg, illetve hangolják. Egy tesztelési prototípus azonban általában csak egy kisméretű tudásbázis felett csökkentett funkciókat (a kritikusnak gondoltakat) valósít meg. Emiatt fennáll annak a veszélye, hogy a feladatok algoritmikus bonyolultsága miatt a valódi rendszer a reális méretű tudáshalmazon teljesen másképp viselkedhet.

2. *Speciális teszteszközök* (special test tools)
   A speciális teszteszközök az egyes funkciók megvalósítása során funkció-szintű, azaz lokális tesztelést tesznek lehetővé. Itt is fennáll azonban annak a veszélye, hogy

a lokálisan tesztelt összetett rendszer nem megfelelően működik, hiszen a funkciók a tudásbázis elemeinek erős összekapcsoltsága miatt erősen összefüggőek.

3. *Tesztelési terv* (test plan)
   Mivel egy intelligens szoftver rendszer teljes (kimerítő) tesztelése NP-nehéz feladat, a tesztelési tervben csak részleges teszteket lehet előirányozni. Itt kényes feladat az, hogy a tesztesetekkel lefedjük a tesztelendő folyamatirányító szakértői rendszer összes funkcióját az előrelátható, a gyakorlat szempontjából legfontosabb esetekben. Ezek meghatározása komoly fejlesztői és üzemeltetési tapasztalat egyidejű jelenlétét követeli meg.

3. fejezet

Tudásábrázolás és következtetés

Az intelligens irányító rendszerek segítségével megvalósított bonyolult problémák megoldásához egyrészt nagy mennyiségű ismeret (tudás), másrészt különböző megoldási módszerek, mechanizmusok szükségesek. Ebben a fejezetben a leggyakrabban alkalmazott tudásreprezentációs módszereket, valamint a szakértő rendszerek legfontosabb problémamegoldó mechanizmusát (szabály alapú következtetés) mutatjuk be.

3.1. Adat és tudás, adat- és tudásábrázolás

Egy intelligens szoftver rendszer „passzív” (futtatható) része a tudásbázis, amelynek szerepe a szoftver rendszerek adat(bázis) részéhez hasonló. A különbség a két rendszer komplexitásából adódik: amíg egy adatbázis sok adatot és viszonylag kevés összefüggést, egy tudásbázis több/kevesebb adatot és sok összefüggést tartalmaz. Ennek megfelelően intelligens rendszerek használata során sok információ (adat és tudás) megadására van szükség, amely a világ objektumainak tulajdonságait tartalmazó adatok illetve tények, valamint az adatok közötti relációkat tartalmazó összefüggések formájában írható le. A feladat megoldása olyan speciális módszerekkel és algoritmusokkal történik, amelyek a megadott információ felhasználásával a probléma megoldását megkeresik.

3.1.1. Adatábrázolás közönséges adatbázisokban

Közönséges adatbázisokban az adatábrázolás rögzített szerkezetű, összetartozó adatcsoportokat megjelenítő úgynevezett *rekord*ok formájában történik, ahol az adatelemeket a rekordok fix típusú mezői tartalmazzák.

Példa egy rekordra:

```
nyers_mért_adat record
    azonosító:      string;
    típus:          character;              {'R', 'B'}
    érték:          real if típus = 'R'
                    boolean if típus = 'B'; {típusfüggő!}
    mérés ideje:    integer array[6];       {mp perc óra nap hó év}
```

```
    hibakód:        string;
end; {nyers_mért_adat}
```

Az azonos szerkezetű rekordok rendezett halmaza alkotja a fájlt, ezek csoportja pedig az adatbázist. Ez a teljesen passzív adatszerkezettel rendelkező adatbázis lehetséges tudásreprezentációs módszerként csak programozással kiegé-szítve használható, mert merev adatszerkezete az összefüggések leírásához meglehetősen korlátozott lehetőségeket biztosít.

3.1.2. Adatábrázolás relációs adatbázisokban

A *relációs adatbázis*okban az összetartozó adatcsoportokat szintén rekordok formájában tárolják, azonban a rekord adategységek logikai csoportjaként jelenik meg, ahol relációk formájában mezők és mező csoportok közötti összefüggések adhatók meg. Ezek a relációk logikai és/vagy aritmetikai típusúak lehetnek, amelyek mezők default vagy megengedett értékeinek definiálására, valamint azonos vagy különböző rekordokban levő mezők értékeinek definiálására használhatók.

Példa relációs adatbázis egy rekordjára:

add_rekord record {$a + b = c$ művelet és eredményeinek tárolása}

```
    a:    real;      {op_1}
    b:    real;      {op_2}
    c:    real;      {eredmény}
end; {add_rekord}
```

reláció: $a + b = c$

Példa relációs adatbázis több rekordja közötti összefüggés definiálására:

```
fájl_1:
mért_adat record
    azonosító:    string;
    érték:        real;
    ...
end; {mért_adat record}
```

```
fájl_2:
nyers_mért_adat record
    azonosító:    string;
    érték:        long_integer;
    ...
end; {nyers_mért_adat record}
```

reláció:

ha mért_adat.azonosító = nyers_mért_adat.azonosító
akkor mért_adat.érték := conv(nyers_mért_adat.érték)

A relációs rekordok halmaza és az ezek közötti relációk együttesen relációs fájlt alkotnak, a relációs fájlok halmazából és az összekapcsoló relációkból pedig relációs adatbázis készíthető. A relációs adatbázisokban így már megjelenik az összefüggések leírására alkalmas aktív elem relációk formájában, amely elvileg alkalmassá tenné tudásbázis megvalósítására, azonban probléma, hogy adatszerkezete még mindig merev.

3.1.3. Tudásábrázolás: szabályok

Mesterséges intelligencia eszközökben és szakértői rendszerekben legelterjedtebb tudásreprezentációs forma a *szabály*okkal történő reprezentáció, amely hatékony eszközt biztosít a heurisztikus ismeretek leírásához. Egy szabály speciális szintaxissal rendelkező logikai kifejezés, amely a logika eszközeivel leírható és kezelhető. Ezért a szabályok tárgyalása előtt röviden áttekintjük a logika ehhez kapcsolódó alapvető fogalmait.

A logika alapfogalmai

A logika alapvető építőelemei az úgynevezett atomok (más néven atomi formulák), amelyek ítéletkonstansok (más néven logikai konstansok), ítéletváltozók (más néven logikai változók) valamint predikátumok lehetnek. Két logikai konstanst különböztetünk meg az igaz és a hamis logikai érték leírására, amelyeket true (T) és false (F) szimbólumokkal jelölünk. Az atomok egyben formulák is. Rekurzív módon a logikai műveleti jelek – **negáció** $(\neg)$, **diszjunkció (vagy)** $(\vee)$, **konjunkció (és)** $(\wedge)$, **implikáció** $(\rightarrow)$, **azonosság** $(\Leftrightarrow)$ – alkalmazásával formulákból újabb formulákat készíthetünk a következőképpen: ha A és B formulák, akkor a $(\neg A), (A \vee B), (A \wedge B), (A \rightarrow B), (A \Leftrightarrow B)$ kifejezések is formulák.

A	B	$\neg A$	$A \vee B$	$A \wedge B$	$A \rightarrow B$	$A \Leftrightarrow B$
T	T	F	T	T	T	T
T	F	F	T	F	F	F
F	T	T	T	F	T	F
F	F	T	F	F	T	T

3.1. táblázat. A műveleti jelek igazságtáblája

A formuláknak igazságértékük ad jelentést a szemantika szabályai szerint, amely során a formulában szereplő ítéletváltozókhoz értéket rendelve a formulát a műveleti jelek szemantikája alapján kiértékeljük. A műveleti jelek szemantikája igazságtáblákban foglalható össze (3.1. táblázat).

A logika kiterjesztéseképpen bevezethetjük az ismeretlen vagy bizonytalan értéket (jelölése: unknown, U), amely tulajdonképpen a „true $\vee$ false” logikai értékét definiálja. Az újabb ítéletkonstans bevezetésével a műveletek szemantikájának kiterjesztésére is szükség van. A „$\vee$” és „$\rightarrow$” műveletek kiterjesztett igazságtábláit a 3.2. táblázat mutatja.

$A \vee B$	T	F	U
T	T	T	T
F	T	F	U
U	T	U	U

$A \rightarrow B$	T	F	U
T	T	F	U
F	T	T	T
U	T	U	U

3.2. táblázat. A $\vee$ és $\rightarrow$ műveletek kiterjesztett igazságtáblái

A logikai műveletek algebrai tulajdonságai alapján egy formula átalakítható vele ekvivalens formulává. Megkülönböztetünk speciális szintaxissal rendelkező formulákat, az úgynevezett *kanonikus alak*okat vagy, *normálforma*kat, amelyek a következők lehetnek:

- *Diszjunktív normálforma* vagy *DNF*: atomi formulák vagy negált atomi formulák konjunkciójának diszjunkciója

  például

  $$(\neg a \wedge b) \vee (c \wedge \neg d)$$

- *Konjunktív normálforma* vagy *CNF*: atomi formulák vagy negált atomi formulák diszjunkciójának konjunkciója

  például

  $$(\neg a \vee b) \wedge (c \vee \neg d)$$

- *Implikációs normálforma* vagy *INF*: implikációs formula, amelynek implikációs előtagjában atomi formulák konjunkciója, utótagjában atomi formulák diszjunkciója szerepel

  például

  $$(a \wedge b) \rightarrow (c \vee d)$$

Szabályok szintaxisa és szemantikája

Egy szabály egy „ha . . . akkor . . . ” szerkezetű mondatként megfogalmazható feltételes állítás:

ha *feltétel* **akkor** *következmény*;

amely szintaxisa szerint a következő elemekből áll:

1. *Predikátum*ok: elemi logikai kifejezések, amelyek **true**, **false** vagy **unknown** értéket vehetnek fel. A predikátumok definiálhatnak aritmetikai relációkat ($=$, $\neq$, $\leq$, $>$, $<$) és tartalmazhatnak kvalitatív vagy szimbolikus állandókat (például **alacsony**, **nagyon magas**, **nyitva**, stb).

   Néhány példa intelligens irányító rendszerekben előforduló predikátumokra:

   $$p_1 = (k = \mathbf{on}) \;\; ; \;\; p_2 = (T < 300) \;\; ; \;\; p_3 = (h = \mathbf{magas})$$

   $$p_4 = (\textit{hiba} = \textit{„tank overflow”})$$

   ahol p_1, p_2 és p_3 aritmetikai predikátumok, az ezekben szereplő változók T hőmérséklet, k on–off állapotú kapcsoló és h szint pedig mért jelek, amelyek időfüggőek. Ha például a hőmérséklet (T) értéke egy adott időpillanatban $350\ ^\circ K$, akkor a p_2 predikátum **false** értéket kap.

   Fontos megjegyezni, hogy mivel a predikátumok értékei időben változó mért jelek értékeitől függnek, ezért ezek logikai értékei önmagukban szintén jelek lesznek.

2. *Logikai kifejezés*ek, amelyek tartalmazhatnak

- atomi formulákat, amelyek predikátumok, logikai változók vagy logikai konstansok
- logikai műveleti jeleket ($\neg$, $\wedge$, $\vee$, $\rightarrow$, $\Leftrightarrow$)

a logikai szintaxisának szabályai szerint.

3. *Szabály*ok: Egy szabály maga egy logikai kifejezés, amely speciális szintaxissal rendelkező, a következő alakban megadható implikációs formula:

$$\textit{feltétel} \rightarrow \textit{következmény}$$

ahol a szabály *feltétel* és *következmény* része logikai kifejezésekből épül fel.

A *szabályok szemantikáját* az implikáció igazságtáblája (lásd 3.1. táblázat) definiálja. A szabályok használata általában a feltétel ellenőrzésével indul, ahol megvizsgáljuk, hogy ez igaz-e. Amennyiben igaz, a szabály alkalmazható (tüzel), az alkalmazás pedig a következmény rész teljesítéséből (igazzá tételéből) áll. A szabály használati módját befolyásolja a következtetés célja is, ezt részletesen a 3.2 fejezetben mutatjuk be.

Példa: egy egyszerű szabályhalmaz:
Predikátumok:

$$P = \{p_1, p_2, p_3, p_4\}$$

Szabályok:

$$\begin{array}{llll} \textbf{ha} & (p_1 \textbf{ és } p_2) & \textbf{akkor} & p_3; \\ \textbf{ha} & (p_3 \textbf{ és } p_4) & \textbf{akkor} & p_1; \end{array}$$

A szabályokkal ekvivalens logikai formulák:

$$\begin{array}{lll} (p_1 \wedge p_2) & \rightarrow & p_3; \\ (p_3 \wedge p_4) & \rightarrow & p_1; \end{array}$$

Datalog szabályhalmaz

A *datalog szabályhalmaz* a szabályhalmazok egy speciális struktúrájú alakja, amelynek szabályai a következő tulajdonságokkal rendelkeznek:

D1: *A szabályok predikátumainak argumentumai nem tartalmaznak függvény-szimbólumot.*

D2: *A predikátumokra nem alkalmazható a negáció, a szabályok alakja a következő:*

$$(p_{i_1} \wedge \cdots \wedge p_{i_n}) \rightarrow q_i;$$

ahol $p_{i_1}, \ldots, p_{i_n}$ és q_i predikátumok.

D3: *A szabályok „biztos szabályok” (angolul „safe rules”), amely azt jelenti, hogy értékük véges számú lépésben meghatározható.*

Az intelligens irányító rendszerek szabályai majdnem mindig datalog formájúak. Amennyiben nem, akkor könnyen azzá alakíthatóak a következő lépések és feltételezések alkalmazásával:

F1: *Függvényszimbólumok eltávolítása* a D1. követelmény biztosítására.
Ez függvények (pl. *sin*, *exp*) Taylor sorba fejtésével, vagy új változók bevezetésével (pl. *log T < 0.3* helyett *Tl := log T*, a szabályban *Tl* használata) biztosítható.

F2: *Negáció (¬) és diszjunkció (∨) műveletek eltávolítása* a D2. követelmény biztosítására
Az intelligens irányító rendszerekben szereplő predikátumok leggyakrabban aritmetikai predikátumok, amelyek negált alakját például a következő átalakítással küszöbölhetjük ki:

$$\neg(a > b)\ helyett\ (a \leq b)\ \ ,\ \ \neg(a = b)\ helyett\ (a \neq b)$$

Ezen túlmenően a negáció kiküszöbölhető a formula implikációs normálformára történő alakításával, amelyben a szabály feltételi része csak konjunkció(ka)t ($\wedge$ művelet), a következmény része pedig csak diszjunkció(ka)t ($\vee$ művelet) tartalmaz. A következmény részben szereplő diszjunkció(k) a szabályok többszörözésével távolítható(k) el a következő módon:

$$(s_i)\colon (p_{i_1} \wedge \cdots \wedge p_{i_n}) \rightarrow (q_{i_1} \vee \cdots \vee q_{i_m});$$

átalakításával

$$\begin{aligned}(s_{i_1})&\colon (p_{i_1} \wedge \cdots \wedge p_{i_n}) \rightarrow q_{i_1};\\ &\vdots\\ (s_{i_m})&\colon (p_{i_1} \wedge \cdots \wedge p_{i_n}) \rightarrow q_{i_m};\end{aligned}$$

F3: *A számítógéppel irányított rendszerekben alkalmazott véges digitális számábrázolás* a D3. követelmény teljesülését biztosítja.

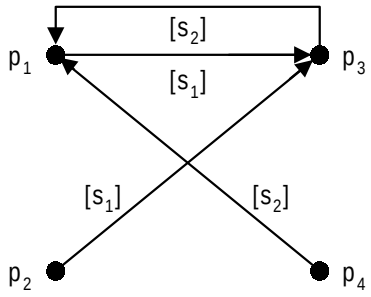
A datalog szabályrendszerek analízisük és végrehajtásuk (következtetés) szempontjából fontos tulajdonságokkal rendelkeznek, szerkezetük leírható úgynevezett *függőségi gráf*fal. Egy datalog szabályrendszer függőségi gráfja $D = (V_D, E_D)$ egy irányított gráf, amelynek felépítése a következő:

1. Csúcshalmaza a szabályhalmaz predikátumainak halmaza, azaz

$$V_D = P$$

2. Élhalmaza a predikátumok közötti kapcsolatokat definiálja, azaz két csúcs p_i és p_j között irányított él $(p_i, p_j) \in E_D$ van, amennyiben van olyan szabály, amelynek feltételi része p_i-t, következmény része pedig p_j-t tartalmazza.

3. Az élek (p_i, p_j) címkézettek, amely az élhez tartozó szabály azonosítóját jelöli.



3.1. ábra. Szabályhalmaz függőségi gráfja

Példa: a 3.1.3 szabályhalmaz függőségi gráfja a 3.1. ábrán látható, amelyen a (p_1, p_3) csúcsok között kör figyelhető meg.

A függőségi gráf információt szolgáltat a predikátumok egymástól való függéséről. A gráf „belépési" pontjainak (a csúcs befoka 0) megfeleltethető predikátumokat *gyökérpredikátum*oknak nevezzük (a példa gyökérpredikátumai p_2, p_4), amelyeknek a szabályhalmaz működtetése előtt értéket kell adni. A gráf irányított körei a végrehajtási sorrendtől való függést mutatják: körmentes gráf esetén a következtetés sorrendjétől függetlenül ugyanazt az eredményt kapjuk, kört tartalmazó gráfnál az eredmény függhet a végrehajtás sorrendjétől.

3.1.4. További tudásábrázolási technikák

Objektumok

Az objektum-orientált programozási nyelveket széles körben használják „hagyományos" szoftver rendszerekben. A következőkben röviden összefoglaljuk ezek legfontosabb jellemzőit, amelyek lehetővé teszik intelligens szoftver rendszerekben történő alkalmazásukat is.

Egy objektum-orientált rendszerben a figyelem középpontjában álló dolgok, egységek az *objektum*ok, amelyeket szigorú fa struktúra szerinti hierarchiába rendezett objektum-osztályok segítségével írunk le. Az adott osztályra jellemző közös tulajdonságokat attribútumok, az objektumokon végzendő műveleteket metódusok formájában definiáljuk, és mindezeket egy egységként kezeljük. Ez az úgynevezett *egységbe zárás* (angolul *encapsulation*) az objektum-orientált rendszerek egyik legfőbb jellemzője. Fontos szerepe van emellett az osztályhierarchiának megfelelő *öröklődés*nek, amely az objektumok (beleértve az attribútumaikat és metódusaikat) származtatását teszi lehetővé. Az osztályokból konkrét objektum példányokat hozhatunk létre megfelelő paraméterezéssel, ezek a példányok létrehozásuk után önálló életet élnek.

Példa egy egyszerű osztályhierarchiára:

```
{parent class   }   class tube
{p-attributes   }   val:                    valve;
{p-procedure    }   procedure               open-valve   (error-code);
                    ...                     {statements to open}
```

```
                    end                     {open-valve}
{p-class body   }   ...                     {statements to initialize}
                    end;                    {tube}

{sub-class      }   tube class meas-tube
{s-attributes   }   T,v:                    measurement-device;
{s-procedure    }   procedure               measure    (value);
                    ...                     {statements to get the value}
                    end                     {measure}
{s-class body   }   ...                     {statements to initialize}
                    end;                    {meas-tube}
```

Keretek

A *keret*ek a 3.1.1. fejezetben bemutatott rekordok kiterjesztéseként foghatók fel, amelynél már megjelennek az összefüggések leírására használható standard aktív elemek. Egy keretnek, mint tudáselemnek az alábbi részei vannak:

- **rések** (slots): a rekordok mezőinek felelnek meg, azonban a típusdeklaráció flexibilisebb és a típus működés során változhat
- **démonok** (daemons): beépített eljárások, amelyek rések értékváltozásaihoz rendeltek

  szokásos démonok: if-added, if-removed, if-needed, if-changed

A keretek felépítése és használata az objektumokhoz hasonló, így jellemző rájuk a hierarchikus struktúra, az öröklődés és a példányosítás. Fő különbség az eljárások számában és szerepében mutatkozik meg: a keretek eljárásai, a démonok kötött készletből választhatók, működésük megváltoztathatja bármely keret rés-értékét, amely a démonok működésének továbbgyűrűzését okozhatja. Egy keret-alapú rendszer működése indirekt módon írható le, melyet a keret példányokban levő démonok határoznak meg. Ennek megfelelően egy keret-alapú tudásreprezentáció rugalmas, azonban nehezen áttekinthető és ellenőrizhető, ezért óvatosan kell használni.

Szemantikus hálók

A *szemantikus háló* egy grafikus eszköz a tudásbázisban levő tudáselemek közötti szemantikai összefüggések leírására. A szemantikus háló irányított gráffal reprezentálja a tudásbázis szerkezetét, amelyben a csúcsok az objektumoknak és attribútumok értékeinek, a címkével ellátott élek pedig a csúcsok közötti összefüggéseknek, relációknak felelnek meg. A relációk többsége előre definiált kategóriák közül vesz fel értéket. A legáltalánosabb relációk a következők:

- **is_a**: alosztály – osztály közötti kapcsolat leírására, pl. class_A is_a class_B
- **instance_of**: példány – osztály közötti kapcsolat leírása, pl. object_A instance_of class_A

- **part_of**: attribútum – osztály közötti kapcsolat leírása, pl. attribute_A part_of class_A

A szemantikus hálók relációi bináris predikátumokkal is leírhatók, pl. class_A is_a class_B szintaktikailag különböző, de vele ekvivalens formája az is_a(class_A, class_B) predikátum.

A szemantikus háló tulajdonképpen *meta-tudás* (tudásbázis elemeiről szóló tudás), amely a tudásbázis szerkezetét mutatja meg. Általában más tudásreprezentációs módszerrel együtt használható (pl. objektumok, keretek) főképpen a tudásbázis verifikálására, validálására és diagnosztikai célokra.

3.2. Következtetés és keresés

Egy szabály alapú szakértői rendszer tudásbázisa a következő két részből áll:

- *Tények* vagy *predikátumok*, amelyek az adott probléma elemeit reprezentálják deklaratív ismeretek formájában. Értékük igaz (true) vagy hamis (false) (kiterjesztett logika esetén ismeretlen(unknown)) lehet, amely a következtetés során (valós idejű rendszerek esetében időben is) változhat.

- *Összefüggések* vagy *szabályok*, amelyek heurisztikus ismereteket, „ökölszabályokat" reprezentálnak tipikusan szituáció-akció formájában megadva. Ez a tárgyköri tudásunk általánosan érvényes része, amelyet a tudásmérnök módosíthat a tudásbázis karbantartása során. A szabályokat a következtető gép működteti, amelynek hatására a ténybázis változhat.

A tudásbázis állapotát a predikátumainak értékét tartalmazó állapotvektor segítségével adhatjuk meg.

$$\underline{a} = \begin{bmatrix} p_1 \\ \vdots \\ p_{n_P} \end{bmatrix}$$

ahol

$p_i = \{t(true),\ f(false),\ u(unknown)\}$ és

n_P a predikátumok száma.

A szabályokat a következtető gép működteti új ismeret vagy információ levezetése céljából. Egy elemi következtetési lépés egy szabály alkalmazását jelenti, amely általában a következő részlépésekből áll:

- *Mintaillesztés*

  Egy szabály alkalmazható (más néven tüzelőképes), ha a feltételi része igaz (datalog szabály esetén a feltételi részében levő predikátumok igazak). Ebben a részlépésben a következtető gép megkeresi a tüzelőképes szabályokat a szabályok feltételi része és a tények közötti illesztéssel, majd a végrehajtható szabályokat egy úgynevezett konfliktushalmazba teszi.

- *Szabály kiválasztása*

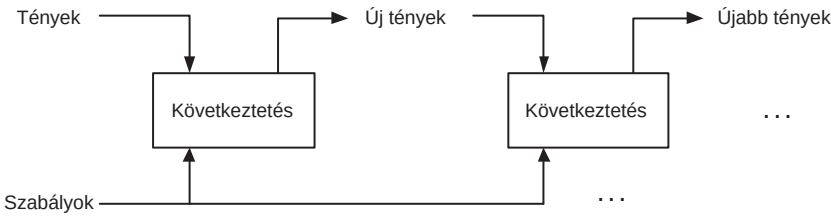
  A következtető gép beépített vezérlési stratégiája alapján a konfliktushalmazban levő szabályok közül választ egyet.

