Szabályalapú rendszerek – bevezetés
(BME MIT, Mészáros Tamás, Strausz György)

1 Szabályalapú rendszerek alapelvei

A szabályalapú rendszerek hatékony eszközt biztosítanak nehezen algoritmizálható problémák megoldásához. Előnyös tulajdonságai miatt elterjedten alkalmazzák számos problématerület problémáinak megoldására. A továbbiakban ismertetjük a szabályalapú rendszerek szerkezeti felépítését, rávilágítunk alkalmazásának előnyös tulajdonságaira.

A szabályalapú rendszerek szerkezetének, működésének leírása során számos fontos fogalom jelenik meg, amelyek pontos megértése, tisztázása elengedhetetlen a tématerület tárgyalásához. Ezért a következőkben ismertetjük a legfontosabb fogalmakat és jelenésüket.

1.1 A szabályalapú rendszerekhez kapcsolódó fontos fogalmak

Az alábbiakban a szabályalapú rendszerekhez kapcsolódó legfontosabb fogalmakat és jelentésüket részletezzük.

- *Előrefelé következtetés (forward chaining)*

  Az előrefelé következtetés során a szokásos IF-THEN szabályokban az IF után álló feltételrész illesztése történik, és amennyiben a szabály illeszkedik, akkor a THEN ágban megadott akciókat lehet a szabály elsütésével végrehajtani.

  Gyakran nem célok, hanem beérkező adatok vezérlik cselekedeteinket. Például ha nagy hőt érzünk a kezünk közelében, akkor elrántjuk. Ezt meg lehet fogalmazni célvezérelt viselkedésként is, de a természetes modellezési megközelítés ebben az esetben az előrefelé következtetés *felismer-cselekszik* működési ciklusa. Mint azt már korábban láttuk, előrefelé következtetésnél a szabályok bal oldalát illesztjük az állapotleírásra, vagyis a munkamemória pillanatnyi állapotára. Az illeszkedő szabályok jobb oldalai által tartalmazott következtetéseket beolvasztjuk az állapotleírásba, így létrejön az új állapot leírása.

  Előrefelé következtetés esetén a mintaillesztés bonyolultabb követelményeket támaszt a probléma-megoldóval szemben, mint hátrafelé következtetés esetében. Tipikusan, ha egy szabály elsüthető, mert a feltételrésze illeszkedik a munkamemória pillanatnyi tartalmára, akkor nagy valószínűséggel a következő ciklusban is elsüthető marad. A rendszernek olyan eszközt kell biztosítani, amivel a szabályelsütéseket nyilván lehet tartani és meg lehet akadályozni az ismételt szabályelsütést. Ennek nagyon lényeges hatékonysági vonzata van.

  Bár léteznek viszonylag egyszerű vezérlési stratégiák, a legtöbb előrefelé következtetést alkalmazó rendszerben (pl. OPS5, CLIPS) nagyon hatékony illesztési algoritmust valósítanak meg (lásd a RETE algoritmus leírását), ami majdhogy nem de facto szabvánnyá vált. Ezen felül számos olyan mechanizmust is biztosítanak, amivel szabályozni lehet a szabályok kiválasztását (tipikusan a konfliktus feloldást).

  Az előrefelé következtetést a fent leírtak alapján szokás adatvezérelt következtetésnek is nevezni, mert a következtetés beindulását a munkamemóriában megjelent új adat triggereli.

  Tipikus alkalmazási köre az adatgyűjtő/adatfeldolgozó rendszerek.

- *Hátrafelé következtetés*

  A hátrafelé következtetés során a szokásos IF-THEN szabályokban a THEN után álló következményrész illesztése történik a munkamemóriára, és amennyiben az illesztés sikeres, akkor a bizonyítandó részcélt sikerült bizonyítani. Az illesztés sikertelensége nem jelent ebben az esetben azonnali kudarcot: a feltételrészek bizonyíthatósága egyben a részcél bizonyíthatóságát is jelenti. Ezért ekkor a szabály jobb oldalán található részcél bizonyításához a rendszer felveszi bizonyítandó részcélnak a szabály bal oldalán található feltételeket és megpróbálja rajtuk keresztül bizonyítani az eredeti részcélt. Jól látszik az algoritmus rekurzív jellege.

  A hátrafelé következtetést szokás célvezérelt következtetésnek is nevezni, mert a következtetés során a bebizonyítandó/igazolandó (rész)célok alapján próbálunk meg szabályokat illeszteni.

  Tipikus alkalmazási köre a tételbizonyítás, diagnosztika.

  Megjegyzés:

  Bár a fenti tipikus alkalmazási körök dominánsak, a szabályok megfogalmazásával befolyásolni lehet, hogy melyik következtetési irány alkalmazása a célszerűbb.

- *A tudás tranzitív lezártja*

  A tudás tranzitív lezártja a munkamemória pillanatnyi tartalma és az abból a szabálybázis szabályai által lekövetkeztethető tudás együttese. Jól látszik, hogy a tudásbázis tranzitív lezártja tulajdonképpen nem más, mint a tudásbázisban az adott pillanatban expliciten (munkamemória tartalma) és impliciten (lekövetkeztethető) tartalmazott tudás összessége.

- *Magyarázat*

  A következtető rendszer által lekövetkeztetett tudáselem (tipikusan válasz egy adott problémára) előállításának megindokolása (milyen tényekből, milyen szabályok alkalmazásával állt elő).

- *Monoton produkciós rendszer*

  Egy produkciós rendszer *monoton*, ha egy szabály alkalmazása sohasem akadályozza meg egy másik szabály későbbi alkalmazását, amelyik az első szabály kiválasztásakor szintén alkalmazható volt.

- *Nem monoton produkciós rendszer*

  Egy produkciós rendszer nem monoton, ha a fent definiált monotonitás tulajdonság nem áll fenn.

- *Összetett és megközelítő illesztés*

  Számos esetben egyrészt a szabályokban nincs pontosan megfogalmazva az előfeltétel, másrészt a munkamemória tartalma által leírt adatelemek is tartalmazhatnak bizonytalanságot. Ilyenkor nagyon hasznos lehet, ha az illesztést nem "fekete-fehéren" tekintjük, vagyis az illesztés kimenetele nem csak illeszkedik/nem illeszkedik lehet, hanem az illeszkedéshez mértéket lehet rendelni. Ez a technika különösen a bizonytalanságkezelés kapcsán kap nagy teret.

A közelítő illesztés legelterjedtebb változata a fuzzy szabályalapú rendszerekben jelenik meg, ahol az illesztés fuzzy értelemben történik.

- *A hasznossági (utility) probléma*

  A mesterséges intelligencia alkalmazás számos területén, így a szabályalapú rendszereknél is jelentkezik az ún. hasznossági probléma. Amennyiben a szabálybázis tartalmaz úgynevezett metaszabályokat (szabályokat a szabályokról, illetve azok alkalmazásáról, amivel pl. a következtetés menetét lehet befolyásolni), akkor felmerül a kérdés, hogy egy adott probléma megoldása során mennyi meta tudást alkalmazzunk. A hasznossági probléma úgy merül fel, hogy egyébként a döntések megfontoltságát, optimalitását elősegítő metaszabályok alkalmazásával, illetve illesztésével olyan sok időt tölthetünk, hogy nem jut idő a probléma valós megoldására. Ez a jelenség több rendszerben is megfigyelhető volt: egy darabig metaszabályok hozzáadása javította a probléma megoldás minőségét, majd a problémamegoldó teljesítmény jelentősen leromlott. Ennek oka az volt, hogy a rendszer idejének nagy részét a metaszabályok illesztésével töltötte.

1.2 A szabályalapú rendszerek strukturális felépítése

A szabályalapú rendszerek az alábbi főbb komponensekből épülnek fel:

- *Következtetőgép*

  A következtetőgép a rendszer aktív komponense. A következtetést végzi. Ebben van megvalósítva:

  - a vezérlési stratégia, amely meghatározza, hogy a szabályok milyen sorrendben lesznek illesztve a tudásbázisra, illetve amennyiben több szabály is illeszthető egyszerre, akkor a konfliktus feloldási stratégiát is rögzíti.
  - egy szabály alkalmazó/elsütő
  - magyarázat generáló

    Mivel számos alkalmazásnál természetes igényként merül fel a következtetés menetének megindokolása, így egyre több következtetőgépnek részét képezi egy magyarázatgeneráló komponens is. Ennek fő oka, hogy ahhoz, hogy egy szakértői rendszer hatékony segédeszköz lehessen, a felhasználóknak egyszerűen képesnek kell lenni kapcsolatot tartani vele.

    Számos problématerületen a keretrendszer felhasználói önmagában a feladat megoldását nem fogadják el, mert bonyolult problémák megoldása során számos buktató lehet jelen, a felhasználó meg akar győződni, hogy logikailag helyes volt a levont következtetés, vagyis az alkalmazott lépések mind megfelelőek voltak. Ez kiváltképp igaz orvosi/orvosbiológiai alkalmazásoknál, ahol a döntés végső felelősségét az orvos viseli, még akkor is, ha a diagnózis felállításához jelentős segítséget kapott egy szakértői rendszertől. Ezért ilyen rendszerekben nagyon fontos, hogy az alkalmazott következtetési mechanizmus a felhasználó számára is érthető lépésekben dolgozzon, haladjon, és hogy elég meta-tudás (tudás, ismeret magáról a következtetési mechanizmusról) álljon rendelkezésre, így az egyes lépések logikai magyarázata, jelentése megadható.