- *Szabály végrehajtása*

  A következtető gép a ténybázis predikátumait módosítja a kiválasztott szabály következmény részében szereplő akciók végrehajtásával, a következmény rész predikátumainak igazzá tételével. Ezt a lépést más néven a szabály tüzelésének nevezzük.

- *Terminálási feltétel bekövetkeztének figyelése*

A következtető gép ezt az elemi következtetési lépést ciklikusan ismétli a terminálási feltétel bekövetkezéséig vagy amíg nincs több alkalmazható szabály (lásd 3.2. ábra)



3.2. ábra. A következtetés lépései

A következtetési lépések sorozata egy az alkalmazott szabályokból álló láncolatot alkot, amely az állapottér kezdeti és végállapota közötti irányított úttal reprezentálható. Ennek megfelelően a következtetés folyamata a tudásbázis állapotterében szabályok segítségével történő elmozdulással illusztrálható.

Példa Tekintsük a következő szabályhalmazt és kezdeti állapotot:

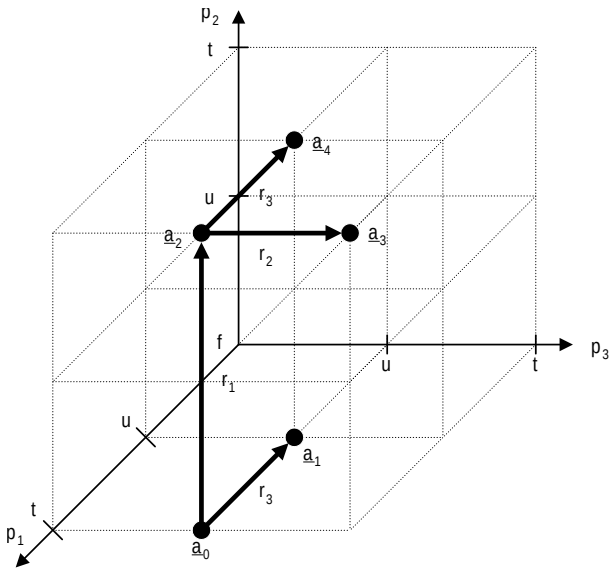
$$\begin{array}{lll} (r_1): & \textbf{ha } p_1 = t, & \textbf{akkor } p_2 = t \\ (r_2): & \textbf{ha } p_2 = t, & \textbf{akkor } p_3 = t \\ (r_3): & \textbf{ha } p_3 = u, & \textbf{akkor } p_1 = u \end{array}$$

$$\underline{a}_0 = \begin{bmatrix} t \\ f \\ u \end{bmatrix}$$

ahol t jelentése **true**, f **false** és u **unknown**.

Kezdeti állapotban két alkalmazható szabály (r_1 és r_3) van. Az r_3 szabállyal az $\underline{a}_1$ állapotba (amely terminális állapot, azaz nincs több alkalmazható szabály) jutunk. Az r_1 szabállyal elérhető $\underline{a}_2$ állapotra ismét két szabályt alkalmazhatunk (r_2 és r_3), amelyekkel az $\underline{a}_3$ illetve $\underline{a}_4$ terminális állapotok érhetők el.

Az állapottérben történő következtetés a 3.3. ábrán látható.



3.3. ábra. Következtetés az állapottérben

A következtetési lépések sorozata *gráfbejárás*nak felel meg a kezdeti állapot és egy vagy több lehetséges, elfogadható vagy optimális célállapot között. Ennek megfelelően a következtetés feladata *keresési feladat*ként fogalmazható meg az állapottérben (keresési térben), ahol a lehetséges akcióknak a szabályok feleltethetők meg.

Egy következtetési lépésben általában több alkalmazható szabály van (több szabály illeszkedik a ténybázisra), a következtetés eredménye pedig általában függ az alkalmazott szabálytól. Ezt a helyzetet *konfliktus*nak nevezzük, amely a keresési térben *elágazás* formájában jelenik meg, ahol a tüzelőképes szabályok száma azonos az elágazások számával. Az alkalmazni kívánt szabály kiválasztásának folyamatát *konfliktusfeloldás*nak nevezzük.

A leggyakrabban alkalmazott konfliktusfeloldó stratégiák a következők:

- egy szabály véletlenszerű kiválasztása,
- az első alkalmazható szabály használata,
- szabályokhoz fontossági sorrend (prioritás) rendelése,
- heurisztikus módszerek alkalmazása.

A következtetés alapja a *modus ponens* nevű levezetési szabály, amelynek általános alakja a következő:

$$\frac{A \qquad A \rightarrow B}{B} \quad \textit{vagy} \quad A,\ A \rightarrow B \ \Rightarrow\ B$$

szavakkal:

Ha A igaz és A-ból következik B, akkor B is igaz.

A modus ponenst a következtető rendszerekben kétféleképpen használhatjuk, amely szerint *adatvezérelt* és *célvezérelt* következtetésről beszélhetünk.

3.2.1. Adatvezérelt következtetés

Adatvezérelt vagy *előrefelé haladó következtetés*nél feladat a tudásbázis kezdőállapotából (tényeiből) egy célállapot elérése vagy megkonstruálása. Az új következ-tetések előállítása a modus ponens alkalmazásával történik a terminálási feltétel eléréséig vagy az összes következmény előállításáig (nincs több alkalmazható szabály).

Az adatvezérelt következtetés alapvető algoritmikus problémaként is megfogalmazható a következőképpen:

ADATVEZÉRELT KÖVETKEZTETÉS DEFINIÁLT CÉLÁLLAPOTTAL

Adott:

- ténybázis kezdeti állapota ($\underline{a}_0$)
- szabálybázis
- ténybázis célállapota(i) ($\underline{a}_g$)

Kérdés:
$\underline{a}_g$ következménye $\underline{a}_0$-nak?
($\underline{a}_g$ levezethető $\underline{a}_0$-ból szabályok alkalmazásával?)

A fenti feladat egy döntési probléma, amelynek megoldása során legrosszabb esetben a teljes állapotteret (keresési teret) be kell járni. Mivel a fa mérete (a csúcsainak száma) exponenciálisan nő a számítási lépések számában, a feladat NP-teljes.

A fenti probléma változata az adatvezérelt következtetés előre definiált célállapot nélkül.

ADATVEZÉRELT KÖVETKEZTETÉS

Adott:

- ténybázis kezdeti állapota ($\underline{a}_0$)
- szabálybázis

Kiszámítandó:
a kezdeti állapot összes következménye.

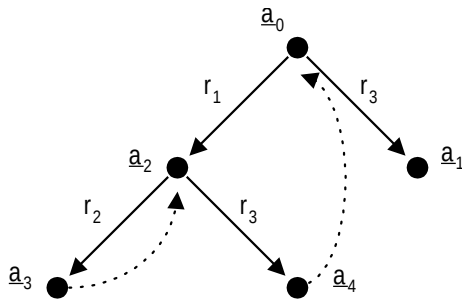
Ez a probléma egy keresési feladat, amely a már a feladatkitűzésből adódóan NP-teljes.

Adatvezérelt következtetés során a kezdeti állapotból ($\underline{a}_0$) kiindulva egy *kereső gráf*ot építünk fel az állapottérben. A gráf bejárása során a szabályok feltételi részét illesztjük a ténybázishoz, és az alkalmazható szabályok közül egyet végrehajtva annak következmény részében meghatározott módon tényeket igazzá illetve hamissá teszünk (azaz a ténybázishoz tényeket hozzáadunk, illetve törlünk). A szabály alkalmazásával egy új állapotba jutunk el.

Ha ez az állapot az ADATVEZÉRELT KÖVETKEZTETÉS DEFINIÁLT CÉLÁLLAPOTTAL feladat egyik célállapota, akkor az algoritmus terminál.

Amennyiben a célállapotot nem sikerült elérni és nincs több alkalmazható szabály, az algoritmus visszaléphet egy olyan állapotba, ahol több alkalmazható szabály volt újabb lehetőség kipróbálása céljából. Ezt a visszalépési folyamatot más néven *backtrack*-nek nevezzük.

Az 3.3. ábrán látható következtetési feladat backtrack mechanizmusát mutatja a 3.4. ábra.



3.4. ábra. Az 3.3. ábra példájának backtrack mechanizmusa

Fontos megjegyezni, hogy backtrack alkalmazása esetén az elágazási pontokhoz tartozó tudásbázis állapotokat tárolni kell a további következtetések konzisztenciájának biztosítására. Ez meglehetősen drága, emiatt adatvezérelt következtetés esetén visszalépést nem, vagy csak nagyon ritkán alkalmaznak.

Adatvezérelt következtetés: egy egyszerű esettanulmány

Legyen a ténybázis kezdeti állapota a következő:

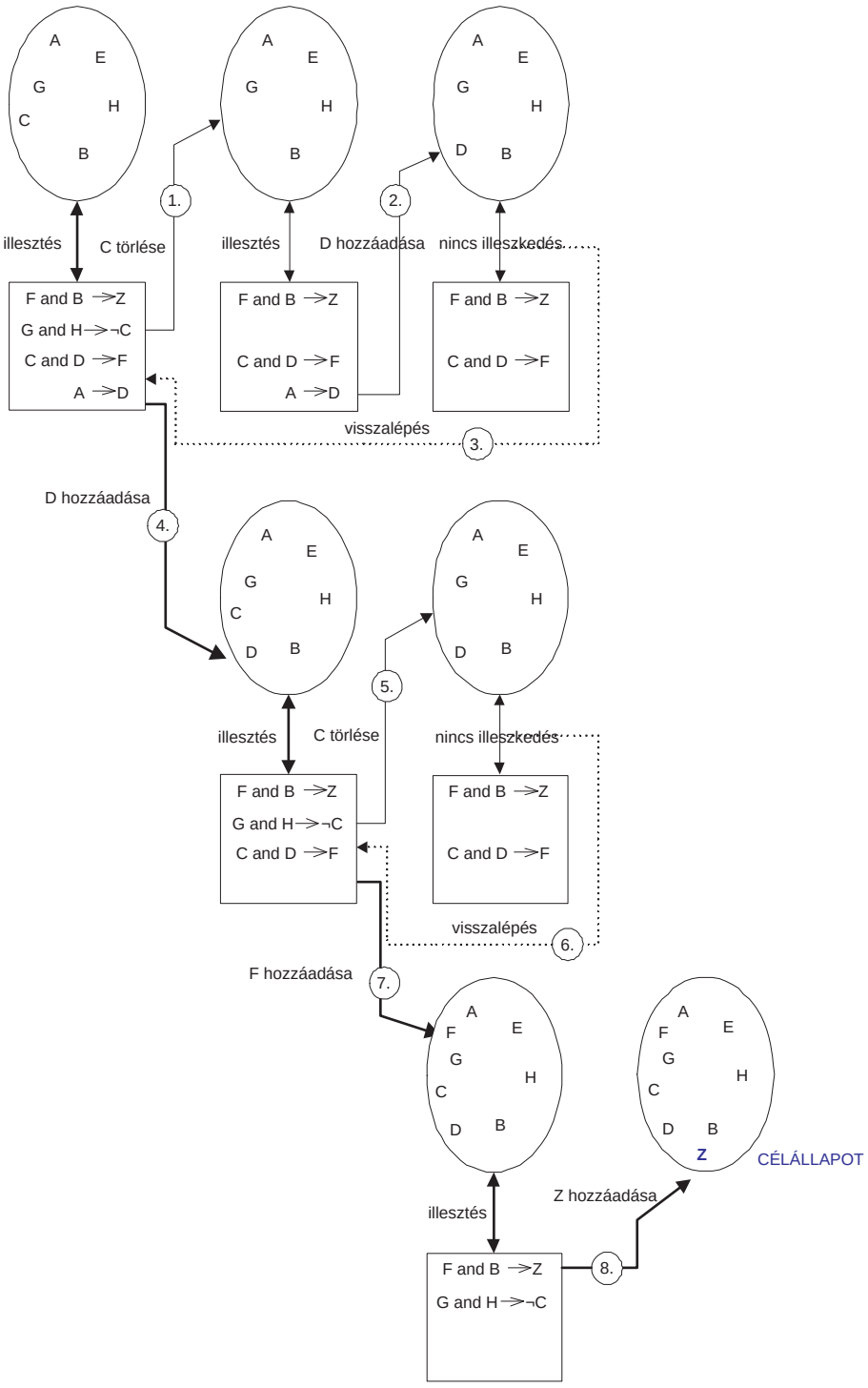
$$\underline{a}_0 = \begin{bmatrix} A = t \\ B = t \\ C = t \\ D = f \\ E = t \\ F = f \\ G = t \\ H = t \\ Z = f \end{bmatrix}$$

Tekintsük az alábbi egyszerű szabályhalmazt:

(r_1): $F \wedge B \rightarrow Z$
(r_2): $G \wedge H \rightarrow \neg C$
(r_3): $C \wedge D \rightarrow F$
(r_4): $A \rightarrow D$

Az $\underline{a}_g$ célállapotban legyen Z igaz (a többi predikátum értéke a célállapot szempontjából indifferens).

Kérdés:
Az $\underline{a}_g$ célállapot (amelyben Z igaz) elérhető-e $\underline{a}_0$ kezdeti állapotból szabályok alkalmazásával?

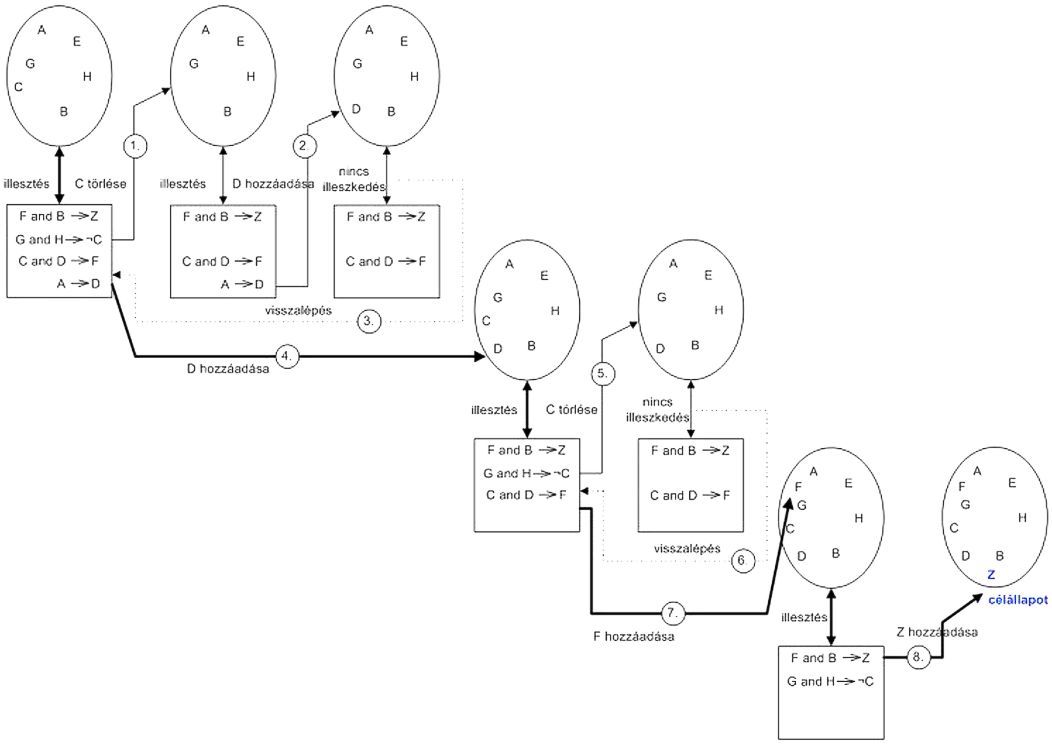


3.5. ábra. Adatvezérelt következtetés: egy egyszerű példa

Megoldás:
A megoldás során konfliktusfeloldó stratégiaként az első alkalmazható szabály végrehajtását használjuk, valamint feltételezzük, hogy egy szabályt csak akkor hajtunk végre, ha az módo-

sítja a ténybázis állapotát. Az előrefelé haladó következtetés lépései a 3.5. ábrán láthatók, amelyet részletesen az „elorefele.wmv” fájl mutat be. (A ténybázis aktuális állapotát az oválisok tartalmazzák, az adott állapotban tüzelőképes szabályokat pedig vastagítva jelenítettük meg.)

3.6. ábra. `elorefele.wmv`



3.2.2. Célvezérelt következtetés

Célvezérelt vagy *visszafelé haladó következtetés* során feladat egy feltételezett célállapot érvényességének igazolása kezdetben érvényes tényekre támaszkodva. Ebben az esetben „fordított irányban” használjuk a modus ponenst az új részcélok előállítására, s a következtetést az összes részcél igazolásáig vagy addig végezzük, amíg nincs több igazolható részcél (nincs több alkalmazható szabály).

A célvezérelt következtetés algoritmusa a következő:

CÉLVEZÉRELT KÖVETKEZTETÉS

Adott:

- célállapot: a ténybázis feltételezett állapota ($\underline{a}_g$)
- szabálybázis
- ténybázis kezdeti állapota ($\underline{a}_0$)

Kérdés:
$\underline{a}_g$ igazolható $\underline{a}_0$-ból?
($\underline{a}_g$ levezethető $\underline{a}_0$-ból szabályok segítségével?)

Ez egy döntési probléma, amelynek megoldásához legrosszabb esetben a teljes keresési teret be kell járni. A probléma az adatvezérelt következtetéshez hasonlóan NP-teljes.

Célvezérelt következtetés: egy egyszerű esettanulmány

Legyen a ténybázis kezdeti állapota a következő:

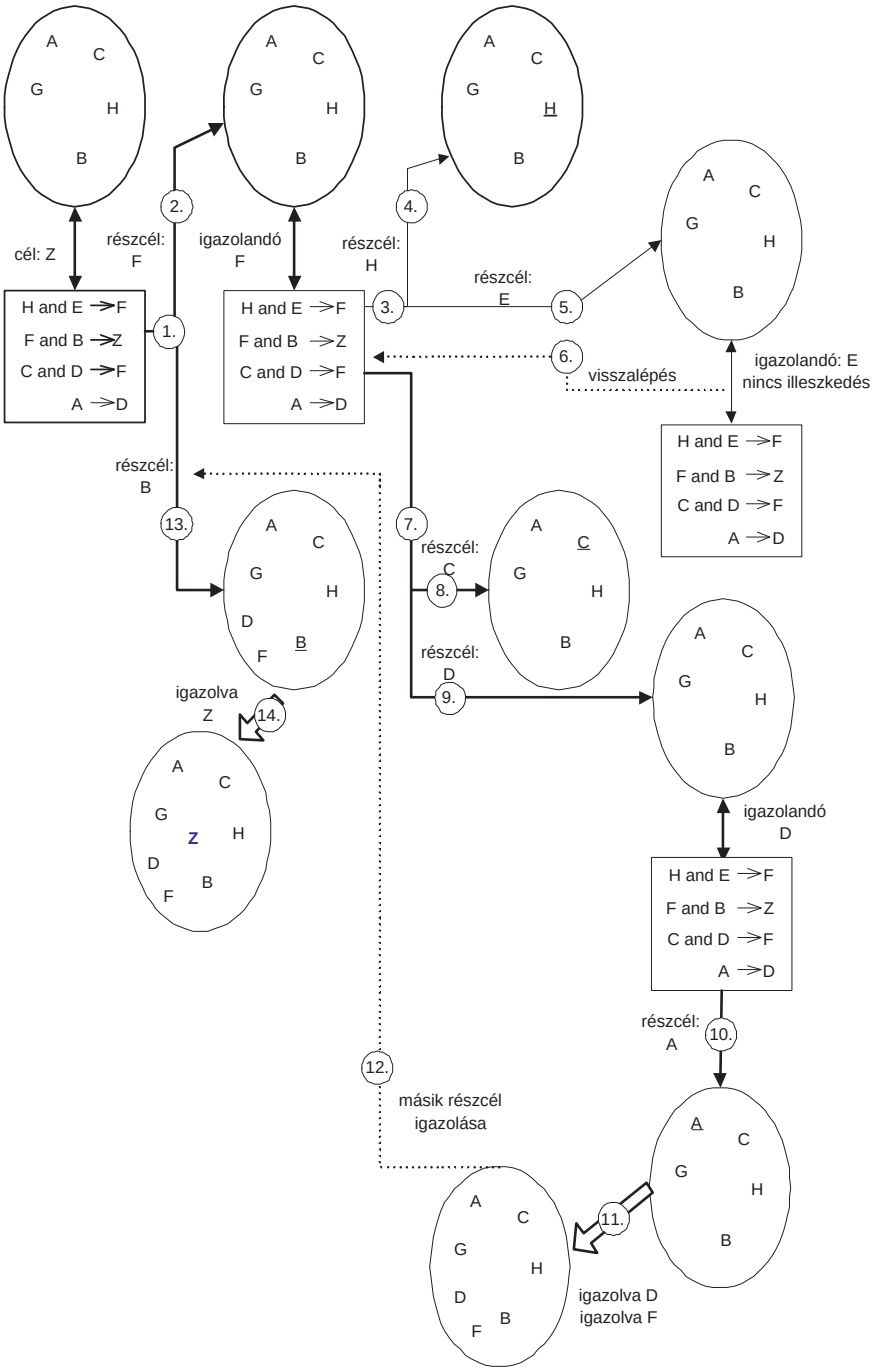
$$\underline{a}_0 = \begin{bmatrix} A = t \\ B = t \\ C = t \\ D = f \\ E = f \\ F = f \\ G = t \\ H = t \\ Z = f \end{bmatrix}$$

Tekintsük az alábbi egyszerű szabályhalmazt:

$(r_1) : H \wedge E \rightarrow F$
$(r_2) : F \wedge B \rightarrow Z$
$(r_3) : C \wedge D \rightarrow F$
$(r_4) : A \rightarrow D$

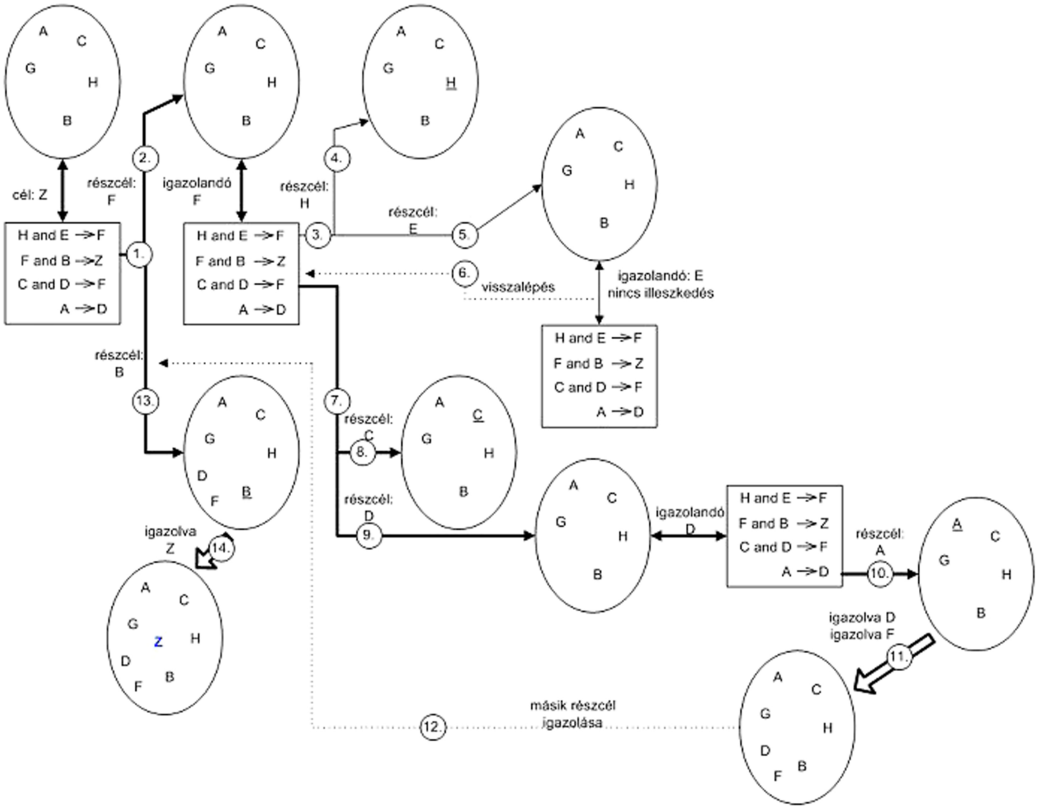
Legyen a célállapotban $Z =$ **true**.
Kérdés:
A célállapot $Z =$ **true** igazolható-e a kezdeti állapotra $\underline{a}_0$ támaszkodva a szabályok alapján?
Másszóval: célunk, hogy igazoljuk Z-t.