    Ehhez fel lehet használni a nem monoton logikát megvalósító igazság-karbantartó rendszereket, de ezt a funkciót tölti be a szabályok azonosítóval

történő ellátása és a szabályalkalmazások naplózása is. A naplózott jelentésből a szakértő (orvos) ellenőrizheti a végső diagnózishoz vezető következtetési út helyességét.

- bizonytalanság kezelési mechanizmus

  Mivel az esetek túlnyomó többségében a megfigyelések zajosak, bizonytalanok, illetve maga a modell is bizonytalan, így a következtetőgépet fel kell vértezni a bizonytalanság kezelésére alkalmas mechanizmussal. Ezen eszközökről részletesebben lásd a bizonytalanság kezelésével foglalkozó fejezetet.

- *Munkamemória*

  Egy vagy több tudásbázis/adatbázis, amely az adott feladat megoldásához szükséges információkat, tudást tartalmazza. Tipikusan az adatbázis egy része állandó, statikus, más része viszont dinamikusan változik, csak az adott probléma megoldása során releváns.

  A munkamemória tárolja a következtetés pillanatnyi állapotát, vagyis hogy a probléma megoldás során az adott pillanatban a problématér mely állapotában vagyunk.

- *Szabálybázis*

  Szabályok egy halmaza, amelyek ún. bal oldalból és jobb oldalból épülnek fel. A bal oldal (minta) alapján lehet eldönteni, hogy a szabály alkalmazható-e, (Mint azt majd a későbbiekben láthatjuk, ez az értelmezés az előre láncoló szabályok esetében természetes. A hátrafelé láncoló rendszerekben gyakran megfordítják az oldalakat.), míg a jobb oldal azokat a műveleteket, tevékenységeket írja le, amiket a szabály alkalmazásakor kell végrehajtani.

  A szabálybázis tartalmazza az adott problémakör szabályokkal megfogalmazott modelljét. A modell alapján lehet következtetéseket végezni.

  Mivel a szakértői rendszerek ereje, használhatósága a tudásbázisukban tartalmazott tudás gazdagságában rejlik, ezért nagyon lényeges, hogy a tudásbázis a lehető legteljesebb és legpontosabb legyen az adott probléma területre nézve. Nagyon gyakran ez a tudás nem létezik explicit módon rögzítve, gyakran a szakértők fejében létezik. Ezen tudást a szakértőnek a programmal való együttműködésével lehet átadni a tudásbázisnak, illetve maga a program nyers adatokból tanulhat.

  Amennyiben a szakértő viszi be a szabályokat, célszerű ehhez egy egyszerű szabály-editort alkalmazni, ami biztosítja, hogy a szabályok struktúrája megfelel a rendszerben alkalmazott konvencióknak.

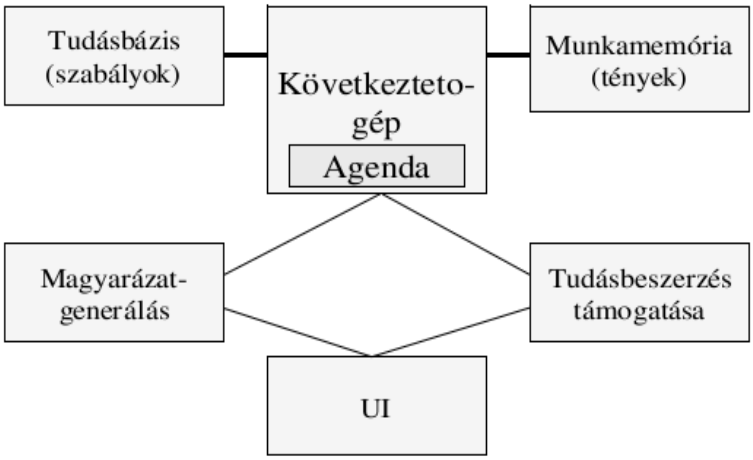
- *Felhasználói kezelői felület*

  A felhasználói felületen keresztül tud a felhasználó a következtetőgéppel kommunikálni.

A fenti komponensek kapcsolatát az 1. ábra mutatja. Mint azt az ábra jól mutatja, a következtetőgép központi szerepet tölt be ebben a struktúrában. A következtetőgép a kezelői felületen keresztül tart kapcsolatot a külvilággal, a felhasználóval. Amennyiben a működéshez szükség van megfigyelt tényeknek a felhasználó általi bevitelére, az ezen a felületen keresztül történhet meg. Másrészt a következtetés

végeredményét, illetve minden közbülső információt itt jelenít meg a következtetőgép. Bár manapság egyre elterjedtebben alkalmaznak grafikus felhasználói felületet, ezen funkciót egy egyszerű szövegorientált felület is tökéletesen betölti.