Megoldás:
A megoldás során konfliktusfeloldó stratégiaként az első alkalmazható szabály végrehajtását használjuk, valamint feltételezzük, hogy egy szabályt csak akkor hajtunk végre, ha az új részcél vizsgálatát idézi elő. A célvezérelt következtetés lépései a 3.7. ábrán láthatók, amelyet részletesen a „visszafele.wmv” fájl mutat be. (A ténybázis aktuális állapotát az oválisok tartalmazzák, az adott állapotban tüzelőképes szabályokat pedig vastagítva jelenítettük meg.)



3.7. ábra. Célvezérelt következtetés: egy egyszerű példa

3.8. ábra. `visszafele.wmv`



4. fejezet

Kvalitatív modellezés

A kvalitatív modellek a közönséges, mindennapi modellek olyan változatai, amelyekben a változók és paraméterek *kvalitatív (minőségi) érték*eket is felvehetnek. Ilyen kvalitatív érték lehet egy intervallum, amikor pl. egy paraméter értékéről csak annyit tudunk, hogy milyen határok közé esik, de lehet egy numerikusan nem meghatározott, szimbolikus érték is, például „*hideg*".

Az irányítási vagy diagnosztikai feladatok megoldására alkalmazható kvalitatív modellek a közönséges dinamikus modellekből származtathatóak, azok megoldását vagy csak szerkezetét jellemzik kvalitatív módon. Ebben a fejezetben megismerkedünk a kvalitatív dinamikus modellek felállításának és megoldásának alapvető fogalmaival és módszereivel. Alapozásképpen pedig röviden az előjel és intervallum aritmetikákkal kapcsolatos ismereteket is áttekintjük a fejezet elején.

4.1. Előjel és intervallum aritmetikák

A kvalitatív modellekben a közönséges, jól megszokott aritmetikai műveletek (pl. összeadás, szorzás) helyett a változók diszkrét értékkészlet halmazán, mint alaphalmazon értelmezett műveleteket használunk.

4.1.1. Diszkrét értékkészlet halmazok

Egy diszkrét értékkészletű változó lehetséges értékeinek halmazát **univerzum**-nak nevezzük. A lehetséges értékek halmazát a legáltalánosabb esetben egy intervallumokból álló halmaz segítségével írhatjuk le, amelyben lévő halmazoknak szimbolikus végpontjai is lehetnek.

A kvalitatív modellekben az alábbi univerzumok használatosak:

- *Általános kvalitatív* univerzum: valós intervallumok fix vagy szabad (szimbolikusan megadott) végpontokkal az alábbi formában

$$U_{\mathcal{I}} = \{[a_\ell, a_u] \mid a_\ell,\ a_u \in \mathcal{R},\ a_\ell \leq a_u\} \tag{4.1}$$

$a \oplus_S b$	$+$	0	$-$	$?$
$+$	$+$	$+$	**?**	$?$
0	$+$	0	$-$	$?$
$-$	**?**	$-$	$-$	$?$
$?$	$?$	$?$	$?$	$?$

4.1. táblázat. Az előjel összeadás műveleti táblája

Az univerzum halmazainak végpontjait az ún. **határpont halmaz**ban (angolul *landmark set*) gyűjthetjük össze:

$$L_{\mathcal{I}} = \{a_i \mid a_i \leq a_{i+1},\ i \in I \subseteq \mathcal{N}\} \tag{4.2}$$

- *Előjel* univerzum, amelyben az alábbi speciális fix végpontú intervallumokat és határpontjaikat definiáljuk

$$\begin{aligned} U_{\mathcal{S}} &= \{+, -, 0; ?\}, ? = + \cup 0 \cup - \\ L_{\mathcal{S}} &= \{a_1 = -\infty, a_2 = 0, a_3 = \infty\} \end{aligned} \tag{4.3}$$

  Az előjel univerzum atomi, vagy generátor halmazai az egymással diszjunkt pozitív $+$, negatív $-$ és nulla 0 halmazok, ezek uniójaként áll elő a *határozatlan előjel* $?$ halmaz.

- *Kiterjesztett logikai* univerzum, ahol

$$U_{\mathcal{L}} = \{\mathbf{true}, \mathbf{false}; \mathbf{unknown}\}$$

4.1.2. Előjel aritmetika

Az előjel aritmetika az (4.3) előjel univerzumon értelmezett összeadásra ($\oplus_S$) és szorzásra ($\otimes_S$) épül, amelyeket - a logikai műveletekhez hasonlóan – ún. *műveleti táblák* segítségével adunk meg.

Az előjel összeadás Ennek műveleti táblája a 4.1 táblázatban látható, ahol az univerzum atomi halmazaira vonatkozó részt kettős vonallal különítettük el. Innen leolvasható, hogy ez a művelet kommutatív (a tábla a főátlójára szimmetrikus), de *növeli a bizonytalanságot*, mert bizonyos esetekben határozott előjelű operandusok esetén a művelet eredménye határozatlan előjelű lesz, például
$+ \oplus - = ?$.

Az előjel szorzás A műveleti tábla (lásd 4.2 táblázat) hasonló az előjel összeadásához, ez a tábla is mutatja azt, hogy a művelet kommutatív. Ez esetben a bizonytalanság csökkenése figyelhető meg, hiszen van eset, amikor határozatlan előjelű operandus 0-val történő szorzása határozott előjelű (azaz 0) eredményt produkál.

$a \otimes_S b$	$+$	0	$-$	$?$
$+$	$+$	0	$-$	$?$
0	0	0	0	$\mathbf{0}$
$-$	$-$	0	$+$	$?$
$?$	$?$	$\mathbf{0}$	$?$	$?$

4.2. táblázat. Az előjel szorzás műveleti táblája

Az előjel aritmetika A fenti összeadás és szorzás, valamint az ezekből származtatott előjel kivonás és osztás műveletekkel generált aritmetika az ún. előjel aritmetika. Az előjel algebra az előjel univerzum feletti olyan algebrai struktúra, amely az előjel összeadással és szorzással, mint műveletekkel van ellátva. A közönséges valós aritmetikához hasonlóan ez a struktúra is nemcsak kommutatív mindkét műveletében, hanem asszociatív is.

4.1.3. Intervallum aritmetikák

Az intervallum aritmetikákhoz egy fix és valós végpontú intervallumokból álló univerzumot választunk alaphalmazként, és ezen definiáljuk az intervallum összeadás ($\oplus_{\mathcal{I}}$) és intervallum szorzás ($\otimes_{\mathcal{I}}$) műveleteket. Kétféle módon is definiálhatóak a fix végpontú intervallumon értelmezett aritmetikai műveletek.

- **Halmaz típusú definíció**: két intervallum, az $U_{\mathcal{I}}$ univerzumbeli $\mathcal{I}_1 = [a_{1\ell}, a_{1u}]$ és $\mathcal{I}_2 = [a_{2\ell}, a_{2u}]$ összegeként vagy szorzataként, amely az a legkisebb $U_{\mathcal{I}}$-beli intervallum, amely lefedi az

  $$\mathcal{I}^* = \{b = a_1 \textbf{ op } a_2 \mid a_1 \in \mathcal{I}_1,\ a_2 \in \mathcal{I}_2\}$$

  intervallumot.

- **Végpont típusú definíció**: *monoton műveletekre* az alábbi E_{op} halmaz felhasználásával, amelyet az operandus intervallumok végpontjaiból képezünk

  $$E_{op} = \{e_{\ell\ell} = a_{1\ell} \textbf{ op } a_{2\ell},\ e_{\ell u} = a_{1\ell} \textbf{ op } a_{2u},\\ e_{u\ell} = a_{1u} \textbf{ op } a_{2\ell},\ e_{uu} = a_{1u} \textbf{ op } a_{2u}\}$$

  Ezzel a $\mathcal{I}_1$ **op** $\mathcal{I}_2$ eredményt az

  $$\mathcal{I}' = [\min(E_{op}),\ \max(E_{op})]$$

  intervallum szolgáltatja.

Miután az összeadás és a szorzás monoton műveletek, az eredményt célszerűen a könnyen kiszámítható végpont típusú definició alapján kaphatjuk meg. Ezt a speciális intervallum-univerzumok (pl. nagyságrendi univerzum, lásd később) műveleti tábláiba szokták rendezni.

A fenti intervallum műveleteknek *szokatlan algebrai tulajdonságai* vannak annak következtében, hogy a $\mathcal{I}^*$ intervallum-eredményt *le kell **fedni*** egy $U_{\mathcal{I}}$-beli intervallummal. Ennek következtében számolni kell azzal, hogy

$a \oplus_{OM} b$	LN	SN	0	SP	LP
LN	LN	LN	LN	$[LN, SN]$	$[LN, LP]$
SN	LN	$[LN, SN]$	SN	$[SN, SP]$	$[SP, LP]$
0	LN	SN	0	SP	LP
SP	$[LN, SN]$	$[SN, SP]$	SP	$[SP, LP]$	LP
LP	$[LN, LP]$	$[SP, LP]$	LP	LP	LP

4.3. táblázat. A nagyságrendi összeadás műveleti táblája

- minden műveletnél *nőhet a bizonytalanság*
- *megszűnik a disztributivitás*, azaz egy összetett művelet eredménye függhet a kifejezés algebrai formájától. Be lehet látni, hogy az a forma a legalkalmasabb, amelyikben legkevesebb az összeadás művelete.

A nagyságrendi aritmetika Ez az aritmetika egy speciális, szimbolikus fix végpontokkal megadott intervallum univerzumra épül, amelyet az alábbi határpont-halmazzal definiálunk:

$$L_{\mathcal{OM}} = \{a_1 = -\infty,\ a_2 = -A,\ a_3 = 0,\ a_4 = A,\ a_5 = \infty\}$$

egy adott $A > 0$ értékkel. Az

$$U_{\mathcal{OM}} = \{LN,\ SN,\ 0,\ SP,\ LP\}$$

nagyságrendi univerzum atomi halmazai a $LN = [-\infty, -A)$, $SN = [-A, 0)$, $0 = [0, 0]$, $SP = (0, A]$, $LP = (A, \infty]$ intervallumok. Ezek segítségével értelmezhetünk ún. pszeudo-intervallumokat, pl. $[SP, LP] = (0, \infty]$ vagy $[LN, LP] = [-\infty, \infty]$, ezek lesznek a nagyságrendi aritmetikában a műveletek eredményei, például

$$LP \ \oplus_{OM} LN = [LN, LP]$$

Illusztrációképpen megadjuk a nagyságrendi összeadás műveleti tábláját a 4.3 táblázatban.

4.2. Kvalitatív modellek fajtái és származtatása

Az irányítástechnikai célra alkalmazható kvalitatív modellek olyan dinamikus modellek, amelyekben a változók és paraméterek értékkészlete véges, azaz egy univerzumot alkot. A modellegyenletekben szereplő aritmetikai műveleteket pedig az adott univerzumon definiált műveletekként értelmezzük. Gyakran a dinamikus modellben szereplő függvénykapcsolatokat is speciális kvalitatív függvényekkel helyettesítjük.

4.2.1. Kvalitatív modellek fajtái

Az értékkészlet univerzum felbontási finomsága, azaz az univerzum atomi intervallumainak száma, valamint a modellben szereplő relációk és műveletek jellege szerint többféle kvalitatív modell fajtát különböztetünk meg.

Előjel univerzumot használó modellek Ide tartoznak a dinamikus modellek szerkezetét súlyozott irányított gráfokkal leíró ún. *SDG modellek*, ahol csak a változók közötti hatások előjelét írjuk le. A dinamikus modellt alkotó algebrai és differenciálegyenletek előjel aritmetikán alapuló változatai pedig az ún. *konfluenciák*.

Intervallum univerzumot használó modellek Ha a dinamikus modellbeli algebrai és differenciálegyenleteket valamely intervallum aritmetikát használva oldjuk meg, akkor *algebrai típusú kvalitatív differenciálegyenlet* típusú modellekhez jutunk. Az ún. *megszorítás típusú kvalitatív differenciálegyenlet*ek ezektől a megoldás módjában és a modellekben szereplő függvénykapcsolatok kezelésének módjában különböznek.

A következő alfejezetekben bemutatjuk ezeket a kvalitatív modell típusokat, származtatási és megoldási módszereikkel és tipikus alkalmazási területeikkel együtt.

Fontos megjegyezni, hogy a mesterséges intelligencia szemszögéből nézve a kvalitatív modellek speciális tudásreprezentációs eszközöknek tekinthetőek, amelyekhez speciális következtetési formák is társulnak.

4.2.2. Kvaltatív modellek származtatása

Az irányítástechnikai célú kvalitatív modelleket fogalmilag és formálisan is a szokásos, differenciál és algebrai egyenletekből álló mérnöki modellekből származtatjuk. Ezeket a modelleket az irányítástechnikában szokásos *állapottér modell* alakúra hozzuk:

$$\begin{array}{rcll} \frac{dx}{dt} & = & f(x,u) & \text{(állapot egyenlet)} \\ y & = & h(x,u) & \text{(kimeneti egyenlet)} \end{array} \tag{4.4}$$

ahol f és h adott nemlineáris függvények.

A *kvalitatív modellek szisztematikusan és formálisan levezethetők* a fenti mérnöki modellekből intervallum értékű változók és paraméterek használatával, és esetenként egyszerűsített, ún. kvalitatív függvényeket alkalmazva.

4.3. Súlyozott irányított gráf modellek

A súlyozott irányított gráf modellek csak a dinamikus modellekben szereplő változók közötti hatások előjelét írják le, így a legkisebb felbontású kvalitatív modellnek tekinthetők.

4.3.1. Dinamikus modellek szerkezete

A (4.4) egyenletbeli nemlineáris állapottér modellt *linearizálhatjuk* egy *állandósult állapota* körül, ekkor az alábbi lineáris időinvariáns állapottér modellhez jutunk

$$\begin{array}{rcll} \frac{dx}{dt} & = & Ax + Bu & \text{(állapot egyenlet)} \\ y & = & Cx + Du & \text{(kimeneti egyenlet)} \end{array} \tag{4.5}$$

amelyet az (A, B, C, D) mártix-négyessel, mint paraméterekkel egyértelműen megadhatunk.

A linearizált állapottér modellben szereplő mátrixok struktúráját, azaz elemeik előjelét ún. struktúra mátrixokkal írhatjuk le. Az A numerikus mátrix $[A]$ struktúra mátrixát a következőképpen definiáljuk:

$$[A]_{ij} = \begin{cases} + & \text{if } a_{ij} > 0 \\ 0 & \text{if } a_{ij} = 0 \\ - & \text{if } a_{ij} < 0 \end{cases}$$

A struktúra mátrixok fogalmát felhasználva a (4.4) egyenletbeli nemlineáris állapottér modell szerkezete a belőle linearizálással kapott (4.5) lineáris időinvariáns modell paraméter mátrixainak struktúra mátrixaival, azaz az ($[A]$, $[B]$, $[C]$, $[D]$) struktúra mátrix négyessel írható le algebrailag.

4.3.2. Hatásgráfok

A hatásgráfok a dinamikus modellek szerkezetének leírására az algebrai leírással (az ($[A]$, $[B]$, $[C]$, $[D]$) struktúra mátrix négyessel) ekvivalens kombinatorikus struktúrát, ún. hatásgráfokat használnak.

Az $S = (V, \mathcal{E}; w)$ hatásgráf egy súlyozott irányított gráf, amelynek

- *csúcshalmaza* tartalmaz egy-egy csúcsot minden állapot, bemenet és kimenet változóra

$$V = X \cup U \cup Y$$
$$X \cap U = X \cap Y = U \cap Y = \emptyset$$

- *élei* megfelelnek a változók közötti *közvetlen* hatásoknak, azaz egy v_i csúcsból akkor mutat él egy v_j csúcsba, ha a v_j változót meghatározó modellegyenlet jobboldalán szerepel a v_i változó,

- *élsúlyai* a változók közötti közvetlen hatások *előjelei*

Fontos megjegyezni, hogy a *hatásgráfok előjellel súlyozott irányított gráfok (angolul Signed Directed Graph, rövidítve **SDG**).*

Példa: Kávéfőzőgép SDG modellje A kávéfőzőgép egy tartály, amelyben egy elektromos, ki-bekapcsolható fűtőtest melegíti a vizet, amelyet a csapból egy beömlő szeleppel pótolhatunk, és az elkészült forró vizet egy kiömlő szelep segítségével ereszthetjük rá az őrölt kávéra (a folyamatábrát ld. a 4.1 ábrán). A dinamikus működést leíró állapottér modellt a gépben levő vízre felírt tömeg és energia mérlegegyenletekből származtathatjuk az alábbi alakban:

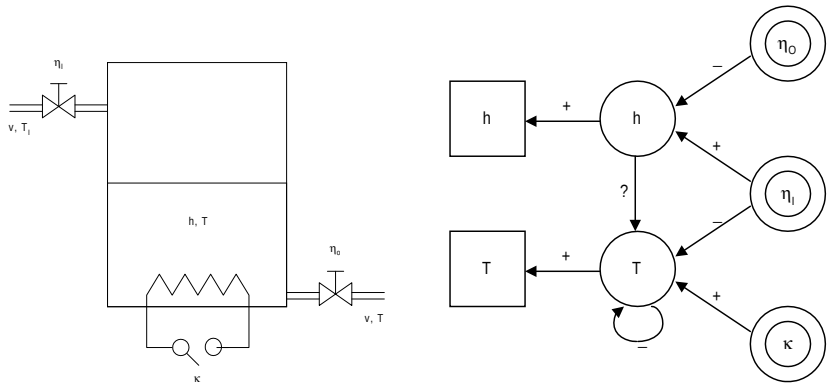
$$\frac{dh}{dt} = \frac{v}{A}\eta_I - \frac{v}{A}\eta_O \qquad \text{(tömeg)}$$

$$\frac{dT}{dt} = \frac{v}{Ah}(T_I - T)\eta_I + \frac{H}{c_p \rho h}\kappa \qquad \text{(energia)}$$

ahol

t	idő [s]	T_I	befolyási hőmérséklet [K]
h	vízszint [m]	H	fűtőteljesítmény [Joule/sec]
v	térfogatsebesség [m^3/s]	A	keresztmetszet [m^2]
c_p	fajhő [Joule/kgK]	η_I	bináris beömlő szelep [1/0]
ρ	sűrűség [kg/m^3]	η_O	bináris kiömlő szelep [1/0]
T	hőmérséklet [K]	κ	bináris fűtéskapcsoló [1/0]

Az SDG modellt szintén a 4.1 ábra mutatja, ahol az állapotváltozóknak megfeleltetett csúcsokat körrel, a bemeneti változókét kettős körrel, a kimeneti változókét pedig téglalappal jelöltük.



4.1. ábra. A kávéfőzőgép folyamatábrája és SDG modellje

Hatásgráfok tulajdonságai A definíció alapján könnyen belátható, hogy a három változóosztálynak megfeleltetett csúcsok a ki- és be-fokuk alapján megkülönböztethetőek. A kimeneti változóknak megfeleltetett csúcsokból nem indul ki él (ki-fokuk nulla), a bemeneti változóknak pedig a be-fokuk nulla.

4.3.3. Diagnosztikai következtetés SDG modelleken

Az SDG (azaz előjellel súlyozott irányított gráf) modellek igen népszerűek és elterjedten használatosak diagnosztikai alkalmazásokban, mert viszonylag kevés információ alapján felépíthetőek. Az előző alfejezetben megismert hatásgráfok mellett heurisztikus módon felépített SDG modelleket is használnak.

A meghibásodások detektálása és azonosítása (tehát a diagnosztika) érdekében a rendszer SDG formájában adott modelljén kívül a rendszer változóinak (bemenetek, állapotok és kimenetek) *előjel* értéket tulajdonítunk, amely az adott változó értékének *eltérését írja le annak jellemző állandósult állapotától*.

A változók közötti *közvetett* (azaz nem direkt) hatásokat pedig a hatásgráfokbeli $P = (v_1, v_2, \ldots, v_n)$, $v_i \in V$, $e_{i,i+1} = (v_i, v_{i+1}) \in \mathcal{E}$ irányított utakkal jellemezzük. Egy P *irányított út értéke* a benne szereplő élsúlyok szorzata, azaz

$$W(P) = \prod_{i=1}^{n-1} w(e_{i,i+1})$$

Diagnosztikai következtetések levonására a fentiek felhasználásával az SDG modell és a változók dinamikájának alábbi összefüggései nyújtanak lehetőséget. Tekintsük a v_i változó pozitív irányú megváltozásának hatását a tőle különböző v_j változóra.

- A v_j változó *kezdeti megváltozása* a két változót összekötő *legrövidebb irányított út(ak)* ***előjele***. Ha ez nem egyértelmű, akkor a legrövidebb utak értékének előjel-összegét kell kiszámítani, ami szerencsétlen esetben határozatlan előjelű lehet (lásd az előjel összeadás műveleti tábláját a 4.1 táblázatban).

- A v_j változó *állandósult állapotának eltérése* az összes lehetséges irányított utak ***előjel összege***, ami igen gyakran határozatlan előjelű. Ha ezen túlmenően az SDG modell irányított köröket is tartalmaz, akkor az állandósult állapotbeli eltérést lineáris előjel-egyenletrendszer megoldásaként kaphatjuk csak meg.

4.4. Konfluenciák

A konfluenciák a mesterséges intelligencia területén de Kleer and Brown [12] által kidolgozott ún. „kvalitatív fizika”-ból származnak, akik a fizikai modellekben szereplő algebrai és differenciálegyenleteket előjel-aritmetika felhasználásával alakítottak át kvalitatív differenciál-algebrai egyenletekké. A kapott konfluenciák segítségével egyszerű diagnosztikai következtetések végezhetők.

4.4.1. Konfluenciák származtatása

A konfluenciák az alábbi egyszerű lépésekben algoritmikusan is származtathatók egy közönséges differenciál és algebrai egyenleteket tartalmazó dinamikus modellből.

1. Minden modellbeli $q(t)$ változóhoz definiáljuk a $[q]$ és δq *kvalitatív változókat* a következőképpen:

   $$q \sim [q] = \mathrm{sign}(q), \quad dq/dt \sim \delta q = \mathrm{sign}(dq/dt)$$

   azaz tekintjük a változó és annak időbeli deriváltja előjel-értékét.

2. A modellbeli *műveleteket előjel műveletekre cseréljük*, azaz

   $$+ \quad \sim \quad \oplus_S, \qquad * \quad \sim \quad \otimes_S \qquad \text{stb.}$$

3. A modellben található *paramétereket a* $+$, $-$ vagy 0 *előjel konstansokra cseréljük* a konfluenciák egyenleteiben, azaz ezek látszólag eltűnnek az egyenletekből.

A konfluenciák megoldása A konfleunciák megoldását az előjel műveletek megadásához hasonlóan *igazságtábla* vagy műveleti tábla formájában adjuk meg, ahol a tábla első oszlopában a konfluencia egyenlet baloldalán szereplő kvalitatív változó értéke áll, a többi oszlopban pedig felsoroljuk az egyenlet jobboldalán szereplő összes változó összes lehetséges érték-kombinációját.

Fontos megjegyezni, hogy egy *konfluencia megoldás táblájá*nak mérete a benne szereplő változók számával exponenciálisan nő.

δh	$[\eta_I]$	$[\eta_O]$
0	0	0
$-$	0	$+$
$+$	$+$	0
?	$+$	$+$

4.4. táblázat. A (4.6) konfluencia megoldás táblája

Példa: Kávéfőzőgép tömegmérlegéből származtatott konfluencia Tekintsük a kávéfőzőgép dinamikus modelljében szereplő tömegmérleget:

$$\frac{dh}{dt} = \frac{v}{A}\eta_I - \frac{v}{A}\eta_O$$

Ebből a fenti algoritmussal az alábbi lépésekben készíthetünk konfluenciát:

1. *kvalitatív változók*: $[\eta_I] \in \{0, +\}$, $[\eta_O] \in \{0, +\}$
2. minden *előjel konstans* értéke „+”
3. a *konfluencia*

$$\delta h = [\eta_I] \ominus_S [\eta_O] \tag{4.6}$$

A konfluencia igazság- vagy megoldás táblája a 4.4 táblázatban látható.