A következtetőgép kétirányú kapcsolatban áll a munkamemóriával is. A két nyíl jelentése: a következtetőgép egyrészt a szabályok illesztéséhez szükséges tényeket, információkat innen olvassa ki, másrészt a következtetés eredményeként előálló tényeket pedig ide írja.



1. ábra: A szabályalapú rendszerek strukturális felépítése

Jól látszik, hogy az ábrán a szabálybázissal már nem szimmetrikus a kapcsolat. Egy tipikus architektúrában a következtetőgép kiolvassa a szabályokat a szabálybázisból, de a szabálybázis a működési ciklus során statikus, az általa leírt modell nem változik. Bonyolultabb, tanulási képességekkel is felruházott szabályalapú rendszerben lehetőség van a szabálybázis működés közbeni módosítására. Ezt jelöli a szaggatott nyíl. Ez a következtetőgép általi új szabályok megalkotását, illetve a metaszabályokban lévő esetleges paraméterek hangolását is jelentheti.

Másrészt feltűnhet, hogy a szabálybázis két részre van osztva: ez egy funkcionális tagolást jelent. A szabálybázis, mint azt korábban láttuk, szabályok formájában tartalmazza az adott probléma terület modelljét. Megfogalmazhatunk azonban a következtetés menetét befolyásoló úgynevezett metaszabályokat is. Ezek általában probléma terület független szabályok, nem a modellt írják le, hanem a következtetőgép működését hangolják, nagyon gyakran heurisztikákat fogalmaznak meg. Célszerű ezeket a modellt leíró szabályoktól elkülönülten kezelni.

2 A szabályalapú rendszerek működési ciklusa

A szabályalapú rendszerek működése ciklikus tulajdonságot mutat, bizonyos jól definiált működési fázisok ciklikusan követik egymást. Ezek az alábbiak:

- Minta illesztés/szabály illesztés

  A mintaillesztési fázisban a következtetőgép valamilyen szisztematikus stratégia szerint a szabályokat megpróbálja a munkamemória tartalmára illeszteni, vagyis eldönteni, hogy egy adott szabály az adott pillanatban alkalmazható-e vagy sem.

- Konfliktus feloldás

  A mintaillesztési fázis eredményeképpen előáll a konfliktus halmaz, amely az összes, az adott pillanatban alkalmazható szabályt tartalmazza. A konfliktus feloldás feladata ezen halmazból egy alkalmazandó szabály kiválasztása.

- Szabály alkalmazás/elsütés

  A konfliktus halmazból a konfliktus feloldási fázis során kiválasztott szabály alkalmazása. Ennek hatására az adott szabály jobb oldalán szereplő akciókat a következtetőgép végrehajtja, aminek a hatására a munkamemória tartalma megváltozhat, vagyis a következtetőgép átléphet a problématér egy másik állapotába.

Mivel a vezérlési stratégia és azon belül a konfliktusfeloldási stratégia kitüntetett szerepet tölt be a következtetőgép működésében, így a továbbiakban részletesen áttekintjük a két stratégát.

2.1 Vezérlési stratégia

A vezérlési stratégia fontos szerepet játszik a konfliktusfeloldásban. Mivel a vezérlési stratégia meghatározza a szabályok illesztési sorrendjét, illetve a konfliktusfeloldást, vagyis az elsütendő szabály kiválasztását, így nyilvánvaló, hogy a döntésnek lényeges hatása van a probléma megoldására, vagy akár arra, hogy a probléma egyáltalán megoldható-e.

2.2 Követelmények egy jó vezérlési stratégiával szemben

Egy jó vezérlési stratégiával szemben az alábbi legfontosabb követelményeket támasztjuk:

- *Mozgást okozzon.*

  Azok a vezérlési stratégiák, amelyek nem okoznak mozgást, előre haladást, soha nem vezetnek el a megoldáshoz.

- *Legyen szisztematikus.*

  A vezérlési stratégia ne véletlenszerű választásokon alapuljon, legyen szisztematikus, ami adott esetben biztosítja majd a problématér teljes bejárását.

A vezérlési stratégiák tipikusan valamilyen keresési algoritmust használnak, amelyeknek a tulajdonságai meghatározzák magát a stratégiát is. Ezek közül a leggyakrabban alkalmazottak:

- mélységi keresés
- szélességi keresés
- best-first jellegű keresés

2.3 Konfliktusfeloldás

2.3.1 A konfliktusfeloldásra alkalmazott stratégiák

A konfliktusfeloldásra számos stratégiát lehet alkalmazni. Ezek általában valamilyen heurisztikán alapszanak. Az adott probléma terület hatással lehet az egyes heurisztikák hatékonyságára, így a megfelelő stratégiát a probléma terület alapos vizsgálata után célszerű kiválasztani.