4.4.2. Szabályrendszerek generálása konfluenciákból

A konfluenciák irányítástechnikai, pontosabban diagnosztikai célra úgy használhatóak, ha a megoldás táblájukban kódolt információt más, a diagnosztikai eljárások számára kezelhető alakra hozzuk. Erre az ad lehetőséget, hogy a *konfluencia megoldás táblájának egy sora szabályként értelmezhető* ha azt jobbról balra haladva olvassuk ki. Ezek a szabályok persze időfüggőek, lásd 3. fejezet.

Például a kávéfőzőgép tömegmérlegének

$$\delta h = [\eta_I] \ominus_S [\eta_O]$$

konfluenciájából az $\eta_I = 0$, $\eta_O = +$ érték-kombináció a $\delta h = -$ értéket adja, amiből a

$$\textbf{if } (\eta_I = \textbf{closed}) \textbf{ and } (\eta_O = \textbf{open}) \textbf{ then } (h = \textbf{decreasing})$$

szabály generálható.

Fontos megjegyezni, hogy a *konfluenciák megoldás tábláiból generált szabályrendszerek mindig* ***teljesek és ellentmondásmentesek***.

4.5. Kvalitatív differenciálegyenletek

A konfluenciák származtatásához hasonlóan a kvalitatív differenciálegyenlet modellek úgy készíthetőek a közönséges differenciál- és algebrai egyenletekből álló modellekből, hogy az azokban szereplő műveleteket valamely intervallum-aritmetika (lásd 4.1.3 alfejezet) felhasználásával alakítjuk át kvalitatív differenciál-algebrai egyenletekké.

4.5.1. Algebrai típusú kvalitatív differenciálegyenletek

Az algebrai típusú kvalitatív differenciálegyenlet alakú modellek a konfluenciák intervallum aritmetikát használó általánosításai.

Az egyenletek származtatása A kvalitatív modell egyenleteket a közönséges, folytonos idejű differenciál- és algebrai egyenleteket tartalmazó modellből a következő lépésekben állíthatjuk elő.

- A differenciál-egyenleteket időbeli diszkretizálással ***differenciaegyenletekké*** alakítjuk.
- Választunk egy alkalmas ***kvalitatív értékkészletet*** (univerzumot) a változóknak és paramétereknek.
- Képezzük az egyenletek kvalitatív formáját a megfelelő intervallum műveletekkel.

Az egyenletek megoldása A konfluenciákéhoz hasonlóan a megoldást itt is egy *megoldás tábla* formájában adjuk meg, amelyet úgy állítunk elő, hogy szisztematikusan összegyűjtjük az egyenlet jobboldalán álló változók összes lehetséges kvalitatív értékkombinációját a táblázat jobboldali oszlopaiban, és az ezekre kiszámított baloldali függő változó értéket a táblázat baloldali oszlopában tűntetjük fel. Fontos megjegyezni, hogy *a táblázat mérete a változók számával exponenciálisan nő.*

Példa: statikus szenzor additív hibával Tekintsünk egy szenzort (érzékelőt), amely a v valódi értéket egy additív E hibával méri, amelynek fennállását a χ ún. hiba-indikátor változó 1 értéke jelzi (hibamentes esetben $\chi = 0$), így a mért érték $v^m = v + \chi \cdot E$. Használjunk egy speciális nagyságrendi aritmetikát (lásd 4.1.3 alfejezet) a változók és konstansok értékeinek leírására, úgy, hogy

$$[v] \in \mathcal{Q} = \{0, L, N, H\}, \ , [v]^m \in \mathcal{Q}_e = \{e-, 0, L, N, H, e+\}, \chi \in B_{-1} = \{-1, 0, 1\} \quad (4.7)$$

és $[E] = L$, ahol L alacsony, N normális, H magas, $e-$ és $e+$ pedig kórosan alacsony illetve magas értéket jelöl.

Ekkor a szenzor kvalitatív modelljének megoldás táblája a a 4.5 táblázattal adható meg.

$[v^m]$	$[\chi]$	$[v]$	mode
N	0	N	normal
H	0	H	normal
L	0	L	normal
0	0	0	normal
$e+$	1	H	faulty
H	1	N	faulty
N	1	L	faulty
L	1	0	faulty
N	-1	H	faulty
L	-1	N	faulty
0	-1	L	faulty
$e-$	-1	0	faulty

4.5. táblázat. A statikus szenzor kvalitatív modelljének megoldás táblája

Dinamikus egyenletek megoldása A dinamikus egyenleteket ennél a megközelítésnél kvalitatív differenciaegyenletek formájúra hozzuk, amelyben a változók *diszkrét idejű és kvalitatív értékkészletű speciális jelek*.

Egy *esemény* egy kvalitatív jel értékének megváltozása, így a kvalitatív jelek időbeli viselkedését *eseménysorozat*okkal írhatjuk le. Például egy $x(t)$ jelnek egy adott időntervallumbeli viselkedését a 4.7 univerzumot felhasználva a (N, L, H) eseménysorozattal írhatjuk le.

Példa: Kávéfőzőgép tömegmérlegének modellje A kávéfőzőgép tömegmérlegét *diszkrét differencia egyenletté alakítva* a következő egyenlethez jutunk:

$$h(k+1) = h(k) + \chi_I(k) \cdot v - \chi_O(k) \cdot v \tag{4.8}$$

amelyben a változók 4.7 univerzumok segítségével felírt értékkészlete $[h] \in \mathcal{Q}_e$, χ_I, $\chi_O \in \mathcal{B}$ és $[v] = L$, valamint $x(k)$ az x változó k-adik diszkrét időpillanatbeli értéke.

Az egyszerűség kedvéért tételezzük fel, hogy a bemenő jelek (χ_I és χ_O) értéke időben nem változik, tehát pl. $(1, 1, 1)$. Ekkor a 4.8 egyenlet megoldás táblája három egymást követő diszkrét időpillanatra a 4.6 táblázattal adható meg.

4.5.2. Megszorítás típusú kvalitatív differenciálegyenletek

A megszorítás típusú kvalitatív differenciálegyenletek [8] különleges helyet foglalnak el a kvalitatív modellezési módszerek között nemcsak származtatásuk speciális módja, hanem megoldásuk előállításának különleges módja miatt is. Ezekben a kvalitatív differenciálegyenletekben nemcsak kvalitatív értékkészletű változók és paraméterek, hanem ún. kvalitatív függvények is előfordulhatnak, ami igen finom, a közönséges kvantitatív modellekhez közeli minőségű leírást tesz lehetővé.

$[h]^T$	$[h](t_0)$	χ_I	χ_O
(N, N, N)	N	(1,1,1)	(1,1,1)
(L, L, L)	L	(1,1,1)	(1,1,1)
...	...	...	...
(N, N, N)	N	(0,0,0)	(0,0,0)
...	...	...	...
$(e+, e+, H)$	N	(1,1,1)	(0,0,0)
$(e+, H, N)$	L	(1,1,1)	(0,0,0)
...	...	...	...
$(e-, 0, L)$	N	(0,0,0)	(1,1,1)
$(e-, e-, 0)$	L	(0,0,0)	(1,1,1)
...	...	...	...

4.6. táblázat. A 4.8 kvalitatív differenciaegyenlet megoldás táblája

A megszorítás típusú kvalitatív differenciálegyenletek és a megoldásukra használható kvalitatív szimuláció részletes ismertetése sajnos meghaladja jegyzetünk terjedelmi korlátait, az érdeklődő Olvasó a [6] irodalomban talál részletes ismertetést.

5. fejezet

Petri hálók

A *Petri háló* az információáramok absztrakt formális modellje. Elnevezését C.A. Petri német matematikusról kapta, aki 1962-ben publikálta a módszert soros automaták kommunikációjának modellezésére.

A Petri háló a modellezendő rendszerből a lehetséges állapotokat és a végbemenő eseményeket emeli ki. Egy esemény bekövetkezéséhez az azt előidéző állapotnak, vagyis az esemény előfeltételeinek teljesülnie kell, míg az esemény lejátszódása/bekövetkezése után új állapot jön létre, azaz esemény következményei jönnek létre. A hálóban, az előfeltételeket és a következményeket *helyeknek*, az eseményeket *átmeneteknek* nevezzük.

A kifejlesztése óta nagyon sokféle területen különböző rendszerek leírására alkalmazták ezt a hálótípust. Az egyes kutatók az alkalmazás során továbbfejlesztették az eredeti módszer modellezési erejét. Ennek megfelelően napjainkban megkülönböztetünk alacsony és magas szintű hálókat, valamint hierarchikus hálókat.

A Petri hálók kiválóan alkalmasak időben változó, azaz dinamikus rendszerek leírására. A szerkezetet leíró statikus modell szimulációja a rendszer időbeni viselkedésének vizsgálatát is lehetővé teszi. A modell elkészítése és működésének vizsgálata már önmagában is fontos információforrás, azonban a háló elemzésének elvégzésével a vizsgált rendszer további tulajdonságait ismerhetjük meg.

E fejezetben ismertetjük az ún. alacsony szintű Petri hálók fogalmát, legfontosabb tulajdonságait, és bemutatjuk az analízis lehetőségeit.

5.1. A Petri háló alapdefiníciói

5.1.1. A Petri háló és gráf

Egy alacsony szintű *C* Petri hálón a következő négyest értjük:

$$C =< P, T, I, O > \tag{5.1}$$

ahol $P = \{p_1, p_2, \ldots, p_n\}$ a helyek véges, nemüres halmaza; $T = \{t_1, t_2, \ldots, t_m\}$ az átmenetek véges, nemüres halmaza; $I : T \rightarrow P^{\infty}$ az előzményfüggvény; $O : T \rightarrow P^{\infty}$ a következményfüggvény.

A P és T halmazok diszjunktak, és az I és az O függvények definíciójában a P^{∞} kifejezés arra utal, hogy egy átmenet lejátszódásához egy előfeltétel teljesülésének többszörösen fenn kell állnia (pl. a szükséges memóriahelyek száma, vagy a szabad erőforrások száma), illetve az átmenet lejátszódása után a következmény többszörösen létrejöhet (pl. az egy művelettel előállított munkadarabok számának megfelelően). Mint látható, a hagyományos halmazelmélet helyett, annak kiterjesztését, az ún. bag-(csomag)-elméletet alkalmazzuk, melynél bármely elem többszörösen szerepelhet egy halmazban.

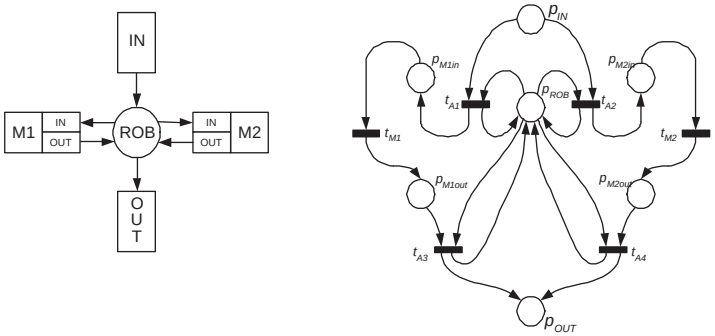
Az így megadott Petri háló közvetlen számítógépes feldolgozásra alkalmas, azonban nem túl szemléletes. Ezen segít a Petri gráf, mely a következő definícióval adható meg:

$$G =< V, A > \tag{5.2}$$

ahol $V = \{v_1, v_2, \ldots, p_s\}$ a csúcspontok véges, nemüres halmaza; $V = P \cup T$; $A = \{a_1, a_2, \ldots, a_r\}$ az irányított élek véges, nemüres halmaza; $a_i =< v_j, v_k >$ és $(v_j \in P \wedge v_k \in \in T) \vee (v_j \in T \wedge v_k \in P)$.

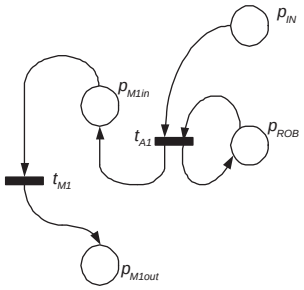
A fenti definíciónak megfelelően a Petri gráf egy irányított páros gráf. Az ábrázolás során a helyeket körökkel, az átmeneteket rövid vastag vonalakkal, a köztük lévő kapcsolatot nyilakkal jelöljük.

Példa Ebben a fejezetben a következő egyszerű példával illusztráljuk a Petri hálóval történő modellezés lehetőségeit. Legyen adott egy gyártó rendszer, melynek központi eleme egy robot (*ROB*)(5.1 ábra). A robot feladata, hogy az *IN* jelű belépő tárgyasztalra érkező munkadarabokat az *M1* vagy *M2* jelű technológia sorok fogadó tárgyasztalaira (*M1-IN* és *M2-IN*) rakja át. A kész munkadarabok a technológiai sorok kiadó tárgyasztalain jelennek meg (*M1-OUT* és *M2-OUT*), ahonnan a robot rakja át azokat a késztermékek asztalára (*OUT*). A gyártó rendszer vázlata és Petri hálója a 5.1. ábrán látható. A továbbiakban, az egyszerűbb tárgyalhatóság érdekében, csak a rendszer egy kiválasztott részműveletével foglakozunk.



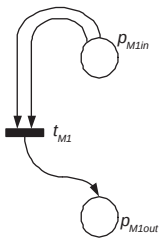
5.1. ábra. A példabeli gyártórendszer vázlata és Petri hálója

A Petri hálóval történő modellezés részletes bemutatására a példabeli rendszerből válasszuk azt a részegységet, melyben a robot a belépő tárgyasztalon (*IN*) megjelenő munkadarabot átrakja az *M1* gyártósor fogadó tárgyasztalára (*M1-IN*), ott megtörténik a megmunkálás, és a kész munkadarab megjelenik a gyártósor kimenetén (*M1-OUT*). A részművelet Petri hálója 5.2 ábrán látható.



5.2. ábra. A részművelet Petri hálója

Az *IN* asztalról az *M1* sor fogadójára történő átrakást t_{A1} átmenet modellezi. Az átrakáshoz szükség van az *IN* asztalon megjelenő munkadarabokra és a robotra, ezért a p_{ROB} (robot) és p_{IN} (belépő tárgyasztal) helyek az átmenet előfeltételei lesznek. A művelet befejeztével a munkadarab átkerül a gyártósor fogadó asztalára és felszabadul a robot, így a t_{A1} átmenet következményei a p_{ROB} és p_{M1in} helyek. Ezután megtörténik a megmunkálás, melyet a t_{M1} átmenet modellez, és a kész munkadarab megjelenik gyártó sor kiadó asztalán, azaz a p_{M1out} helyen. Az ismertetett rendszerben valamennyi hely egyszeres előfeltételként vagy következményként szerepel a hálóban. Ha feltételezzük, hogy a t_{M1} átmenet két munkadarab összeszerelését modellezi, akkor ennek a p_{M1in} helynek kétszeresen kell az átmenet előfeltételei között szerepelnie, amint ez a 5.3 ábrán látható.



5.3. ábra. Átmenet kétszeres előfeltétellel

5.1.2. A Petri háló jelfüggvénye

Mint a bevezetőben említettük, a Petri háló alkalmas a vizsgált rendszer viselkedésének vizsgálatára is. Ehhez azonban szükség van a vizsgált rendszer pillanatnyi állapotának megadására, melyet a jelfüggvény segítségével adhatunk meg.

A $C =< P, T, I, O >$ Petri háló μ jelfüggvényén a

$$\mu : P \rightarrow \mathcal{N} \tag{5.3}$$

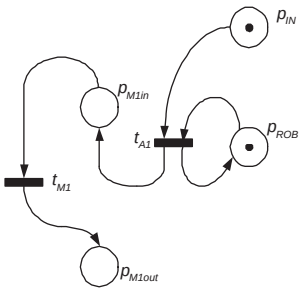
hozzárendelést értünk. Ennek megfelelően a μ jelfüggvény a háló grafikus reprezentációjában a helyekhez adott számú jelzőpontot, ún. tokent rendel hozzá. Ha egy helyen nincs jelzőpont, akkor a hely által reprezentált állapotot inaktívnak/passzívnak tekintjük, ha van rajta pont,

akkor aktív, ha pedig több pont is van rajta, akkor ha azt a hely értelmezése megengedi, akkor többszörösen aktívnak tekintjük.

A μ jelfüggvény segítségével tehát megadhatjuk az egy p_i helyen lévő jelzőpontok számát. Ha ezt egyidejűleg a háló valamennyi helyére megtesszük, akkor ezzel a jelzőpontoknak hálóbeli eloszlását írjuk le, ami megfelel a modellezett rendszer állapotának. A háló pillanatnyi állapotának leírásakor a μ jelfüggvényt, mint n-dimenziós vektort értelmezhetjük: $\mu^T = [\mu_1, \mu_2, \ldots, \mu_n]$, ahol $n = |P|$ és $\mu_i = \mu(p_i)$.

A helyek és a hozzájuk rendelt jelzőpontok értelmezése erősen függ a modellező szándékától. Így például, ha egy tartály telítettségét akarjuk modellezni, akkor ha a $p_{\text{tartály}}$ helyen nincs token, akkor ez egyaránt jelezheti azt, hogy a tartály üres, vagy nincs benne az előírt mennyiségű folyadék, míg ha van hozzárendelve jelzőpont, akkor a tartályban lévő folyadék szintje eléri a szintkapcsoló által meghatározott értéket. Bizonyos esetekben azonban szükség lehet a tartály üres állapotának modellezésére (pl. a betöltés megkezdése előtt). Az ekkor értelmezett $p_{\text{tartály\_üres}}$ hely által reprezentált üres tartály állapot akkor teljesül, ha $p_{\text{tartály\_üres}}$ helyen *van* jelzőpont. A jelzőpont hiánya arra utal, hogy a hely által reprezentált feltétel nem teljesül, vagyis egy adott minimális szintnél több folyadék van a tartályban. Nyilvánvaló, hogy ebben az esetben maximum egy jelzőpont lehet a $p_{\text{tartály\_üres}}$ helyen. Ugyanakkor, ha egy tartályparkban a $p_{\text{van\_üres\_tartály}}$ hely a pillanatnyi üres tartályok számát adja meg, akkor itt a jelzőpont hiánya arra utal, hogy valamennyi tartály foglalt, míg az egy vagy több jelzőpont az éppen üres tartályok számát jelentheti.

Példa – 1.folytatás Az ismertetett robotrendszerben a jelzőpontok jelentése kétféle lesz. A belépő tárgyasztalt és a gyártósor fogadó és kiadó asztalát modellező helyeken megjelenő jelzőpontok a munkadarabokat szimbolizálják, így, bizonyos megfontolások mellett, több is lehet belőlük egy adott helyen. A robotot szimbolizáló p_{ROB} helyhez rendelt jelzőpont az eszköz elérhetőségét (van rajta jelzőpont) vagy foglaltságát (nincs rajta jelzőpont) adja meg, így itt a jelzőpontok száma nulla vagy egy lehet. A részművelet jelzőpontokkal kiegészített hálója a 5.4 ábrán látható. Az háló egy olyan induló állapotot modellez, amikor egy munkadarab van az *IN* belépő tárgyasztalon és a robot szabad.



5.4. ábra. A részművelet induló állapota

5.1.3. Végrehajtási szabályok

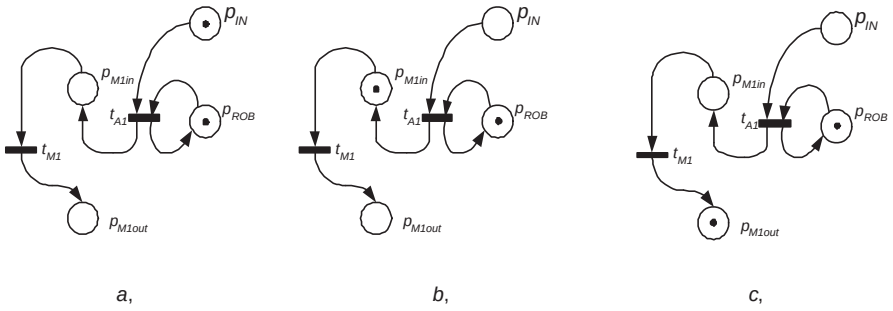
A Petri-háló végrehajtásán a modellezett rendszerben bekövetkező változásoknak a szimulációját értjük.

A változások szimulációját a helyeknek feltételekként, az átmeneteknek pedig eseményekként való értelmezése teszi lehetővé. A változások menetének követése a jelfüggvény segítségével végezhető el. Egy esemény lejátszódásához az előfeltételeinek teljesülniük kell. Ez a hálóban azt jelenti, hogy az eseményt modellező átmenet valamennyi bemeneti helyének aktívnak kell lennie. Az esemény bekövetkezte, vagyis az átmenet lejátszódása után az előfeltételek aktív állapota megszűnik és a következmények vagyis az átmenet kimeneti helyei lesznek aktívak. Azonban a modellezett rendszerben is előfordulhat, hogy egyidejűleg több lehetséges esemény közül csak egy következik be. Éppen ezért a hálóbeli átmenetek lejátszódását két részre bontjuk. Első lépésként megvizsgáljuk, hogy egy adott jelzőpont eloszlásnál mely átmenetek *engedélyezettek*, majd az engedélyezett események közül választjuk ki azokat, amik *le is játszódnak*.

Engedélyezettség. Egy t_j átmenet lejátszódása egy C Petri háló μ jelzőpont eloszlásánál akkor és csak akkor ***engedélyezett***, ha $\mu(p_i) \geq \sharp(p_i, I(t_j)), \forall p_i \in P$ esetén. A $\sharp(p_i, I(t_j)$ függvény megadja, hogy a p_i hely hányszor szerepel a t_j átmenet előzményhalmazában.

Lejátszódás. Egy t_j átmenet a C Petri hálóban csak akkor ***játszódhat le***, ha az engedélyezett. Az átmenet lejátszódása után a μ jelfüggvény új értékét a következő módon határozzuk meg: $\mu'(p_i) = \mu(p_i) - \sharp(p_i, I(t_j)) + \sharp(p_i, O(t_j))$.

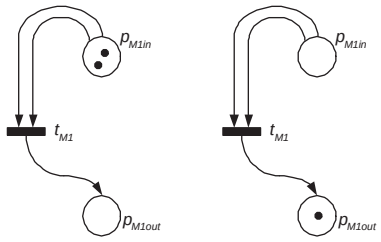
Példa – 2. folytatás A részművelet 5.5,*a* ábrán látható induló állapotában csak a t_{A1} átmenet engedélyezett, hiszen mindkét előfeltételén (p_{IN} és p_{ROB}) van egy-egy jelzőpont, ami megfelel az előzményhalmazbeli számosságuknak. A t_{M1} átmenet nem engedélyezett, hiszen a p_{M1in} helyen nincs jelzőpont. A t_{A1} lejátszódása után a 5.5,*b* ábrán látható állapot jön létre: a munkadarab átkerült a gyártósor fogadó asztalára és a robot újra szabad. Ebben a helyzetben indulhat a feldolgozás, azaz a t_{M1} átmenet, és a művelet befejeztével a munkadarab megjelenik a kiadó tárgyasztalon (5.5,*c* ábra).



5.5. ábra. A részművelet végrehajtása

Ha alkalmazzuk azt a korábbi feltevésünket, hogy a gyártósoron két munkadarabot szerelünk össze, akkor e művelet lejátszódásának modellezése a 5.6 ábrán követhető nyomon.

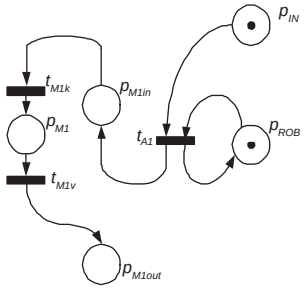
A Petri-háló végrehajtása során az átmenetek lejátszódását pillanatszerűnek tekintjük, és az ilyen átmeneteket *primitív átmeneteknek* nevezzük. A csak primitív átmeneteket tartal-



5.6. ábra. Két munkadarab összeszerelése

mazó hálóban az idő csak az átmenetek lejátszódásának sorrendiségében jelenik meg. Valós rendszereknél az események általában nem pillanatszerűen játszódnak le, és a lejátszódásuk során eltelő idő jelentősen befolyásolhatja a sorrendiséget is. Egyszerűbb esetekben ez a probléma feloldható úgy, hogy a nemprimitív esemény lejátszódását a hálóban két átmenettel (nemprimitív esemény kezdete; nemprimitív esemény vége) és egy hellyel (nemprimitív esemény lejátszódása) modellezzük. Bonyolultabb esetekben ez az eljárás nehézkes, és jelentősen megnöveli a háló méretét, ezért ekkor célszerű az idő fogalmának bevezetése az átmenetek lejátszódásánál.

Példa – 3. folytatás Az eddigiek során mind az átrakási műveletet, mind a megmunkálást pillanatszerű műveleteknek tekintettük. Ha ettől az egyszerűsítéstől, például a megmunkálásnál eltekintünk, akkor a t_{M1} átmenet felbontásával modellezhetjük ennek hosszabb időtartamát. Az 5.7 ábrán a t_{M1k} a megmunkálás kezdetét, a p_{M1} a megmunkálást és a t_{M1v} a megmunkálás végét jelenti.



5.7. ábra. A megmunkálási művelet felbontása

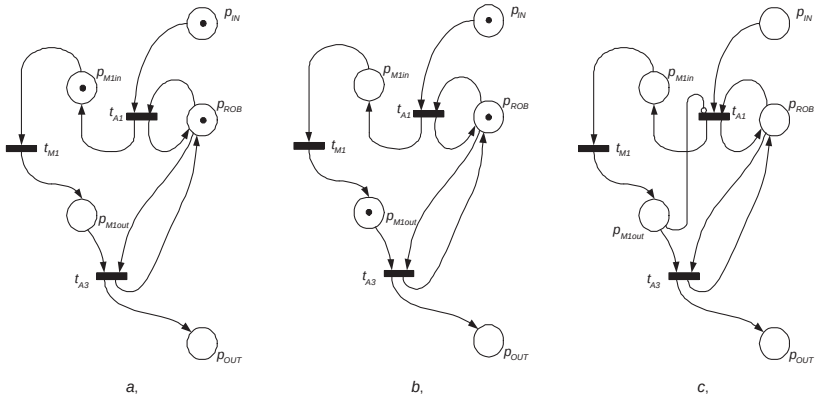
Konfliktus helyzet. Ha a háló egy adott állapotában egynél több átmenet engedélyezett, akkor meg kell vizsgálnunk, hogy ezek az engedélyezett átmenetek egymástól függetlenül lejátszódhatnak-e. Amennyiben az egyik átmenet lejátszódása után a másik átmenet engedélyezettsége megszűnik, akkor *konfliktus helyzet*ről beszélünk. A konfliktus helyzetben lévő átmenetek közül csak az egyik játszódhat le. Ez azt jelenti, hogy a modellezett rendszerben ilyenkor véletlenszerűen következik be az egyik vagy másik esemény, ami a rendszer nemdeterminisztikus működését jelenti.

A konfliktus helyzetek kezelésére az egyik lehetséges megoldás az *inhibítor nyíl* bevezetése. Az inhibítor nyíl segítségével a konfliktus helyzetben lévő átmenetek előfeltételei

közül kötünk egyet össze valamelyik átmenettel, és hatására ez az átmenet csak akkor lesz engedélyezett, ha ezen az előfeltételén nem lesz jelzőpont.

Amennyiben egy adott állapotban kettő vagy több átmenet egymástól függetlenül engedélyezett, azaz nincsenek konfliktus helyzetben, akkor ezek az átmenetek párhuzamosan lejátszódhatnak, és valamennyi ilyen átmenet lejátszódás után kapjuk meg a háló következő állapotát.

Példa – 4. folytatás A konfliktus helyzet és a párhuzamos lejátszódás bemutatására egészítsük ki a hálót azzal a művelettel, amikor a robot az elkészült munkadarabot a gyártósor kiadó asztaláról átrakja a késztermék asztalra. A háló a 5.8 ábrán látható. Az *a* ábrán a t_{A1} és t_{M1} átmenetek a Petri háló végrehajtási szabályai szerint egyaránt engedélyezettek, és miután bármelyik lejátszódása után a másik engedélyezett marad, ezért párhuzamosan bekövetkezhetnek. (Megjegyezzük, hogy természetesen lehet olyan technológiai előírás, mely szerint először a t_{M1} átmenetnek kell lejátszódnia, ha a fogadó asztalon maximum egy munkadarab lehet.) Tekintsük most a *b* ábrát. Itt a t_{A1} és a t_{A3} átmenetek egymással konfliktus helyzetben vannak, hiszen mindkét átrakási művelet elvégzéséhez a robot kell. Így bármelyik átmenetet is játszatjuk le, a másik engedélyezettsége megszűnik. A konfliktus helyzet – technológiai megfontolások alapján úgy oldható meg, hogy t_{A1} engedélyezettségéhez az is kell, hogy a p_{M1out} helyen *ne* legyen jelzőpont. Ezt a p_{M1out} helyről a t_{A1} átmenetbe vezető inhibítor nyíl segítségével oldhatjuk meg, amint ez a *c* ábrán látszik.



5.8. ábra. Párhuzamos lejátszódás és konfliktus helyzet

Egy adott μ hálóállapotban nulla, egy vagy több átmenet engedélyezett. Ha nincs engedélyezett átmenet, akkor a szimuláció befejeződött, azaz modellezett rendszer működése leállt. Ha csak egy engedélyezett átmenet van, akkor annak lejátszódásával folytatjuk a szimulációt, és a létrejövő új hálóállapotban újra megvizsgáljuk az átmenetek engedélyezettségét. Ha több engedélyezett átmenet van, és azok nincsenek egymással konfliktus helyzetben, akkor valamennyi átmenet lejátszódása után határozzuk meg az új hálóállapotot. Amennyiben kettő vagy több átmenet egymással konfliktus helyzetben van, akkor a szimulációt irányítónak kell döntenie arról, hogy kiválasztja-e a lejátszódó átmenetet, vagy megszakítja szimulációt és módosítja-e a hálót a konfliktus helyzet elkerülése érdekében.

5.2. A háló elemzése

A Petri háló segítségével végzett modellezésnek kettős: statikus és dinamikus célja van. A modell megalkotásával ismereteket szerezhetünk a vizsgált rendszer felépítéséről, az alrendszerek, elemek meghatározása és a köztük lévő kapcsolatok feltárása jelentősen segítheti a vizsgált rendszer szerkezetének a megértését. A hálóbeli átmenetek egy adott induló állapotból történő végrehajtása a rendszerbeli folyamatok lejátszódását szimulálja, azaz a létrejövő állapotokról és azok egymásra épüléséről kapunk információt.

A következőkben először bemutatjuk a Petri háló legfontosabb dinamikai tulajdonságait, majd azok vizsgálatának lehetőségeivel foglalkozunk.

5.2.1. A Petri háló dinamikai tulajdonságai

Korlátosság. A Petri háló alapértelmezése szerint egy adott helyen a jelzőpontok száma nem feltétlenül korlátos. A jelzőpontok számának tetszőleges, egynél nagyobb értéke bizonyos rendszerek esetében nem értelmezhető, ezért ezekben az esetekben célszerű megvizsgálni, hogy egy adott kezdőállapotból kiindulva az átmenetek végrehajtása során, az egyes helyeken mennyi lesz a jelfüggvény maximális értéke. Ez a tulajdonság a ***korlátosság*** definíciója alapján vizsgálható, mely szerint a háló egy helye akkor lesz korlátos, ha a jelzőpontok száma a végrehajtás során semmikor nem halad meg egy előre megállapított k értéket. Ha ez a feltétel a háló valamennyi helyére teljesül, akkor korlátos hálóról beszélünk. Ha a k értéke 1, akkor a hálót biztonsági hálónak szokás nevezni, $k > 1$ esetén pedig k-szorosan korlátos hálónak.

Példa – 5. folytatás A korlátosság vizsgálatát a példában a háló p_{M1in} és p_{M1out} helyeinél alkalmazhatjuk, ha feltételezzük, hogy az *M1* megmunkáló egység fogadó és kiadó tárgyasztalain egyidejűleg csak egy munkadarab lehet.

Konzervativizmus. A jelzőpontok nemcsak egy adott hely aktív vagy passzív állapotát jelezhetik, hanem megfelelhetnek például a gyártósoron végigmenő munkadarabnak is. Ilyen rendszerek modellezésénél lényeges kérdés lehet, hogy változik-e a jelzőpontok összege. Az olyan hálót, melyben a jelzőpontok összege konstans, ***szigorúan konzervatív*** hálónak nevezzük. Egy háló szigorú konzervativizmusa nagyon erős követelmény, ezért az ellenőrzését elvégezhetjük a háló helyeinek egy kiválasztott csoportjára is, vagy az egyes helyekhez súlyokat rendelve, vizsgálhatjuk a súlyozott összeg állandóságát is.

Példa – 6. folytatás A konzervativitást a háló azon helyeire vezethetjük be, ahol a jelzőpontok munkadarabot jelentenek, így a p_{IN}, a p_{M1in}, p_{M1out} és p_{OUT} helyeken a jelzőpontok összegének változatlannak kell lennie. (Megjegyezzük, hogy ha az átrakási műveleteket primitív műveletként modellezzük, akkor az egész hálóra teljesül a konzervatív tulajdonság, így a háló szigorúan konzervatív.) Ha viszont azt az eset választjuk, amikor két munkadarabot szerelünk össze a gyártósoron, akkor nem teljesül a konzervativitás (illetve csak súlyozással lehetne kielégíteni).

Átmenetek élősége. A rendszer működése során vizsgálhatjuk azt is, hogy egy esemény bekövetkezik-e az adott induló helyzetből. Ez a modellben annak ellenőrzését jelenti, hogy

az eseményt szimbolizáló átmenet a szimuláció során engedélyezett lesz-e az adott kezdőállapotból kiindulva. Azokat az átmeneteket, amelyek soha nem lehetnek engedélyezettek egy adott kezdőállapot esetén, *halott átmeneteknek* nevezzük, ellenkező esetben az átmenet *élő*. Egy átmenet élő volta azonban különböző lehet annak megfelelően, hogy csak egy adott hálóállapotban lesz-e engedélyezett, vagy ez több, akár tetszőleges hálóállapot esetén teljesül-e. Ennek alapján az átmenetek élősége különböző szinteknek feleltethető meg.

Ha egy rendszer Petri hálójának szimulációja során egy olyan állapotba kerülünk, amikor nincs egy engedélyezett állapot sem, akkor a szimuláció megszakad. Ez egyben azt is jelenti, hogy a hálóban egyetlen átmenetre sem teljesül a legmagasabb élőségi szint. Az ilyen helyzeteket holtpontnak nevezzük, és az olyan hálót, melyben nincs ilyen állapot, holtpontmentesnek hívjuk. Ezek a holtpontok természetesen nem mindig jelentenek hibás működést, hiszen megfelelhetnek a normális üzemmenet végének is, míg a holtpontmentesség utalhat végtelen ciklusba kerülésre.

Példa – 7. folytatás Példánkban természetes leállási pontot jelent valamennyi munkadarab feldolgozása, azaz ha indulóállapotban a p_{IN} helyen lévő összes jelzőpont „átkerül” a késztermékek helyére (p_{OUT}).

Elérhetőség. Az *elérhetőség* esetén azt vizsgáljuk, hogy a rendszer a működése során eljut-e egy kívánt vagy elkerülendő állapotba. Ez a vizsgálat a hálóban azt jelenti, hogy az adott kezdőállapotból indítva a szimulációt, valamennyi lejátszódási lehetőséget megvizsgálva létrejön-e a kérdéses hálóállapot. Az elérhetőségi vizsgálatnak kiterjesztése a *lefedési vizsgálat*. Egy hálóállapot akkor fed le egy előre megadott állapotot, ha annál a jelfüggvény értéke valamennyi helyen nagyobb vagy egyenlő a megadott állapotban az adott helynél szereplő értéknél.

Az elérhetőség speciális esete a *visszafordíthatóság*. A visszafordítható hálóknál tetszőleges állapotból elérhető a kezdőállapot.

Ugyancsak az elérhetőségi vizsgálatok tárgya lehet az *alapállapot* létének az ellenőrzése. Alapállapotnak azt a hálóállapotot nevezzük, amely bármely más hálóállapotból elérhető. Ez a kitüntetett állapot egy technológiai rendszer esetében megfeleltethető az egyensúlyi állapotnak.

5.2.2. A Petri háló analízisének lehetőségei

Az elkészült modell vizsgálatával, vagyis a háló analízisével választ kaphatunk a dinamikai tulajdonságokra vonatkozó kérdésekre. A háló analízisét gyakorlati és formális módszerekkel lehet elvégezni. Az első csoportba a szimuláció tartozik, míg a tényleges analízist matematikai módszerek segítségével lehet elvégezni.

A Petri háló elemzése két problémaosztályra bontható. Az első esetben azt vizsgálhatjuk meg, hogy egy adott kezdőállapotból indítva, milyen megállapítások tehetők a rendszer működésére. A vizsgálatok másik csoportja az induló állapottól független tulajdonságok elemzését végzi el. Az adott induló állapothoz tartozó tulajdonságok felderítését az elérhetőségi fa elemzésén alapuló módszerek, míg az attól független tulajdonságok vizsgálatát a hely- és átmenetinvariánsokat felhasználó eljárások segítségével lehet vizsgálni.

5.2.3. A háló szimulációja

A Petri hálóban lejátszódó átmenetek és a létrejövő állapotok legegyszerűbb és talán legszemléletesebb eszköze a *szimuláció*. A szimuláció során, egy adott kezdőállapotból kiindulva, a végrehajtási szabályoknak megfelelően végrehajtjuk az indítható átmeneteket. A szimuláció következetes végigvitelével a 5.2.1 fejezetben ismertetett tulajdonságok egy része közvetlenül megfigyelhető, de azok általánosságban nem igazolhatóak. Egyszerűbb hálók esetében a szimuláció akár kézzel is elvégezhető, de számos – sok esetben ingyenes – szoftver eszköz is létezik, ami segíti a háló felépítését, és a szimuláció elvégzését.

A szimuláció tehát csak a durva hibák kiszűrésére ad lehetőséget, és alkalmazásával nem bizonyíthatóak az ellenőrizni kívánt dinamikai tulajdonságok. A szimuláció fő ereje, hogy különösen összetett rendszerek esetében segíti a modell felépítését, a működésének megértését, illetve az elsődleges vizsgálatok elvégzését.

5.2.4. Az elérhetőségi fa

Ha egy adott kezdőállapotból kiindulva felvesszük az onnan elérhető valamennyi lehetséges állapotot, akkor megalkothatjuk a háló *elérhetőségi halmazát*. Ezt a halmazt, a könnyebb áttekinthetőség kedvéért, főleg kevés számú állapot esetén szokás gráfként, ún. *elérhetőségi fa* formájában ábrázolni.

Egy tetszőlegesen megadott μ_k kezdőállapotból elérhető valamennyi hálóállapotot tartalmazó $R(\mu_k)$ elérhetőségi halmazt a következő módon hozhatjuk létre:

1. $\mu_k \in R(\mu_k)$,
2. ha $\mu' \in R(\mu_k)$ és létezik olyan t_i átmenet, mely a μ' állapotban engedélyezett, akkor a t_i átmenet lejátszódása után létrejövő μ'' állapotra is igaz: $\mu'' \in R(\mu_k)$.

Az elérhetőségi fa felépítésének a szabályai a következők:

1. A fa gyökere a μ_k kezdőállapot lesz.
2. Ahány átmenet engedélyezett a kezdőállapotban, annyi ágat, azaz irányított élet kapcsolunk a gyökérhez. Ezeknek az éleknek a csúcspontján az él által reprezentált átmenet lejátszódása után létrejövő új állapot szerepel, az élhez pedig hozzárendeljük a lejátszatott átmenet szimbólumát.
3. Ha kapott új állapotban nincs engedélyezett átmenet, azaz a neki megfelelő ágponthoz nem lehet további ágpontot hozzákötni, akkor a fa egy zárópontjához jutottunk. Ekkor, ha van még olyan csúcspont, amelynél nem ellenőriztük az átmenetek indíthatóságát, akkor azzal folytatjuk a fa építését, ha pedig valamennyi csúcsot ellenőriztük, akkor megkaptuk az adott induló állapothoz tartozó elérhetőségi fát.
4. Ha a kapott új állapotban van engedélyezett átmenet, akkor annak lejátszatása előtt megvizsgáljuk, hogy az ágpontnak megfelelő állapot nem szerepel-e valahol korábban a fán. Ha igen, akkor ez azt jelenti, hogy az onnan elérhető ágpontok is szerepelnek, tehát a fa méretének csökkentése érdekében nem érdemes a fát ebből az ágpontból

továbbépíteni. Ilyenkor ismétlődő ponthoz jutottunk és utalunk az első előfordulás helyére.

5. További redukciós lehetőséget jelent annak ellenőrzése, hogy nincs-e a gyökértől az adott ágpontig vezető úton olyan állapot, amelytől kezdve egy vagy több helynél a jelzőpont-eloszlás értéke monoton növekvő. Ha van ilyen hely, akkor ez azt jelenti, hogy ott egy olyan körút található, amely tetszés szerinti számban bejárható, és ennek megfelelően a jelzőpontok száma az adott helyen tetszőleges nagy lehet. Ekkor, bár elvileg különböző állapotokat kapunk, a fa véges méretének megőrzése érdekében, ezen a helyen a jelzőpont-eloszlás értékét külön szimbólummal (pl. ∞) jelöljük, és az ismétlődés ellenőrzését is ennek alapján végezzük el.

6. Amennyiben az új ágpont nem ismétlődő ágpont, akkor az engedélyezett átmenetek lejátszatjuk és a 2. ponthoz visszatérve folytatjuk a fa felépítését.

A 5.2.1 fejezetben felsorolt dinamikai tulajdonságok vizsgálata az elérhetőségi fa esetében elsősorban keresési módszerek segítségével végezhető el. Így ellenőrizhetjük, hogy

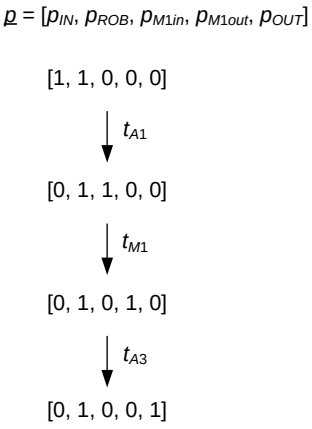
- az egyes ágpontoknál a jelfüggvény értékei korlátosak-e (nyilvánvaló, hogy ha egy helyen a ∞ szimbólum megjelenik, akkor ott nem) – *korlátosság vizsgálat*;
- az egyes ágpontoknál a jelzőpontok összege konstans-e – *konzervatív jelleg vizsgálata*;
- van-e olyan átmenet, mely egy élhez sincs hozzárendelve, illetve melyik átmenet milyen körülmények között indítható – *átmenetek élőségének vizsgálata*;
- a fa záróspontjai leállási állapotok-e – *holtpont mentesség vizsgálata*;
- szerepel-e olyan állapot fán, amelynél a jelzőpont-eloszlás értéke megegyezik egy előre megadott értékkel, illetve ez lefedi-e azt – *elérhetőségi, lefedési, visszafordíthatósági vizsgálatok*.

Mint látható, az elérhetőségi fán alapuló elemzés segítségével meghatározhatóak a háló dinamikai tulajdonságai, de a módszer nagy hátránya, hogy a fa még a bevezetett egyszerűsítések ellenére is könnyen nagyon nagy méretű lehet, ami megnehezíti a vizsgálatok elvégzését.

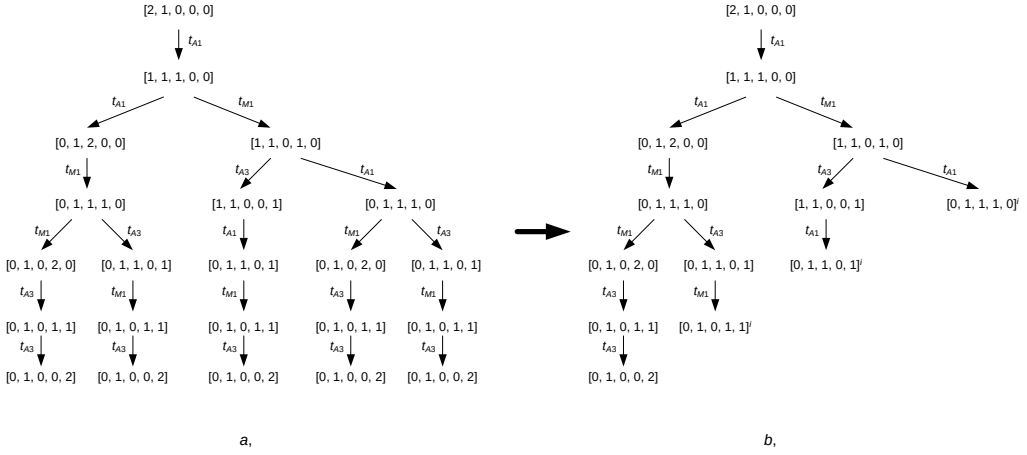
Példa – 8. folytatás Vizsgáljuk meg a robotos példa részhálójának (5.5. ábra) elérhetőségi gráfját a következő induló állapot esetén: egy munkadarab van *IN* belépő tárgyasztalon és a robot szabad. A kapott elérhetőségi gráf a 5.9 ábrán látható.

A kisméretű háló és az alkalmasan megválasztott kezdőállapot miatt az elérhetőségi fa nagyon egyszerű szerkezetű lett, melyen a felsorolt elemzések könnyen elvégezhetők.

Módosítsuk az induló állapotot úgy, hogy a belépő tárgyasztalon induláskor két munkadarab várjon feldolgozásra. A 5.10.*a* ábrán látható elérhetőségi gráf lényegesen nagyobb méretűvé vált. A fa csúcspontjainak vizsgálata alapján megállapíthatjuk, hogy több ismétlődő állapot is előfordul, így ezek figyelembe vételével a *b* ábrán látható egyszerűbb fát kapjuk.



5.9. ábra. Elérhetőségi fa egy munkadarab esetén



5.10. ábra. Elérhetőségi fa két munkadarab esetén

5.2.5. Invariáns analízis

A Petri hálók elemzésének egy másik lehetősége a háló algebrai reprezentációján alapul. Az ún. *hely-* és *átmenetinvariánsok* megkereséséhez alkossuk meg a háló incidencia mátrixát. Egy Petri háló incidencia mátrixa alatt a következő mátrixot értjük:

$$W = W^+ - W^- \tag{5.4}$$

ahol

- W^+ egy olyan mátrix, amely az átmenetek számával megegyező számú oszlopot, és a helyek számának megfelelő számú sort tartalmaz, és $W^+(i,j) = \sharp(p_i, O(t_j))$;
- a W^- mátrix dimenziója megegyezik a W^+ mátrixéval, és $W^-(i,j) = \sharp(p_i, I(t_j))$.

A helyinvariánsok. A Petri háló egy *helyinvariánsán* a következő egyenletrendszer egy megoldás vektorát értjük:

$$\underline{z}^T \cdot W = \underline{0} \tag{5.5}$$

A $\underline{z}$ vektor egy súlyvektornak felel meg, amely megadja, hogy egy hely szerepel-e az adott megoldáshoz tartozó helyinvariánsban. Belátható, hogy a helyinvariáns alapján a hálóban az átmeneteknek egy olyan sorozatát lehet meghatározni, melynek lejátszódása során a helyinvariánsban szereplő helyeken a jelzőpontok összege változatlan marad.

Az átmenetinvariánsok. A Petri háló egy *átmenetinvariánsán* a következő egyenletrendszer egy megoldás vektorát értjük:

$$W \cdot \underline{v} = \underline{0} \tag{5.6}$$

A $\underline{v}$ vektor ebben az esetben is egy súlyvektornak felel meg, amely megadja, hogy egy átmenet szerepel-e az adott megoldáshoz tartozó átmenetinvariánsban. Az átmenetinvariáns az átmenetek olyan sorozatának feleltethető meg, melynek végrehajtása után a háló visszakerül a kezdőállapotába. Az átmenetinvariánsok és a helyinvariánsok egymásnak duálisai.