Általánosságban igaz, hogy célszerű az "Egyszeri szabályelsütés" elvét érvényesíteni, azaz ugyanazon argumentumokkal minden szabályt csak egyszer süssünk el.

A legfontosabb konfliktusfeloldási stratégiák az alábbiak lehetnek:

- Szabályokon alapuló preferenciák

  A szabályokhoz preferenciát két módon szokásos hozzárendelni. A legegyszerűbb, hogy a preferenciát a szabályok felsorolási sorrendje tükrözi. Ekkor tulajdonképpen a modellalkotó feladata, hogy a szabályokat a preferenciáinak megfelelő sorrendben vigye be a tudásbázisba. A PROLOG is ezt a sémát alkalmazza.

  Egy másik szokásos módja a szabályokon alapuló preferenciának, hogy a speciális eseteket leíró szabályokat előnyben részesítjük az általánosabb szabályokkal szemben. Ezen speciális szabályok célja, hogy az emberi szakértők által alkalmazott tudást is meg lehessen fogalmazni, , amikor közvetlenül, keresés nélkül oldják meg a feladatot. Ilyenkor mivel a speciális szabályok növelnék a keresési teret, úgy szokták implementálni ezt a fajta preferenciát, hogy a keresés során a már meglévő szabályoknál általánosabb szabályokat már nem vizsgálja.

  A szabályillesztő az alábbi kritériumok alapján tudja eldönteni, hogy melyik szabály az általánosabb/speciálisabb:

  Ha egy szabály előfeltétel halmaza tartalmazza egy másik szabály összes előfeltételét és még legalább egy másik feltételt, akkor a második szabály általánosabb, mint az első.

  Ha egy szabály feltétel halmaza megegyezik egy másik szabály feltétel halmazával, de az első szabályban változók vannak megadva ott, ahol a második szabályban konstansok, akkor az első szabály általánosabb a másodiknál.

- Objektumokon alapuló preferencia

  A keresési mechanizmust tehermentesíteni lehet, ha az illesztéseket az illesztett objektumok fontosságán alapján sorrendezzük. Itt figyelembe lehet venni az illesztésben résztvevő objektumok frissítési idejét (mikor került be a munkamemóriába), illetve az objektumokhoz hozzá lehet rendelni valamilyen mértéket, ami az adott objektumnak a rendszer állapotában betöltött szerepét jellemzi.

- Az állapotokon alapuló preferencia

  Amennyiben a szabályok következmény része visszavonható, megoldható, hogy egy adott pillanatban alkalmazható összes szabályt időlegesen alkalmazzuk és megvizsgáljuk az alkalmazás utáni állapotokat. Ha valamilyen mérték alapján minősíteni lehet az állapotokat, ki lehet választani azt a szabályt, ami a legjobb állapotba vezet. Ez a megközelítés megegyezik a best-first keresési algoritmussal. Ezt a konfliktus feloldási stratégiát tipikusan keresés vezérlési mechanizmusként szokás kódolni.

2.4 Választás az előrefelé és a hátrafelé következtetés között

Produkciós rendszerekben az előrefelé és a hátrafelé következtetést szimmetrikus folyamatnak lehet tekinteni. Vegyük észre, hogy ugyanazokat a szabályokat lehet előrefelé és hátrafelé következtetésre is használni.

Számos tényező van hatással arra, hogy melyik következtetési irányt célszerű választani. Ezek közül a legfontosabbak:

- *Milyen állapotból van több: kiinduló állapotból vagy célállapotból.*

  A kisebb állapothalmazból célszerű a nagyobb halmaz felé haladni (amit valószínűleg könnyebb megtalálni).

- *Melyik irányban nagyobb az elágazási tényező (azon csomópontok átlagos száma, amiket egy csomópontból közvetlenül el lehet érni).*

  A kisebb elágazási tényezővel rendelkező irányba szeretnénk haladni, mert ez minimalizálná a feldolgozott csomópontok számát.

- Szükség van-e magyarázat generálásra, a megoldás indoklására?

  Amennyiben igen, célszerű a felhasználó gondolatmenetét leginkább tükröző irányt követni.

- *Milyen esemény fogja indítani a probléma megoldó lépést?*

  Amennyiben *új tény beérkezése*, akkor az előrefelé következtetés tűnik ésszerűnek. Amennyiben egy kérdés, amire választ kell előállítani, akkor a hátrafelé következtetés tűnik a természetesebb iránynak.