Az invariánsok segítségével végzett analízisnek számos kedvező tulajdonsága van. A háló incidencia mátrixa minden esetben véges ábrázolása a háló szerkezetének, és miután általában egy átmenet csak kevés számú hellyel áll kapcsolatban, ezért az incidencia mátrixok ritka mátrixok. A hely- és átmenetinvariánsok keresése lineáris egyenletrendszerek megoldását jelenti. A helyinvariánsokban szereplő helyek korlátosak és konzervatívak, a helyinvariánsban szereplő állapotok elérhetőek. Az átmenetinvariáns alapján dönteni lehet az átmenetek élőségéről, és bizonyos esetekben a visszafordíthatóságról és alapállapotról. Az invariánsok a tervezéshez a priori rendszertulajdonságokként megadhatók, és a háló felépítése során a meglétük ellenőrizhetők. További egyszerűsítést jelenthet, ha az invariánsanalízist nem a teljes hálóra, hanem annak csak egy kiválasztott részére végezzük el.

6. fejezet

Fuzzy irányítási rendszerek

A bizonytalan és/vagy kvalitatív információn alapuló irányítási és diagnosztikai módszerek tervezésének és megvalósításának napjainkban egyik legelterjedtebb, és méltán népszerű módszere a *fuzzy halmazok* elméletén és az azokon való műveleteken alapul. Ebben a fejezetben megismerkedünk a fuzzy irányítórendszerek használatának és tervezésének alapjaival.

6.1. Fuzzy aritmetika

A fuzzy aritmetika a fuzzy halmazokkal való műveletek definícióját és tulajdonságait tárgyalja.

6.1.1. Fuzzy halmazok

A fuzzy halmazokat a közönséges halmazok általánosításaként definiáljuk úgy, hogy a halmaz minden eleméhez egy 0 és 1 közötti úgynevezett tagsági értéket rendelünk, amely megmondja, hogy az adott elem „mennyire része" a halmaznak (1=teljesen, 0=semennyire). Formálisan tekintsünk egy véges zárt valós halmazt, $U \subset \mathbb{R}$ (az alaphalmazt, vagy *univerzum*-ot) és egy ezen értelmezett μ tagsági függvényt, amelyre $1 \leq \mu(x) \leq 0$ minden $x \in U$. Ekkor az (U, μ) pár egy fuzzy halmaz.

Az ugyanazon univerzum felett értelmezett fuzzy halmazokat *nyelvi azonosítók*kal látjuk el és jellemezzük, például „*kicsit meleg*", „*nagyon hideg*", stb.

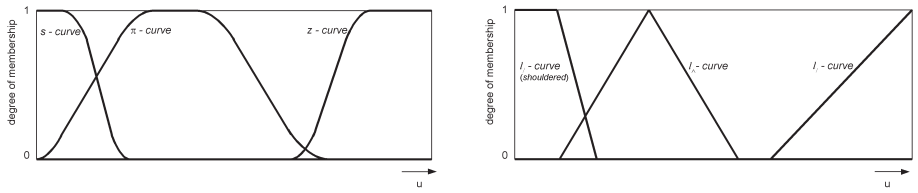
A tagsági függvényeket a gyakorlatban használatos fuzzy irányító rendszerekben adott függvényosztályból lehet kiválasztani. A leggyakoribb valós intervallum univerzumra alkalmazható függvényosztályok a folytonos függvényekből álló s, z, π, valamint a szakaszonként lineáris függvényekből álló *lineáris függvénycsalád* (lásd 6.1 ábra).

Ha az *univerzum egy diszkrét halmaz*, azaz $U = \{x_1, \ldots, x_n\}$, akkor az A fuzzy halmazt egy $\langle x_i, \mu(x_i)\rangle$ párokból, úgynevezett *fuzzy singletonok*ból álló halmazzal jellemezhetjük:

$$A = \{\langle x, \mu(x)\rangle\} = \{\langle x_1, \mu(x_1)\rangle, \langle x_2, \mu(x_2)\rangle, ldots, \langle x_n, \mu(x_n)\rangle\}$$

amelyből – az univerzum halmaz pontjait elhagyva – az alábbi, sor-vektor formájában adott reprezentációt kapjuk:

$$a = \mu_{\mathbf{a}} = [\mu(x_1)\ \mu(x_2)\ \ldots\ \mu(x_n)]$$



6.1. ábra. Tagsági függvény családok

6.1.2. Fuzzy műveletek

Az úgynevezett primitív fuzzy halmaz műveletek a közönséges alapvető halmazműveletek, az egyesítés (únió), metszet és a komplemens képzés fuzzy kiterjesztései. Ezeket az $A = \{\langle x, \mu_A(x)\rangle\}$ és $B = \{\langle x, \mu_B(x)\rangle\}$ közös U univerzum feletti fuzzy halmazokra értelmezhetjük.

- **fuzzy egyesítés**: $A \cup B = A$ **or** $B \triangleq A$ **max** B
  $\mu_{A\cup B}(x) = \mathbf{max}(\mu_A(x), \mu_B(x))$ minden $x \in U$-ra
- **fuzzy metszet**: $A \cap B = A$ **and** $B \triangleq A$ **min** B
  $\mu_{A\cap B}(x) = \mathbf{min}(\mu_A(x), \mu_B(x))$ minden $x \in U$-ra
- **fuzzy komplemens**: $\neg A =$ **not** $A \triangleq 1 - A$
  $\mu_{\neg A}(x) = 1 - \mu_A(x)$ minden $x \in U$-ra

Fontos megjegyezni, hogy a fent definiált fuzzy műveleteknek a közönséges halmazműveleteknél megszokott tulajdonságai vannak, azaz

kommutativitás	a **or** $b = b$ **or** a
	a **and** $b = b$ **and** a
asszociativitás	$(a$ **or** $b)$ **or** $c = a$ **or** $(b$ **or** $c)$
	$(a$ **and** $b)$ **and** $c = a$ **and** $(b$ **and** $c)$
disztributivitás	a **or** $(b$ **and** $c) = (a$ **or** $b)$ **and** $(b$ **or** $c)$
	a **and** $(b$ **or** $c) = (a$ **and** $b)$ **or** $(b$ **and** $c)$
DeMorgan	**not** $(a$ **and** $b) = ($ **not** $a)$ **or** $($**not** $b)$
	not $(a$ **or** $b) =$ (**not** a) **and** (**not** b)
abszorpció	$(a$ **and** $b)$ **or** $a = a$
	$(a$ **or** $b)$ **and** $a = a$
idempotencia	a **or** $a = a$
	a **and** $a = a$
kizárás	a **or** $\neg a \neq 1$
(nem teljesül!)	a **and** $\neg a \neq \emptyset$

A fuzzy műveletek fenti definíciója a legelterjedtebb, a legtöbb esetben ezt használjuk, de fontos tudni, hogy több lehetséges módon is megadhatók az egyesítés, metszet és komplemens műveletek fuzzy halmazokon, például az úgynevezett T-normák segítségével (részletesen lásd például [9]).

Példa: Fuzzy metszet Tekintsük a gépkocsik hengerűrtartalmának értékeiből álló

$$U = \{1.0, 1.2, 1.4, 1.6, 1.8, 2.0\}$$

univerzumot, és e felett két fuzzy halmazt.

Az *alacsony fogyasztás (LC)* fuzzy halmaz az alábbi tagsági függvénnyel adott.

U	1.0	1.2	1.4	1.6	1.8	2.0
μ_{LC}	1.0	0.9	0.7	0.5	0.2	0.0

A *nagy gyorsulás (HA)* fuzzy halmaz pedig az alábbi:

U	1.0	1.2	1.4	1.6	1.8	2.0
μ_{HA}	0.0	0.1	0.4	0.5	0.8	1.0

Ezekkel az *alacsony fogyasztás* **and** *nagy gyorsulás* fuzzy halmaz formálisan is képezhető az alábbi tagsági függvénnyel a közös univerzum felett.

U	1.0	1.2	1.4	1.6	1.8	2.0
μ_{LC}	1.0	0.9	0.7	0.5	0.2	0.0
μ_{HA}	0.0	0.1	0.4	0.5	0.8	1.0
$\mathbf{min}(\mu_{LC}, \mu_{HA})$	0.0	0.1	0.4	0.5	0.2	0.0

Ez a fuzzy halmaz megfelel az elvárásainknak, hiszen az alacsony fogyasztású és jó gyorsulású gépkocsik valóban a közép-kategóriás hengerűrtartalmú példányok.

Fuzzy nyelvi módosítók Ezek olyan egyoperandusú műveletek, amelyek egy adott tagsági függvény alakját, illetve érték halmazát változtatják meg a mindennapi nyelvben megszokott módosítóknak (például „*nagyon*”, „*kicsit*”, stb.) megfelelően. Az alábbi módosítókat szokták használni.

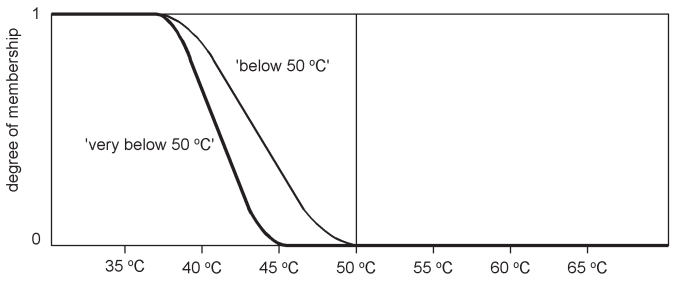
- *Fuzzy halmazok közelítése*: egy skalár értékkel megváltoztatja az eredeti fuzzy halmaz tartóját (azokat az értelmezési tartománybeli pontokat, ahol a tagsági függvény nem nulla), ilyenek a **about**, **around**, **near** és a **close to** operátorok.
- *Fuzzy halmazok megszorítása*: megváltoztatja az eredeti fuzzy halmaz alakját, ilyenek a **below** és a **above** operátorok.
- *Fuzzy halmazok élesítése és higítása*, amelyet formálisan a

  $$\mathbf{op}\ \mu(x) = \mu^n(x)$$

  módon értelmezünk. Ezen belül megkülönböztetünk

  - *élesítést (intensification)*: a **very** ($n = 2$) és a **extremely** ($n = 3$) operátorokkal,
  - *higítást (dilution)*: a **somewhat** ($n = 1/2$) és a **greatly** ($n = 5/7$) operátorokkal.

Egy öszetett nyelvi módosításra láthatunk példát a 6.2 ábrán.



6.2. ábra. Tagsági függvény összetett nyelvi módosítása

6.2. Következtetés fuzzy szabályokon

Az előző 6.1 alfejezetben megismert fuzzy alapműveletek segítségével fuzzy relációkat és fuzzy implikációt (azaz fuzzy szabályokat) is értelmezhetünk.

6.2.1. Fuzzy relációk

A fuzzy relációkkal két fuzzy halmaz elemeinek kapcsolatát (relációját) írhatjuk le fuzzy módon.

Tekintsünk két univerzumot, P-t és Q-t. Egy ezek között értelmezett $\mathbf{R}_{PQ}$ *(bináris) fuzzy reláció* leírja, hogy egy P-beli a elem *milyen mértékben (mennyire) van relációban* egy Q-beli b elemmel. A fuzzy relációkat úgynevezett *relációs táblával* adjuk meg, amely – diszkrét univerzumokat feltételezve – minden (a, b) párhoz megad egy tagsági értéket (0 és 1 közé eső számot). Ez a relációs tábla egy *reláció-mátrix*ként is felfogható, amelynek $\dim(P)$ sora és $\dim(Q)$ oszlopa van, ahol $\dim(X)$ az X diszkrét univerzum elemeinek számát jelenti.

Példaként egy 3 elemű P és egy 2 elemű Q halmaz közötti fuzzy reláció relációs tábláját mutatjuk be.

$\mathbf{R}_{PQ}$	q_1	q_2
p_1	0.3	0.9
p_2	0.4	0.5
p_3	0.1	0.8

Fuzzy halmazok kompozíciója Bináris fuzzy relációkat létrehozhatunk két fuzzy halmaz kompozíciójával, vagy más néven *belső max-min szorzatával* is a következő módon. Tekintsünk két fuzzy halmazt U-t és V-t (nem feltétlenül ugyanazon univerzum felett). Ekkor ezek kompozíciója W az alábbi reláció-mátrix alakban definiálható:

$$W = U \circ V \tag{6.1}$$

$$w_{ij} = (u_{i1} \wedge v_{1j}) \vee (u_{i2} \wedge v_{2j}) \vee \ldots \vee (u_{ip} \wedge v_{pj}) = \bigvee_{k=1}^{p} u_{ik} \wedge v_{kj}$$

ahol $\circ$ az *inner or-and product* vagy másképpen a *max-min kompozíció*. Látható, hogy W egy speciális fuzzy reláció U és V között.

Implikáció fuzzy halmazokon Legyen A és B két fuzzy halmaz, nem feltétlenül ugyanazon az univerzumon. Az $\mathbf{R}_{A\to B}$ *fuzzy implikáció* egy, a két fuzzy halmazon értelmezett *fuzzy reláció*

$$A \longrightarrow B \quad \triangleq \quad A^T \times B$$

ahol $\times$ az alábbi mátrixok (az A^T oszlop, illetve B sorvektor) ún. *külső szorzata* amelyet a fuzzy **and** ($\wedge$) művelettel állíthatunk elő

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \times \begin{bmatrix} b_1 & b_2 & \dots & b_m \end{bmatrix} = \begin{bmatrix} a_1 \wedge b_1 & a_1 \wedge b_2 & \dots & a_1 \wedge b_m \\ a_2 \wedge b_1 & a_2 \wedge b_2 & \dots & a_2 \wedge b_m \\ \vdots & & & \vdots \\ a_n \wedge b_1 & a_n \wedge b_2 & \dots & a_n \wedge b_m \end{bmatrix} \tag{6.2}$$

Fontos megjegyezni, hogy a fuzzy implikáció fenti definíciója csupán a legelterjedtebb, az úgynevezett Mandani implikáció. Egyéb lehetséges implikációk is használatosak, például a Sugeno-típusú fuzzy következtetésnél [9].

Példa: Fuzzy implikáció Tekintsünk egy e hibajelet, és egy u bemeneti változót az alábbi univerzumokkal

$$U_e = \{-5,\ -2.5,\ 0,\ 2.5,\ 5\}\ ,\ U_u = \{-2,\ 0,\ 2\}$$

A fenti diszkrét univerzumok felett az alábbi fuzzy halmazokat definiáljuk.

- az e hibajelre

nagy pozitív eltérés (**lpe**)	$\mu_{\mathbf{lpe}}$ = [0 0 0 0.6 1]
kis pozitív eltérés	$\mu_{\mathbf{spe}}$ = [0 0 0.3 1 0.3]

- az u bemenetre

pozítív szabályozó jel (**pcs**)	$\mu_{\mathbf{pcs}}$ = [0 0.2 1]
nulla szabályozó jel	$\mu_{\mathbf{zcs}}$ = [0.1 1 0.1]

Számítsuk ki a $\mathbf{R}_{A\to B}$ fuzzy implikáció reláció-mátrixát az

$$A^T = \mu^T_{\mathbf{lpe}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.6 \\ 1 \end{bmatrix} \qquad B = \mu_{\mathbf{pcs}} = \begin{bmatrix} 0 & 0.2 & 1 \end{bmatrix}$$

fuzzy halmazokra. A fuzzy implikáció definíciója (6.2) szerint az $\mathbf{lpe} \longrightarrow \mathbf{pcs}$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.6 \\ 1 \end{bmatrix} \times \begin{bmatrix} 0 & 0.2 & 1 \end{bmatrix} = \begin{bmatrix} 0 \wedge 0 & 0 \wedge 0.2 & 0 \wedge 1 \\ 0 \wedge 0 & 0 \wedge 0.2 & 0 \wedge 1 \\ 0 \wedge 0 & 0 \wedge 0.2 & 0 \wedge 1 \\ 0.6 \wedge 0 & 0.6 \wedge 0.2 & 0.6 \wedge 1 \\ 1 \wedge 0 & 1 \wedge 0.2 & 1 \wedge 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0.2 & 0.6 \\ 0 & 0.2 & 1 \end{bmatrix} \tag{6.3}$$

6.2.2. Fuzzy szabályok és a fuzzy következtetés

A fenti relációk, a max-min kompozíció és a fuzzy implikáció felhasználásával a közönséges szabályrendszereknél megismert következtetés fuzzy megfelelőjét definiálhatjuk.

Fuzzy szabályok Az előzőekben definiált $A_i \longrightarrow A_j$ fuzzy implikáció tekinthető egy $\mathbf{if}\ A_i\ \mathbf{then}\ A_j$ szabálynak is adott A_i és A_j fuzzy halmazok között, ahol az A_i az U_i, az A_j pedig az U_j univerzum fölötti fuzzy halmaz. Diszkrét univerzumok esetén egy fuzzy szabályt egy megfelelő méretű $\mathbf{R}_{A_i \to A_j} = W_{A_i \to A_j}$ reláció-mátrixszal írhatunk le.

A fentiek alapján egy fuzzy szabályrendszer adott $U_1, ..., U_n$ univerzumok fölött egy $W_{A_i \to A_j}$ reláció-mátrix halmaz.

Fuzzy következtetés A következtetés az *általánosított modus ponens* fuzzy változatával

$$\frac{A',\ \ A \longrightarrow B}{B'}$$

végezhető el, ahol A és A' az U_A, B és B' pedig az U_B univerzum fölötti fuzzy halmazok. Szavakban ez azt jelenti, hogy az adott A' fuzzy halmaz és $A \longrightarrow B$ fuzzy szabály felhasználásával a B' következmény fuzzy halmaz adódik a következtetés eredményeképpen.

Fontos megjegyezni, hogy az eredeti modus ponens esetén, azaz amikor A és A' logikai értékkel bíró predikátumok, akkor csak $A = A'$ esetén adódik a $B = B'$ következtetés a matematikai logika implikáció műveletének felhasználásával. A fuzzy következtetés lehetővé teszi következmény kiszámítását az $A \neq A'$ esetben is.

A fuzzy következtetéshez tehát adottnak tekintjük a $A \longrightarrow B$ fuzzy implikációt, mint szabályt, és az A' fuzzy halmazt. A B' eredmény fuzzy halmazt a

$$B' = A' \circ R_{A \to B} \tag{6.4}$$

művelettel kapjuk, ahol a $\circ$ az (6.1) egyenlet szerinti kompozíció, amelyet az A' fuzzy halmaz és az $R_{A \to B}$ fuzzy reláció-mátrix oszlopai, mint fuzzy halmazok között végzünk el.

Példa: Fuzzy következtetés Alkalmazzuk a $\mathbf{R}_{\mathbf{lpe} \to \mathbf{pcs}}$ fuzzy szabályt (lásd (6.3) egyenlet) az alábbi A' fuzzy halmazra.

$$\mathbf{R}_{\mathbf{lpe} \to \mathbf{pcs}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0.2 & 0.6 \\ 0 & 0.2 & 1 \end{bmatrix}$$

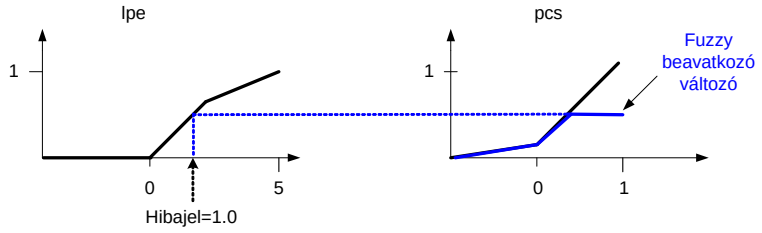
$$A' = \textbf{somewhat lpe} = \begin{bmatrix}0 & 0 & 0 & 0.6 & 1\end{bmatrix}^{\frac{1}{2}} = \begin{bmatrix}0 & 0 & 0 & 0.77 & 1\end{bmatrix}$$

Ekkor a következtetés (6.4) egyenletbeli definíciójából kapjuk, hogy

$$\begin{bmatrix}0 & 0 & 0 & 0.77 & 1\end{bmatrix} \circ \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0.2 & 0.6\\ 0 & 0.2 & 1\end{bmatrix} = \begin{bmatrix}0 & 0.2 & 1\end{bmatrix}$$

Példa: Fuzzy következtetés szemléltetése Az alábbi 6.3 ábrán szemléltetjük a fuzzy következtetést a fuzzy implikációs példában bemutatott $\textbf{lpe} \longrightarrow \textbf{pcs}$ fuzzy szabály és az U_e halmazból vett $e = 1$ éles hibajelre. Megfigyelhető, hogy a következtetés az éles hibajel

6.3. ábra. Következtetés éles feltételi értékkel



tagsági értékének megfelelő értéknél „levágja” a fuzzy szabály következményében szereplő fuzzy halmazt, és ez lesz a következtetés eredményeképpen kapott fuzzy bemeneti változó értéke.

6.3. Fuzzy alapú intelligens irányítórendszerek

A fuzzy alapú intelligens irányítórendszerek tartalmaznak egy beépített fuzzy szakértői rendszert (lásd például [9]), amely beépített szolgáltatásokkal segíti a megvalósítást. Egy komplett, fuzzy szabályokon alapuló tartály szintszabályozási példa megtalálható a Matlab Fuzzy Logic Toolbox felhasználói leírásában a

http://www.mathworks.com/help/toolbox/fuzzy/fp13935.html

webcímen.

Ebben az alfejezetben ezekről a beépített szolgáltatásokról, működésükről és hangolásukról lesz szó.

Egy fuzzy szakértői rendszerben alábbi fontos komponenseket különíthetjük el.

- **Előfeldolgozó egység (pre-processing unit)**: amely az adatokat fuzzy halmazokká alakítja,
- **Fuzzy szabálybázis** az univerzumokkal és tagsági függvényekkel, valamint a fuzzy szabályokkal,
- **Defuzzifikáló egység**: a fuzzy halmazokat konvertálja vissza szokásos „éles” adatokká

6.3.1. Fuzzy szakértői rendszerek hangolása és működése

A fuzzy szakértői rendszerek használatakor számos szolgáltatás és komponens hangolása igényel fokozott figyelmet.

Az univerzumok és tagsági függvények megválasztása Az *univerzumok* általában*szabványosítva* vannak az összes változóra: a $[-1, 1]$ vagy a $[-100, 100]$ valós intervallumok használhatók. Így az eredetileg adott változókat centrálni és normalizálni kell a szakértői rendszerbe történő adatbevitel előtt.

A *tagsági függvényeket* 3-5 különböző, nyelvi leírással (pl. **small**, **medium** és **large**) lehet kiválasztani minden fuzzy halmazhoz, ezek utána nyelvi módosítókkal (lásd 6.1.1 alfejezet) átalakíthatók. A *tagsági függvények formája* adott függvénycsaládokból választható. *Diszkrét fuzzy halmazokat* fuzzy szingletonokkal adhatunk meg, intervallum típusú univerzumokra pedig *trianguláris*, azaz lineáris szakaszokból álló folytonos függvénycsaládból választhatunk. Utóbbi esetben fontos, hogy a választott tagsági függvények a teljes univerzumot lefedjék, azaz ne legyen olyan $x \in U$, amelyre valamennyi tagsági függvény értéke nulla.

A fuzzy következtetés Fontos tudni, hogy a fuzzy szakértői rendszerekben a *fuzzy szabálybázis* formálisan az összes fuzzy szabály konjunkciója, azaz

$$\mathcal{R} = \bigvee R_i$$

Ha ezután egy adott A' fuzzy halmazra, mint *input*ra szeretnénk elvégezni a fuzzy következtetést, akkor ezt a *fuzzy következtető gép* az alábbi lépésekben végzi el:

1. *valamennyi (alkalmazható) fuzzy szabályt* alkalmazza az A' fuzzy halmazra, azaz képezi a $B_i = A' \circ R_i$ eredményhalmazokat,

2. a *következtetési eredményt* a $B' = \vee B_i$ formulával számítja ki, amihez a *fuzzy* ***or*** műveletet használja.