A két megoldás ötvözete is megvalósítható, vagyis a kiindulási állapotból előrefelé haladunk előrefelé következtetéssel, míg a célállapotból hátrafelé haladunk hátrafelé következtetéssel. A következtetés akkor ér célt, ha a két következtetési út metszi egymást. Ezt a technikát kétirányú keresésnek nevezik. Ez a technika akkor válik különösképp vonzóvá, ha a minden egyes lépésnél a csomópontok száma exponenciálisan növekszik. Fel kell hívni azonban a figyelmet, hogy a kétirányú keresés nagyon rossz hatékonyságúvá válhat, amennyiben lehetséges, hogy a következtetési utak divergáljanak, vagyis nem metszik egymást. Nagyon sok probléma területen nem lehet számszerűsíteni a következtetés állapotát, így nincs visszajelzés, hogy a két következtetési irány mikor találkozik.

2.5 Kapcsolat a produkciós rendszer kategóriák és a probléma osztályok között.

Minden megoldható problémához végtelen sok produkciós rendszert lehet konstruálni, amelyek valamilyen módon megoldást keresnek az adott problémára. Ezek közül lesz olyan, amelyik természetesebb, jobban illeszkedő leírást biztosít, mások pedig hatékonyabb megoldást biztosítanak. Minden olyan probléma, amit meg lehet oldani valamilyen produkciós rendszer absztrakcióval, megoldható kommutatív produkciós rendszerrel is, azonban praktikus szempontból ez a legszigorúbb leírás gyakran haszontalan.

Ezért bár elvi megfontolásokból nem igazán lehet a probléma osztályokat produkciós rendszer kategóriákhoz rendelni, praktikus oldalról tekintve mégis ésszerű ezt megtenni. A 2. ábra ezt a csoportosítást mutatja.

Implementációs szempontból a részlegesen kommutatív, monoton produkciós rendszerek fontosak, mert anélkül is megvalósíthatók, hogy beépítenénk visszalépési stratégiát (előző állapot visszaállítása).

Azonban a szisztematikus keresés támogatása érdekében ezen rendszereket is érdemes kibővíteni visszalépési képességgel, így a probléma állapotát leíró adatbázist nem feltétlenül kell visszaállítani. Ez gyakran jelentős hatékonyság növekedést

eredményez, különösképp mert számos adminisztratív terhet le lehet venni a rendszer válláról (hol milyen változtatások történtek az adatbázisban).

Ez a kategória elsősorban olyan problémák leírására alkalmas, ahol a dolgok nem változnak, hanem új dolgok keletkeznek.

Ezzel szemben a részben kommutatív nem monoton rendszerek jó reprezentációt biztosítanak olyan problémák leírására, ahol változások következnek be, de ezek a változások reverzibilisek, a korábbi állapotok visszaállíthatók és a műveletek sorrendje nem befolyásolja az eredményt. A részben kommutatív rendszerek általában az állapotok többszöri bejárásához vezetnek.

A nem monoton, részben kommutatív produkciós rendszerek elsősorban olyan problémák leírásában nyújthatnak segítséget, ahol irreverzibilis változások következnek be. Ekkor az operátorok alkalmazási sorrendje döntően befolyásolja a rendszer működését.

2.6 A szabályalapú rendszerek alkalmazásának előnyös tulajdonságai

- *Homogén tudásábrázolás.*

  A szabályalapú rendszerek mint tudásábrázolási és következtető rendszerek rugalmasak és nagy mozgásteret engednek, mivel homogén tudásábrázolást alkalmaznak. Ez a homogén tudásábrázolás, leírási mód egy rugalmas, de mégis kötött struktúrát követ, ami támogatja a szabályok (magának a tudásnak) megértését mások számára is.

- *Modularitás.*

  A szabályalapú rendszerek további szabályok menet közbeni hozzáadásával lehetővé teszik a rendszerben foglalt modellezési tudás inkrementális bővítését, illetve amennyiben a modellünk dinamikusan változik, ezek a változások akár szabályok törlésével is érvényre juttathatók. A szabályok hozzáadása, törlése vagy módosítása a többi szabálytól függetlenül végezhető. Ennek elsődleges oka, hogy a szabályok csak a munkamemórián keresztül kommunikálnak.

- *Természetes leírásmód.*

  A produkciós szabályok felépítése az emberi problémamegoldás sémáját tükrözi. Egy adott helyzetben a produkciós szabályok fejezik ki a legjobban, hogy "mit is kell tenni a következő lépésben" típusú utasításokat, illetve gyakran követik az emberi problémamegoldás során alkalmazott következtetési lépéseket, azokat "másolják le".