6.3.2. Fuzzy szabálybázis ellenőrzése

A közönséges szabálybázisok ellenőrzéséhez hasonlóan, a fuzzy esetben is igen fontos, hogy a fuzzy szabálybázist feltöltés után, de még használat előtt ellenőrizzük. Az ellenőrzéshez a fuzzy esetben szükség van egy „*hasonlósági mértékre*", amely két adott fuzzy halmaz „átlapolódását" méri. Ez a legegyszerűbb esetben, azonos diszkrét univerzum feletti A és B fuzzy halmazokra lehet a $d(A, B) = ||\mu_{A \wedge B}||$ vektor norma, amely a két halmaz $A \wedge B$ fuzzy metszete tagsági függvényének normája.

A fuzzy szabálybázis ellenőrzése során vizsgálandó legfontosabb tulajdonságok az alábbiak:

- *Teljesség*: annak ellenőrzésére, hogy bármely nem azonosan nulla tagsági függvényű bemenő fuzzy halmazra nem azonosan nulla tagsági függvényű következtetési eredmény adódik-e,

- *Konzisztencia*: annak vizsgálata, hogy két vagy több „nagyon hasonló”, vagy azonos feltételi részű szabály különböző következtetési eredményt generál-e,
- *Redundancia*: annak eldöntésére, hogy van-e két vagy több olyan szabály, amelynek a feltételi és a következmény része is „nagyon hasonló”,
- *Függőségi vizsgálatok* a szabályrendszert alkotó szabályok között.

6.3.3. Defuzzifikálás

A defuzzifikálás művelete egy adott fuzzy halmazból előállít egy valós „éles” értéket. Ez a művelet a fuzzy halmaz jellemzőtől és a valós defuzzifikált érték felhasználási céljától függően különböző lehet, ennek megfelelően különböző *defuzzifikálási eljárások* használatosak. A legfontosabbakat az alábbiakban foglaljuk össze.

- A *maximumok átlaga (mean of maxima)*, ezt az adott fuzzy halmaz x_{mj} maximális tagsági függvény értékkel rendelkező pontjaiból az

$$u = \frac{\sum_{j=1}^{l} x_{mj}}{l}$$

  összefüggéssel számoljuk.

- A *területi középpont (centre of area)* módszer diszkrét univerzumok feletti tagsági függvényekre a következőképpen számolható:

$$u = \frac{\sum_{j=1}^{l} \mu(x_{mj}) \cdot x_{mj}}{\sum_{j=1}^{l} \mu(x_{mj})}$$

  Fontos tudni, hogy több lokális maximummal rendelkező tagsági függvény esetén mindkét előző módszer eredményezhet olyan defuzzifikált értéket, amelynél távolról sem maximális a tagsági függvény értéke.

- A legegyszerűbb, és irányítási alkalmazásokhoz a legmegfelelőbb módszer a fuzzy univerzumból a tagsági függvény *maximális értékének megfelelő érték* kiválasztása. Ha ilyenből több is van, akkor az egyiket, mondjuk a legkisebbet választjuk defuzzifikált értéknek.

7. fejezet

A G2 keretrendszer áttekintése

A Gensym cég G2 nevű terméke [4] egy objektum-orientált, grafikus környezettel rendelkező szakértői keretrendszer, amely a folytonos és intelligens monitorozást, diagnosztizálást és irányítást igénylő valós-idejű alkalmazások fejlesztését támogatja. Számos konkrét alkalmazása közül megemlíthető például a NASA űrhajók működésének monitorozása, az Ericsson telekommunikációs hálózatán működő problémamegoldó rendszer, valamint a Toyota autógyártó rendszerének koordinálása.

Ebben a fejezetben egyszerű példák segítségével bemutatjuk a G2 alapvető szolgáltatásait és legfontosabb jellemzőit.

7.1. A G2 legfontosabb jellemzői

A G2 strukturált, természetes nyelven definiált objektumok, szabályok, függvények és eljárások használatát teszi lehetővé, amely az alkalmazások könnyű megértését, tesztelését és módosítását biztosítja. Lehetővé teszi továbbá a modellekben szereplő tudás dinamikus működtetését, azaz következtetések levonását, javaslatok adását és akciók végrehajtását valós időben vagy szimulált valós időben. A G2 egyidejűleg több szálon futó következtetéseket és nagyszámú adat analizálását is képes kezelni.

A G2 fő erőssége abban rejlik, hogy hatékony szolgáltatást nyújt a következő feladatok megoldásában:

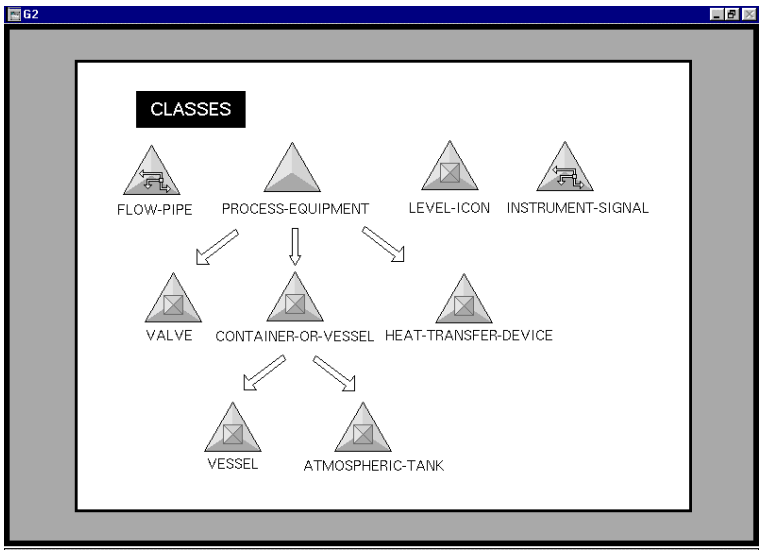
- intelligens döntéstámogatás,
- szabály alapú következtetés,
- korszerű irányítás és optimalizálás,
- dinamikus ütemezés,
- rendellenes események észlelése.

7.2. Tudásreprezentáció G2-ben

A G2 tudásbázisának elkészítéséhez egy jól-strukturált, magas-szintű, grafikus fejlesztői környezetet tartalmaz számos előre definiált tudáselemmel (items). Ezek a tudáselemek különböző típusúak lehetnek, beleértve a hagyományos programozásban használt függvényeket és eljárásokat, az objektum-orientált programozásban használt osztályokat és példányokat, a tudásalapú rendszerek szabályait, s egyéb speciális elemeket.

7.2.1. Objektumok

Egy G2 alkalmazás készítésének első lépéseként a fejlesztő a konkrét feladat leírásához szükséges egységeket (például tartály, szelepek, kapcsolók) definiálja G2 *objektum*ok formájában. A G2 objektumok is rendelkeznek az objektum-orientált rendszerekre jellemző tulajdonságokkal, így osztályhierarchiába rendezettek (lásd 7.1. ábra), tulajdonságaik öröklődnek, példányaik hozhatók létre, amelyek önálló életet élnek.

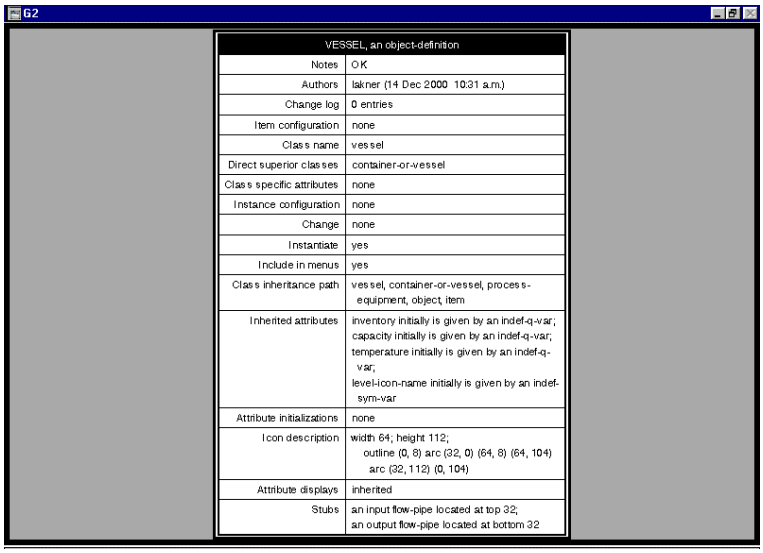


7.1. ábra. Objektumok hierarchiája

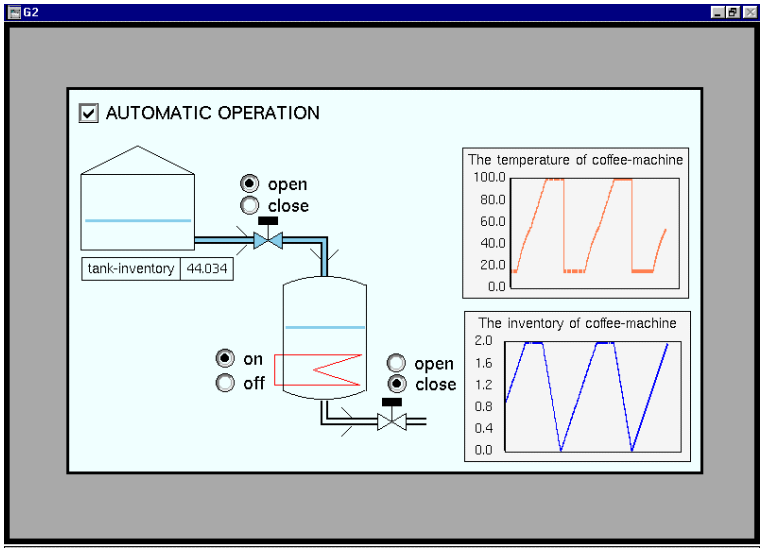
Minden tudáselem, így az objektum is rendelkezik *tulajdonságtábl*aval (lásd 7.2. ábra), amelyben a lényeges jellemzők definiálására van lehetőség. Itt a fejlesztő megad(hat)ja többek között az objektum nevét (*class name*), közvetlen szülő osztálya(i)nak nevét (*direct superior classes*), osztály-specifikus tulajdonsága(i)t (*class specific attributes*), lehetséges kapcsolódási pontjait (*stubs*), valamint megtervezheti grafikus megjelenítését (*icon description*).

Az objektumok definiálása után a fejlesztő a konkrét objektum *példány*okat manuálisan hozza létre egy vagy több munkaterületre való elhelyezésükkel, tulajdonságaik és kapcsolataik konkrét megadásával. Ennek eredménye egy sematikus diagram, amelyre példa a 7.3 ábrán látható vízmelegítő rendszer (amelynek egyik elemeként tekinthető a 4. fejezetben bemutatott kávéfőzőgép) G2 folyamatábrája.

A reszdszer főbb objektumainak tulajdonságtáblát mutatja be a 7.4. ábra.



7.2. ábra. Objektumok tulajdonságtáblája

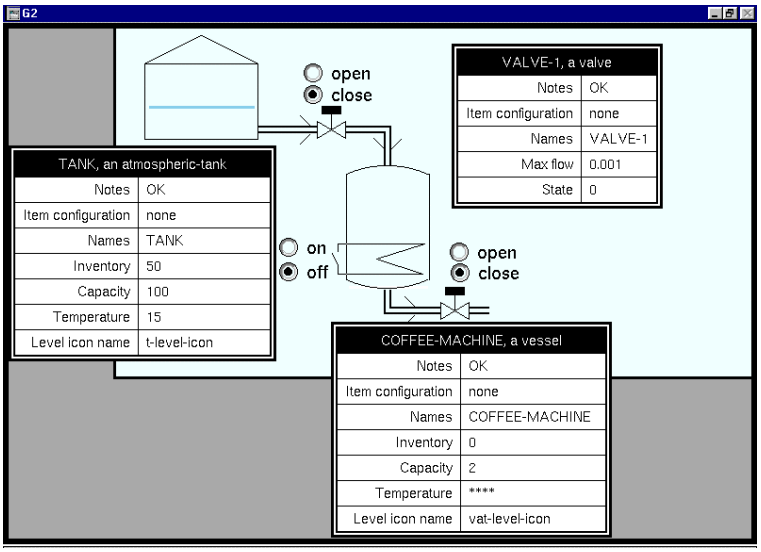


7.3. ábra. Egy egyszerű sematikus diagram

7.2.2. Változók, paraméterek

A *változó*k és *paraméter*ek beépített objektum osztályok, amelyek segítségével időben változó értékeket (pl. hőmérséklet, szint, kapcsoló állapota) reprezentálhatunk. Ez a két objektumtípus számos szempontból hasonló: attribútumokkal rendelkeznek, osztály hierarchiába rendezhetők, saját ikon rendelhető hozzájuk, valamint az időbeli viselkedésük leírásához történet megőrzési specifikációt (*history keeping spec*) definiálhatunk.

A fő különbség e két tudáselem között, hogy egy változó értéke megszűnhet úgy, hogy



7.4. ábra. Egy egyszerű sematikus diagram főbb objektumainak attribútum táblái

egy adott pillanatban nincs értéke, míg egy paraméter mindig rendelkezik értékkel. Az érvényességi tartomány (*validity interval*) attribútum definiálja, hogy a változó utolsó rögzített értéke meddig érvényes. Amennyiben egy változó értékére szükség van, azt a változó adatszolgáltatója vagy adatforrása határozza meg, amely lehet G2 szimulátor, G2 következtető gép (visszafelé haladó következtetéssel) vagy külső adatforrás. A változók értékeik kiszámításához rendelkezhetnek formulákkal és/vagy szimulációs formulákkal. Mivel egy paraméternek mindig van értéke, kiinduláskor kezdeti értékének megadására van szükség, amely következtetések vagy eljárások végrehajtása révén változhat meg.

7.2.3. Munkaterületek

A *munkaterület*ek téglalap alakú területek, amelyek egységeket (objektum, kapcsolat, szabály, eljárás, stb.) tartalmazhatnak. Egy tudásbázis elemeit tetszőleges számú munkaterületre helyezhetjük el, amelyeket a tudás rendszerezéséhez különböző hierarchiaszinteken definiálhatunk. Lehetőség van a munkaterületek aktiválására és deaktiválására, amely az adott munkaterületen elhelyezett egységek használatát engedélyezi, illetve tiltja a rendszer működtetése során. Az állandó munkaterületek mellett ideiglenes munkaterületek is létrehozhatók, amelyek csak a rendszer működése során léteznek, s nem mentődnek el a tudásbázissal.

7.2.4. Kapcsolatok, relációk

Egy sematikus diagramon két objektum összekötését (pl. csővezetékkel, elektromos vezetékkel) *kapcsolat*nak nevezzük. Ezt a G2 fejlesztő manuálisan, az objektumok grafikus összekapcsolásával adhatja meg, amely lehetővé teszi az objektumokra történő hivatkozást kapcsolataik alapján (pl. *„any tank connected to any valve”*), valamint generikus szabályok használatát.

A *relációk* a kapcsolatokhoz hasonlóan objektumok összeköttetésére használhatók, azonban ezek dinamikusan létrehozott objektum-társítások, amelyek grafikus reprezentációval nem rendelkeznek, s nem részei a tudásbázisnak (nem mentődnek el).

7.2.5. Szabályok

A szakértői tudás reprezentálása a G2-ben *szabály*ok segítségével történik. A G2 szabályok a 3. fejezetben bemutatott szerkezettel rendelkeznek, azaz feltételi részt és következmény részt tartalmaznak.

A működés szempontjából az alábbi szabály-típusok különböztethetők meg:

- „if” szabályok: közönséges szabályok

  for any valve V
  if *the state of V = 1*
  then *change the center stripe-color of every flow-pipe connected to V to sky-blue*

- „when” szabályok: az „if” szabályokhoz hasonlóak, azonban a G2 ezeket előre- és visszafelé haladó következtetés során nem veszi figyelembe

  for any container-or-vessel CV
  when *the value of the inventory of CV = 0*
  then *conclude that the temperature of CV has no value*

- „initial” szabályok: a tudásbázis indításakor meghívandó szabályok

  initially *for any container-or-vessel CV*
  if *the inventory of CV > 0*
  then *conclude that the temperature of CV = 15*

- „unconditional” szabályok: feltételi rész nélküli szabályok

  initially for any valve V
  unconditionally *conclude that the state of V = 0*

- „whenever” szabályok: eseményvezérelt szabályok

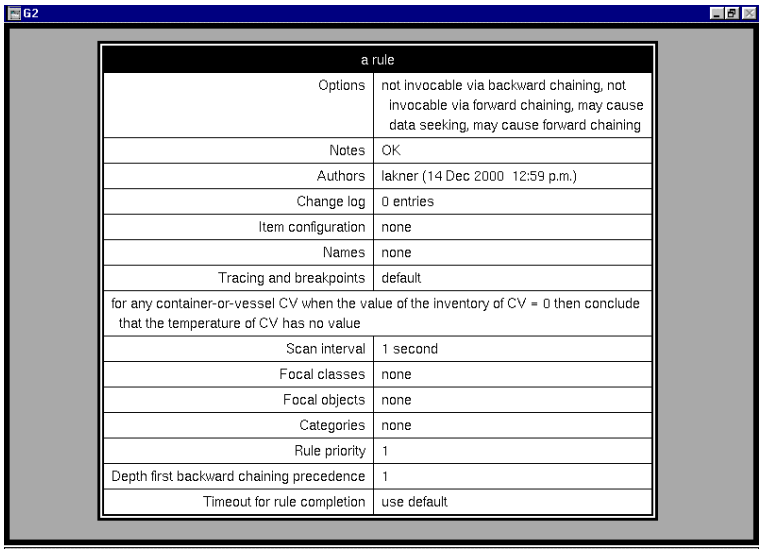
  whenever *auto-manual-state receives a value and when the value of auto-manual-state is auto*
  then *start auto()*

A fenti példákban *for* szócskát tartalmazó szabályok *generikus szabály*ok, amelyek egy adott objektum osztályra alkalmazhatók.

A szabályok a 7.5 ábrán látható attribútumokkal rendelkeznek, amelyek közül a legfontosabbak a következők:

- „options”: a szabályok használatának módját tartalmazza,

- „scan interval”: a szabály meghívásának gyakoriságát adja meg,



7.5. ábra. Szabály attribútum táblázata

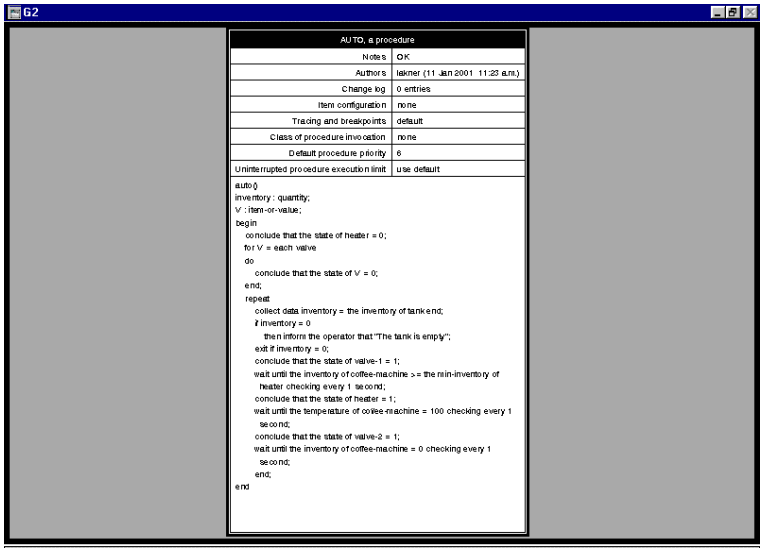
- „focal object”, „focal classes”: a szabályhoz kapcsolódó objektumokat, osztályokat jelöli ki,
- „rule priority”: a szabály fontosságát definiálja,
- „depth-first backward chaining precedence”: a szabályok vizsgálatának sorrendjét adja meg visszafelé haladó következtetés során,
- „timeout for rule completion”: a szabály feltételi részének kiértékelési idejét korlátozza.

7.2.6. Eljárások

Az *eljárás* a G2 által végrehajtandó művelet- és utasítás-sorozat, amely a magas-szintű programnyelvekhez hasonlít. A G2 procedurális nyelve tartalmazza az alapvető programstruktúrákat (pl. feltételes utasítások, iterációk, ...), ezen kívül számos szolgáltatással (pl. „do-in-parallel”) rendelkezik a valós idejű programozás biztosítására.

Az eljárások a következő három fő részből épülnek fel:

- eljárás feje, amely az eljárás nevét, argumentumait és visszatérési értékét definiálja,
- lokális deklarációs rész, amely lokális változókat és ezek típusait valamint kezdeti értékeit jelöli ki,
- eljárás törzs, amely az eljárás utasításait tartalmazza.



7.6. ábra. Eljárás attribútum táblázata

7.2.7. Függvények

A *függvény* előre definiált, visszatérési értékkel rendelkező, névvel ellátott műveletsorozat, amit kifejezésekben hívhatunk meg nevének és szükséges argumentumainak megadásával. A G2 számos beépített függvényt tartalmaz (például sqrt(x), max(x,y,z), abs(x)), emellett lehetőség van felhasználó által definiált függvények létrehozására, valamint idegen nyelvű függvény interfész (pl. C, Fortran) használatára.

7.3. Következtetés és szimuláció G2-ben

7.3.1. Valós-idejű következtető gép

A G2 legfőbb erőssége a *valós-idejű következtető gép*, amely egy alkalmazás aktuális állapota alapján következtet, emellett kommunikál a végfelhasználóval és következtetései alapján egyéb tevékenységeket kezdeményez.

A következtető gép működését a következő információforrások alapján végzi:

- tudásbázis tartalma,
- szimulált értékek,
- szenzoroktól és egyéb külső adatforrásoktól származó értékek.

Következtető gép az alábbi képességekkel rendelkezik:

- a szabályok „scan interval” attribútumában szereplő rendszeres időközönként *meghívja a szabályokat*,

- *szabályokra fókuszálás* során a kulcs objektumokhoz tartozó összes szabályt meghívja,
- *adatkérést kezdeményez*, amennyiben egy változó értéke lejárt és értékére szükség van,
- a változó értékére várás esetén *„ébreszti" a szabályokat*, amennyiben a változó értéket kap,
- *visszafelé haladó következtetést végez*, amennyiben egy változó értékét más forrásból nem tudja meghatározni,
- *előrefelé haladó következtetést végez*, amennyiben egy szabály feltételi része igazzá válik.

A két utóbbi képesség az intelligens rendszerek következtető mechanizmusa alapján működik, míg a többi az állapotfigyelésen alapuló valós-idejű szolgáltatásokat biztosítja.

7.3.2. G2 szimulátor

A G2 *szimulátor* egy független szoftver egységként megjelenő speciális adatszolgáltató, amely egy alkalmazás komponenseinek modellezésére szolgál a változók és paraméterek szimulált értékekeinek formulák és eljárások segítségével történő meghatározásával. A szimulátor a következő fontos tulajdonságokkal rendelkezik:

- a G2 egyéb egységeihez szorosan kapcsolódik,
- algebrai, differencia és elsőrendű differenciálegyenleteket old meg,
- változókhoz specifikus szimulációs formulák, változó osztályokhoz generikus szimulációs formulák rendelhetők,
- párhuzamosan működhet más valós idejű folyamatokkal.

A szimulátor alkalmazásának célja lehet a tudásbázis tesztelése normál és hibás működés esetén, ritkán előforduló események vizsgálata a szimulációs idő felgyorsításával, állapotbecslés nehezen mérhető változók esetén, valamint egy rendszer működésének vizsgálata on-line alkalmazás előtt.

A G2 szimulátortól a következő kategóriájú változók kaphatnak értéket:

- algebrai egyenletekkel leírható *függő változó*k

  height * diameter * pi

- differenciaegyenletekkel meghatározott *diszkrét állapotváltozó*k

  state variable: next value = the inventory of tank – the max-flow of valve-1 * the state of valve-1, with initial value 100

- differenciálegyenletekkel megadott *folytonos állapotváltozó*k

  state variable: d/dt = – the max-flow of valve-1 * the state of valve-1, with initial value 100

Egy állapotváltozó értéke előző értékétől függ, ezért használatához kezdeti érték megadására van szükség.