- *A belső ábrázolású tények jó értelmezhetősége.*

  A 3. ábra mutatja a valós problémában megjelenő tények és ezeknek a következtető rendszerekben történő belső ábrázolása közötti kapcsolatot. Az ábrán jól látható, hogy az eredeti cél, hogy a megoldandó probléma kiindulási tényei alapján egy logikailag teljes következtetéssel megkapjuk a probléma végső tényeit, vagyis a probléma megoldását (felső vízszintes szaggatott nyíl). A probléma megoldására azonban tipikusan valamilyen eszközt alkalmazunk, amely a tényeket egy belső ábrázolási formára konvertálja, illetve amikor a megoldás előáll a belső ábrázolási formában, azt az eszköz konvertálja a "probléma nyelvére". (Az ábra függőleges nyilai jelzik ezt a kétirányú konverziót.) Tehát a problémát egy adott leírási formalizmussal egy adott eszközzel oldjuk meg

(folytonos vízszintes nyíl). Ennek a megoldásnak a közbülső lépései gyakran nehezen vagy egyáltalán nem követhetők.

A szabályalapú rendszerekben a belső ábrázolási forma a felhasználó számára jól érthető, mert a munkamemóriában pontosan azok a primitívek, fogalmak jelennek meg, amelyeket a probléma modellezése során a szakértő a szabályokban alkalmazott, ezért a probléma megoldás közbülső állapotaiban a munkamemória tartalma az eredeti probléma primitívjeire, fogalmaira kivetítve is jól érthető, értelmezhető.

2.7 A szabályalapú rendszerek hátrányos tulajdonságai

A szabályalapú rendszerek két fő hátrányos tulajdonsággal rendelkeznek:

- *Merevek.*

  Az egységesség bár egyik oldalról rugalmasságot is jelent, másrészt egy merev struktúrát vezet be, ami a problémamegoldás során megnehezíti a következtetés menetének nyomon követését. Az általános szabályalapú rendszer architektúrában nehézkesen lehet megvalósítani több hierarchikus szinten történő következtetést, ebből adódóan nehéz elválasztani a magas szintű absztrakciókat az alacsony szintű részletektől. Ezen probléma megoldásához segítségül lehet hívni a meta-szabályokat, de alkalmazásuk erre a célra egy kissé körülményes. Pszeudo-hierarchikus szabályalkalmazást lehet elérni a szabályokhoz rendelt precedencia mértékekkel.

- *Nem hatékonyak.*

  A produkciós szabályok végrehajtása nem hatékony. Minden akciónak végig kell menni az illesztés-cselekvés cikluson a munkamemóriában, ami miatt nehéz hatékonyan végrehajtani előre meghatározott, adott helyzettől függő cselekvéssorozatokat, vagy a következtetést absztraktabb szintre emelni és nagyobb lépésekben haladni. Mint azt a későbbiekben majd látni fogjuk, a hatékonysági problémát a RETE hálók alkalmazásával nagy mértékben csökkenteni lehet, azonban az sem oldja meg az absztrahálás problémáját. Metaszabályokkal kicsit körülményesen, de részben kezelni lehet a problémát.

2.8 A produkciós szabályok hatékony alkalmazási területei

Barr és Feigenbaum (1981) szerint a produkciós szabályokat az alábbi területeken lehet a leghatékonyabban alkalmazni:

- *Szétszórt tudásterületek.*

  Ezek a területek sok tényt tartalmaznak, mint pl. orvosi alkalmazások.

- *Olyan területek, ahol a tudásábrázolás és a kontroll, vezérlés szétválasztható.*

  Ezek azok a területek, ahol jól elkülönül maga a tudás és annak a felhasználása, mint pl. biológiai osztályozási taxonómiák.

- *Független cselekvéseket tartalmazó területek.*

  Ezek olyan területek, amelyeken a végbemenő folyamatokat független cselekvéshalmazokkal lehet reprezentálni, mint például orvosi betegfelügyelő rendszerek.

A szabályalapú rendszerek lehetőséget nyújtanak nem tervezett, de hasznos egymásra hatások figyelembe vételére. Egy tudáselemet a rendszer bármikor, amikor az alkalmazható, alkalmazhat, nem csak akkor, amikor azt a rendszer programozója előre jósolta, így esetleg olyan finom kapcsolatok is napvilágot látnak, amelyek megléte még magában a modellalkotóban sem kristályosodott ki.

2.9 Sebességi problémák kezelése

Szabályalapú rendszerek futtatási statisztikái kimutatták, hogy a szabályalapú rendszerek működési ciklusában a mintaillesztési fázis domináns. Egyes felmérések azt mutatják, hogy a futási idő akár 90%-át is kiteheti a mintaillesztési fázis. Ez a felismerés felhívta a figyelmet a hatékony illesztés megvalósításának fontosságára. A problémát több megközelítésben is lehet enyhíteni, az alábbiakban ezek közül a legfontosabbakat tárgyaljuk.

2.10 Indexelés

2.10.1 Tábla alapú indexelés

A tudásbázist (munkamemóriát hash táblaként is meg lehet szervezni, ami a visszakeresést jelentős mértékben felgyorsítja. A hash tulajdonsága, hogy nagyjából állandó idő alatt tud visszakeresést végezni.

2.10.2 Fa alapú indexelés

Az alkalmazható szabályok kiválasztását egyszerű kereséssel is meg lehet oldani, amely során a teljes szabálybázist végigkeressük, és illesztjük a szabályokat. Ezzel az egyszerű megoldással két alapvető probléma van:

- Bonyolult feladatok megoldásához sok szabályt kell alkalmazni. A megoldás minden lépése során az összest megvizsgálni szinte reménytelen és nem hatékony.
- Nem mindig azonnal eldönthető, hogy egy szabály feltételrészei teljesülnek-e egy adott állapotban.

2.11 A RETE ALGORITMUS

Bár az egyesítést alkalmazni lehet az előfeltételek és az állapotokat leíró elemek keresztszorzatán, de sokkal hatékonyabb lehet egy *sok-sok* illesztési algoritmus, amelyben számos szabályt illesztünk egyszerre számos munkamemória tartalommal, egymással párhuzamosan.

Manapság az előrefelé következtető rendszerekben az egyik legelterjedtebben használt, a sok-sok illesztést hatékonyan megvalósító algoritmus a RETE algoritmus, a szabályalapú keretrendszerek többsége annak valamilyen változatát alkalmazza.

Az algoritmus a hatékonyságát elsősorban az alábbi három forrásból meríti:

- *Az adatok időleges természete.*

  A szabályok általában nem változtatják meg lényegesen az állapotleírásokat. Általában csak egy-két új elemet illesztenek be a munkamemóriába vagy törölnek onnan. Ennek hatására a munkamemória tartalma (vagyis az állapotleírás) nem változik meg lényegesen. A RETE felépít és karbantart egy állapot hálót, és a munkamemória változásai alapján határozza meg, hogy mely új szabályok váltak

alkalmazhatóvá (illetve mely szabályok váltak a változás hatására alkalmazhatatlanná).

- *A szabályokban fellelhető strukturális hasonlóság.*

  Eltérő szabályoknak számos közös előfeltételük lehet. Ezeket együtt illesztve sok időt lehet megtakarítani az illesztés során. A RETE háló olyan formában tárolja, hogy azok a memória struktúrákon is osztottan használják, így a több szabály feltétel részében is megjelenő feltételeket ciklusonként (legfeljebb) egyszer illesztődnek.

- *A változó kötések konzisztenciájának állandósága.*

  A RETE háló letárolja a változó kötéseket, így nem kell azokat minden illesztéskor újraszámolni, az hatékonyan hozzáférhető., illetve új változó értékkötéseket hatékonyan tud ötvözni a már meglévő kötésekkel.

2.12 Szakértői rendszerek implementálása

Kezdetben minden szakértői rendszert az alapoktól kezdve terveztek és implementáltak. Miután számos rendszert kidolgoztak, nyilvánvalóvá vált, hogy számos közös tulajdonságuk van, és ezek nagy része probléma terület független. Mivel a rendszerek nagy része valamilyen deklaratív leírást (tipikusan szabályokat) tartalmazott, amiket egy szabályértelmező használt, lehetővé vált elválasztani az interpretert magától a probléma területet leíró tudáselemektől, így olyan rendszer jött létre, ami újabb szakértői rendszerek magját képezhette. Az így kialakult rendszereket nevezik szakértői keretrendszereknek (vagy Expert System Shell-eknek).

2.13 A szabálybázis kidolgozása, hangolása

Ezekben a keretrendszerekben a tudásmérnök feladata, hogy a probléma terület modelljét szabályok formájában megfogalmazza és bevigye a keretrendszerbe. Az elkészült szabálybázisnak megfelelően a keretrendszer működését be kell hangolni, illetve, amennyiben szükséges, megfelelő meta-szabályokat kell megfogalmazni.

Ma számos public domain, illetve kereskedelmi célú szakértői keretrendszer áll rendelkezésre, amik magukban foglalják a legfontosabb keretrendszer funkciókat, azonban lényeges eltérés mutatkozik a felkínált szolgáltatásokban.

Ezek nagy része tipikusan támogatja a szabályokat, kereteket, igazságkarbantartó rendszereket és számos más következtetési mechanizmust.