7.4. Tudásbázis fejlesztés és hibamentesítés

7.4.1. Fejlesztői interfész

Egy szakértői rendszer készítése során a tudásmérnök fejlesztői interfészen keresztül lép kapcsolatba a tudásbázis kezelő-fejlesztői alrendszerhez. A G2 fejlesztői interfésze a következő fő tulajdonságokkal rendelkezik:

- angol-szerű szöveget használ a tudás leírására,
- szöveg editort tartalmaz szövegek beviteléhez és módosításához,
- grafikus reprezentációt biztosít az interpretáció és használhatóság megkönnyítésére,
- ikon editorral rendelkezik objektumok grafikus reprezentációjának elkészítéséhez,
- számos eszközt tartalmaz nagy és összetett tudásbázis építéséhez, módosításához, használatához,
- dokumentációs lehetőséget biztosít a tudásbázishoz,
- hibakereső és -javító szolgáltatásokat tartalmaz.

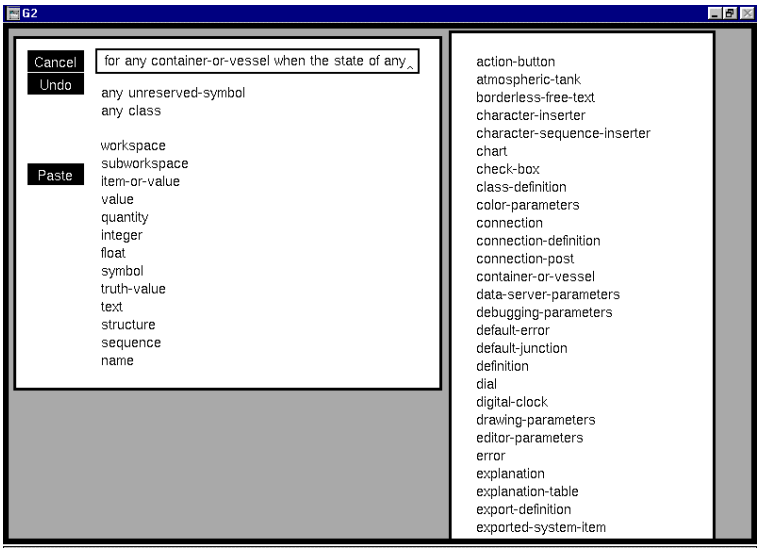
A G2 nyelvezete

A G2 strukturált, angol-szerű nyelvezettel rendelkezik, amelynek segítségével az egységekre többféleképpen is hivatkozhatunk:

- névvel (pl. coffee-machine),
- osztálynévvel (pl. the vessel),
- osztály egy példányaként, amely egy egységhez legközelebb esik (pl. the level-icon nearest to coffee-machine),
- osztály egy példányaként, amely egy másik egységhez kapcsolódik (pl. the valve connected at the output of coffee-machine),
- osztálynév előtt „for any” (pl. for any valve), amely generikus szabályok és formulák használatát teszi lehetővé (pl. initially for any valve V unconditionally conclude that the state of V = 0).

Interaktív szöveg editor

A szabályok, eljárások, függvények és egyéb szövegszerű állítások szerkesztése interaktív szöveg editor (lásd 7.7. ábra) segítségével történik. Ennek fő része a szöveg-editáló munkaterület, amely a beírt szöveg szerkesztését megkönnyítendően lehetséges opció listával és szintaktikai ellenőrző funkciókkal egészül ki. Hibás bevitel esetén a szöveg editor megjelöli és hibaüzenettel látja el az inkorrekt szöveget, valamint javaslatot ad ennek javításához.



7.7. ábra. Interaktív szöveg editor

Grafikus reprezentáció

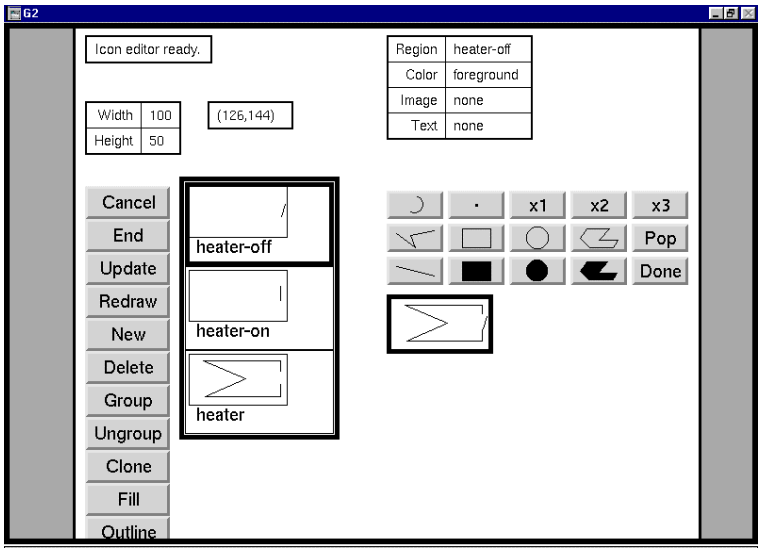
Egy alkalmazás építése a grafikus modell elkészítésével kezdődik, amelynek során az objektumokat ikonokkal reprezentáljuk, ezeket munkaterületekre helyezzük és összekapcsoljuk. Ennek eredménye az alkalmazás sematikus ábrája.

Az egységekhez tartozik egy úgynevezett pop-up menü, amely tartalmazza mindazokat a műveleteket, amelyeket a fejlesztő végezhet (pl. törlés, méret- és színváltoztatás, mozgatás), valamint a tulajdonságok definiálásához és megváltoztatásához használható attribútum táblát.

Interaktív ikon editor

A grafikus reprezentáció elkészítését interaktív ikon editor segíti, amely rétegek, régiók megszerkesztésével készített leírást a G2 nyelvezetére konvertálja. Az ikon editor főbb részei a 7.8 ábrán látható módon a következők:

- ikon megjelenítő,
- grafikus gombok,
- ikonméret jelző,
- kurzor-helyzet jelző,
- réteg megjelenítő.



7.8. ábra. Interaktív ikon editor

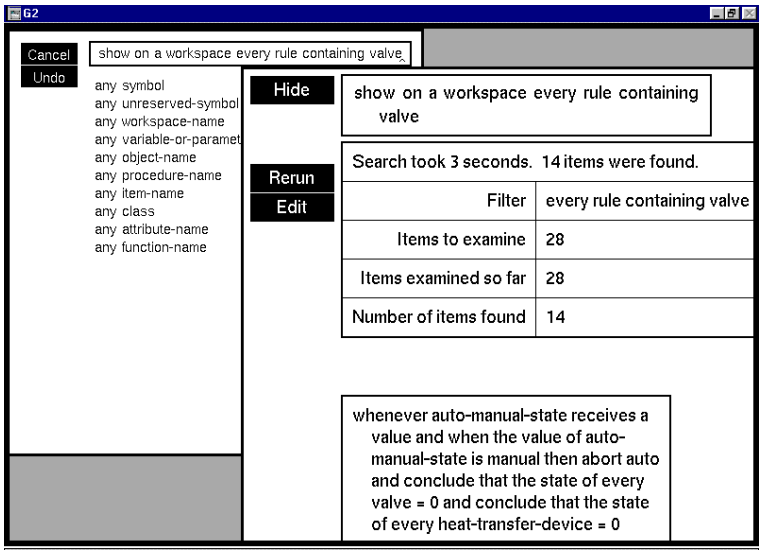
Tudásbázis kezelő eszközök

A nagy és/vagy összetett tudásbázis elkészítéséhez, módosításához és működtetéséhez a G2 számos tudásbázis kezelő eszközzel rendelkezik:

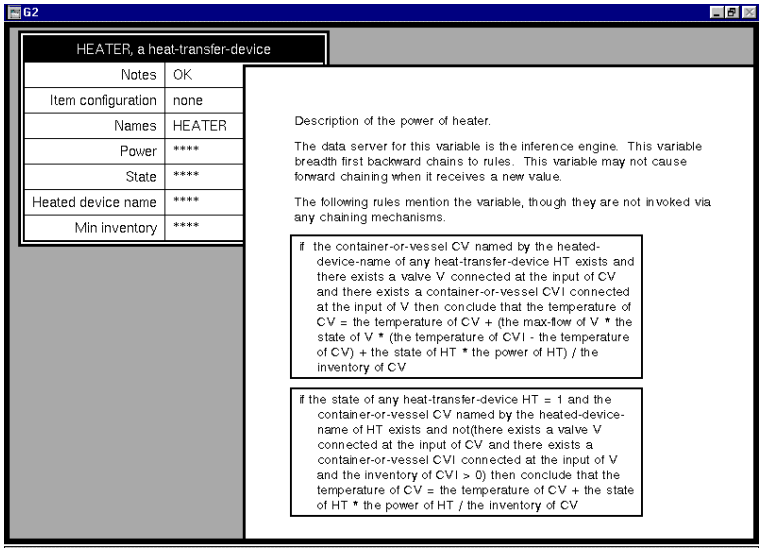
- Az *objektumok, állítások származtatása* (clone) segítségével a hasonló egységek hatékony létrehozását lehet elvégezni.
- A *műveletek végzése objektum-csoporton* kiiktatja a többszörözött műveleteket.
- A *tudásbázis megvizsgálása* (inspect) lehetővé teszi a tudáselemek gyors elérését nagyméretű tudásbázisban is (lásd 7.9. ábra).
- A *változók leírása* specifikálja a változóhoz tartozó adatszolgáltatót és azokat a szabályokat, amelyek segítségével a változó értéket kaphat (lásd 7.10. ábra).
- A *tudásbázis hierarchikus szervezése* megkönnyíti a tudásbázis használatát és vizsgálatát.
- A *tudásbázisok összeépítése* (merge) lehetővé teszi tudáskönyvtár(ak) elkészítését, valamint több részrendszer egymástól független fejlesztését, majd ezek összekapcsolását.

Dokumentáció a tudásbázisban

A tudásbázis dokumentálásához a munkaterületekre szövegek (free texts) helyezhetők. Ezek nincsenek hatással a tudásbázisra, csak dokumentációs célokat szolgálnak. Ezen kívül a fejlesztő dokumentáció objektumokat definiálhat, amelyeknek almunkaterülete szövegablakokat tartalmazhat.



7.9. ábra. Tudásbázis megvizsgálása



7.10. ábra. Változók leírása

Nyomkövetés és hibamentesítés

A G2 dinamikus visszacsatolást szolgáltat a fejlesztőnek a szabályok meghívása, valamint formulák, függvények és eljárások végrehajtása esetén. A G2 nyomkövetési és hibamentesítési lehetőségei a következők:

- *figyelmeztető üzenetek* megjelenítése nem várt események és hibák esetén,
- *csapda üzenetek* megjelenítése, amelyek mutatják a

  - változók, kifejezések aktuális értékét,
  - változók, formulák, szabályok, eljárások, függvények kiértékelésének kezdési és befejezési idejét,
  - minden lépés végrehajtásának idejét,

- *megszakítási pontok* generálása, amelyeknél a G2 felfüggeszti működését,
- meghívott szabályok *élesebb megvilágítása*.

A G2 a különböző típusú felhasználóknak a tudásbázis eléréséhez és használatához különböző jogosultságokat rendelhet, amelyek a következők lehetnek:

- menü opciók korlátozása,
- egységek mozgatásának, összekapcsolásának, másolásának, stb. korlátozása,
- attribútum tábla részeihez való hozzáférés,
- attribútumok szerkesztésének korlátozása.

A jogosultságok alapján különböző felhasználói kategóriákat definiálhatunk, pl. operátor, adminisztrátor, fejlesztő.

7.4.2. Felhasználói interfész

A felhasználó és a G2 közötti kommunikációt számos eszköz biztosítja. A fejlesztői interfész bemutatása során megismert lehetőségeken kívül a végfelhasználóval való kommunikáció biztosítására alkalmas fontosabb eszközök a következők:

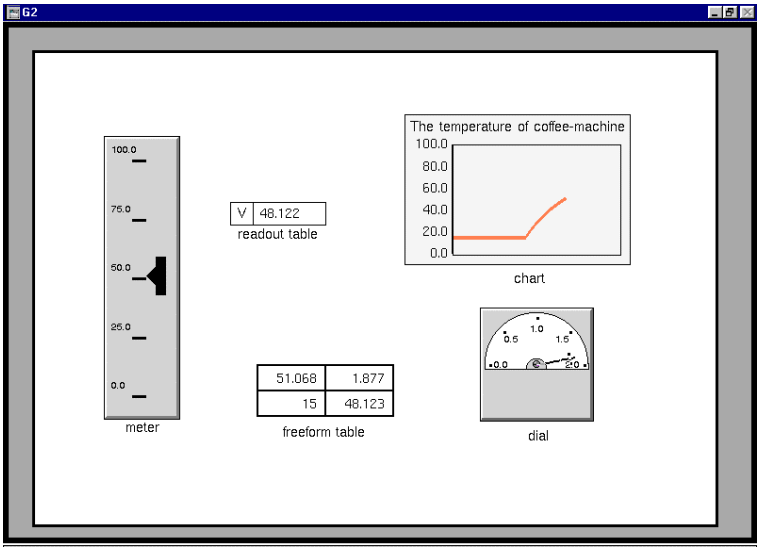
- *megjelenítő*k, amelyek változók, paraméterek, kifejezések értékeinek megjelenítésére használhatók,
- *végfelhasználói kontroll*ok, amelyek lehetővé teszik az alkalmazás irányítását a felhasználó által,
- *üzenetek, üzenet tábla, napló*, amely a felhasználóval való kommunikációt biztosítja.

Megjelenítők

A *megjelenítő*k a felhasználó számára lehetővé teszik a változók, paraméterek, kifejezések értékeinek megtekintését. A G2 ötféle megjelenítőt tartalmaz (lásd 7.11 ábrát):

- *kijelző* (readout table): doboz, amely egy változó, paraméter, kifejezés nevét és értékét mutatja,
- *grafikon* (graph): kétdimenziós megjelenítő, amely egy vagy több változó értékének időbeli sorozatát mutatja,

- *mérőóra* (meter): téglalap alakú megjelenítő, amely egy aritmetikai kifejezés értékét mutatja egy függőleges kijelzőn,
- *számlap* (dial): kör alakú skála, amely egy aritmetikai kifejezés értékét mutatja,
- *táblázat* (freeform table): változók értékét mutatja táblázatos formában.

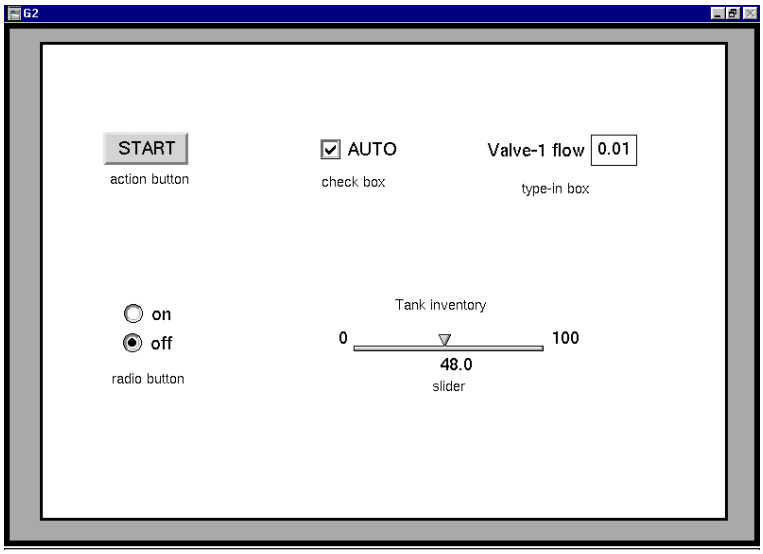


7.11. ábra. Megjelenítők

Végfelhasználói kontrollok

A *végfelhasználói kontroll*ok olyan eszközök, amelyek a felhasználónak biztosítanak lehetőséget a folyamatba történő beavatkozáshoz. Ahogy a 7.12 ábrán látható, a G2 ötféle végfelhasználói kontrollt tartalmaz:

- *akció gomb*ok (action buttons): lekerekített téglalap alakú dobozok a start, conclude, show, stb. akciók elindítására,
- *rádió gomb*ok (radio buttons): kis kör alakú egységek, amelyek segítségével változókhoz előre definiált értékek rendelhetők,
- *ellenőrző ablak* (check box): kis négyzet alakú doboz, amelynek segítségével egy változóhoz „on” és „off” érték jelölhető ki,
- *csúsztató* (slider): mindkét végén számmal rendelkező vízszintes egység, amely mutatója segítségével változóhoz numerikus értéket rendelhetünk,
- *editáló ablak* (type-in box): segítségével változóknak értéket adhatunk billentyűzetről.



7.12. ábra. Végfelhasználói kontrollok

Üzenetek, üzenettábla, napló

Az *üzenet* szöveges érték megjelenítésére alkalmas egység. A G2 üzeneteken keresztül informálja a felhasználót, amelyek az informáló (inform) akció eredményeképpen jelennek meg speciális munkaterületeken, amelyek *üzenettábla* vagy a *napló* lehetnek.

7.4.3. Külső interfészek

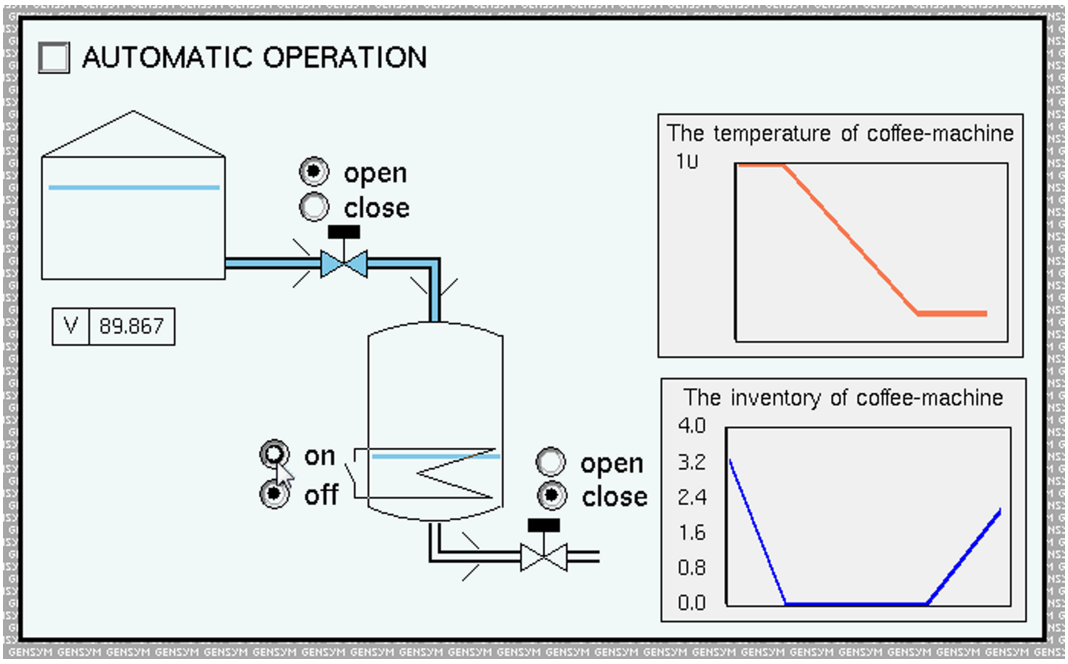
A G2 számos interfésszel rendelkezik, amelyek az egyéb folyamatokhoz való kapcsolódást és a külső forrásokból érkező adatok fogadását támogatják. Ezeket az interfészeket könnyű konfigurálni és mivel ezek automatikusan futnak a tudásbázis működése során, használatuk egyszerű. A G2 a következő külső interfészekkel rendelkezik:

- G2 Standard Interfész (GSI): G2 és külső folyamatok, rendszerek összekapcsolására,
- G2 Fájl Interfész (GFI): adatfájlok írására, olvasására,
- G2-G2 Interfész: két G2 kommunikációjára,
- Idegen Nyelvű Interfész: C, Fortran függvények használatára.

7.5. Egy egyszerű példa

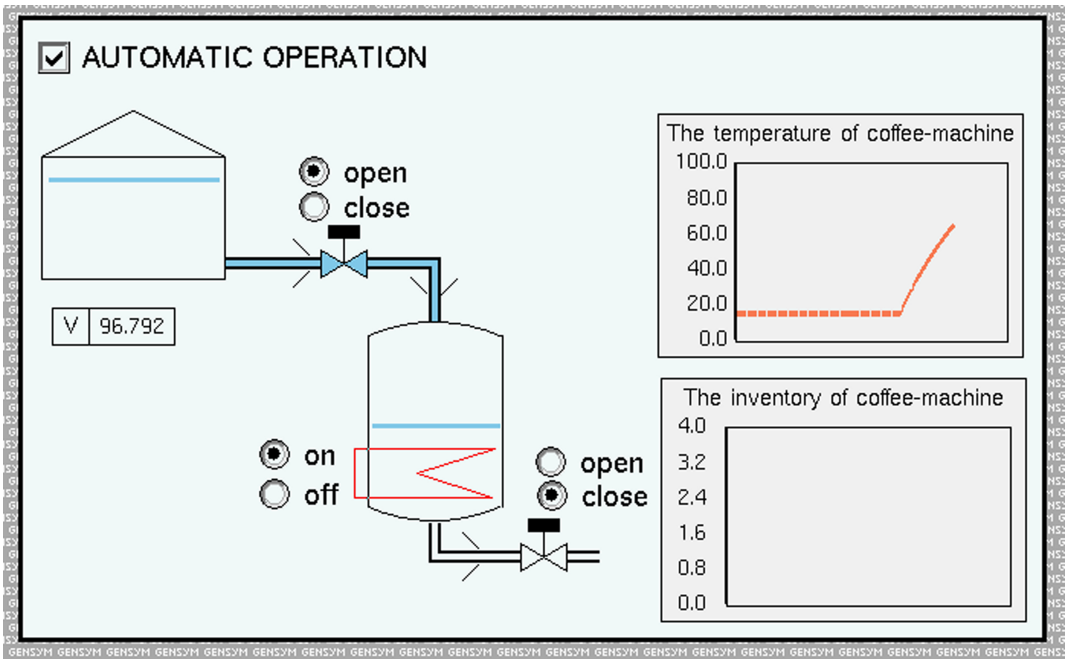
Tekintsük a 4 fejezetben bemutatott kávéfőzőgépet, amelyben egy elektromos, ki-bekapcsolható fűtőtest melegíti a vizet. A folyadékot egy tartályból engedhetjük be egy beömlő szelep segítségével, az elkészült forró vizet pedig egy kiömlő szelep kinyitásával engedhetjük ki.

7.13. ábra. `g2manual.avi`



A kávéfőző rendszer G2 folyamatábrája a 7.3. ábrán látható. A rendszer működését a beavatkozó szervek manuális kezelésével a „g2manual.avi”, automatikus üzemmódban pedig a „g2auto.avi” fájl futtatásával vizsgálhatjuk meg.

7.14. ábra. `g2auto.avi`



Irodalomjegyzék

[1] Faltings, B., Struss, P.: *Recent Advances in Qualitative Physics*, The MIT Press, Cambrdge, MA.(1992)

[2] Forbus, K. D.: Qualitative Process Theory, *Artificial Intelligence*, **24**, pp. 85–168. (1984)

[3] Futó, I.: *Mesterséges intelligencia*, Aula Kiadó, (1999)

[4] *G2 Reference Manual (for G2 Version 3.0)* Gensym Corporation, (1992)

[5] Hangos, K. M., Gerzson, M., Lakner, R., Gál, I.: *Intelligens irányító rendszerek*, Veszprémi Egyetemi Kiadó, Veszprém, pp. 1-119 (1995)

[6] Hangos, K. M., Lakner, R., Gerzson, M.: *Intelligent Control Systems: An Introduction with Examples*, Kluwer Academic Publisher, New York (2001)

[7] Jensen, K., Rozenberg, G.: *High-level Petri Nets: Theory and Applications*, Springer Verlag, London, (1991)

[8] Kuipers, B.: Qualitative Simulation. *Artificial Intelligence*, **29**, pp. 289–388. (1986)

[9] MATLAB Fuzzy Toolbox.

[10] Murata, T.: Petri Nets: Properties, Analysis and Applications, *Proc. of IEEE*, **77**, 4, (1989) DOI: 10.1109/5.24143

[11] Russel, S. J., Norvig, P.: *Mesterséges intelligencia - modern megközelítésben*, Panem-Prentice Hall, (2000)

[12] Weld, D. S., de Kleer, J. (Eds.): *Readings in Qualitative Reasoning about Physical Systems*, The Morgan Kaufman (1990)