Mészáros Tamás okl. villamosmérnök, doktorandusz,
Vadász Balázs okl. villamosmérnök, tanársegéd
Budapesti Mûszaki Egyetem, Mûszer- és Méréstechnika Tanszék

A RETE MATCH ALGORITMUS BÕVÍTÉSE: AZ ELÕRE ÉS HÁTRAFELE LÁNCOLÓ KÖVETKEZTETÉS EGYIDEJÛ TÁMOGATÁSA

Bevezetés

A Rete illesztõ algoritmus széles körben az egyik legelfogadottabb megoldás a nagyszámú mintát és nagyszámú adatot feldolgozó mintaillesztési feladat elvégzésére [Forgy79]. Az algoritmust elsõsorban produkciós rendszerek elõre következtetés alatti mintaillesztõ algoritmusában felmerülõ mintaillesztési feladat megoldására alkalmazzák. Legfontosabb elõnyeinek egyike, hogy az illesztés állapotinformácóit átmenti az elõre következtetés egyes ciklusai között. Az egyes ciklusokban csak a munkamemória változásaival végzi el az illesztést, valamint kiaknázza a szabályok feltételrészeinek strukturális hasonló-ságában rejlõ lehetõségeket.
Az algoritmusnak azonban vannak hátrányai. Nem teszi lehetõvé a szabálybázis futásidejû módosítását, nincs benne jól elkülönített, koncentrált munkamemória, valamint nem támogatja a szabályalapú rendszerek visszafelé következtetõ mûködését.

A cikk egy megoldást mutat be ezen utóbbi problémára. Leírja azokat a bõvítéseket, amelyek a Rete illesztõ algoritmust képessé teszik a visszafelé következtetés alatti illesztési feladatok elvégzésére is. A cikkben bemutatjuk a visszefelé következtetõ algoritmus fõbb lépéseit, valamint részletesen megadjuk a Rete algoritmus felhasználását az egyes lépésekben, továbbá ismertetjük az algoritmus adatszerkezeteinek és algoritmusainak szükséges bõvítését.

A cikkben leírt megfontolások alapján elkészült egy teljes következtetõ gép, amely egyszerre támogatja mind az elõre- mind a hátrafele következtetést.

A visszafelé következtetés áttekintése

A produkciós rendszerek visszafelé következtetõ, másnéven a hipotézisvizsgáló mûködése a szabálybázis és a munkamemória adott tartalma alapján egy állítást próbál igazolni. Általában ez az állítás egy megkívánt, de hiányzó munkamemória elem. A szabályalapú rendszer állapotát a munkamemória tartalmával megadva a visszafelé következtetés célját megfogalmazhatjuk úgy, mint annak az állapotnak az elérését, amelyben a munkamemória tartalmazza a megkívánt elemet.

A visszafelé következtetés az elõre következtetéshez hasonlóan ciklikus mûködésû. Az elsõ ciklus kezdõpontja a megkívánt munkamemória elem. Minden egyes ciklus bemeneteként megkapja a megkívánt munkamemória elemek halmazát, majd helyettesíti e halmazt egy (lehetõleg kisebb méretû) másikkal, amely a következõ ciklus számára szolgál bemenetként. A cél egy üres halmaz elérése, ekkor a hipotézisek igazolódtak. Más szavakkal: a produkciós rendszer megkívánt és jelenlegi állapota közötti út elõállt.

A visszafelé következtetõ ciklus lépései a következõek. Elsõként a bemenõ halmaz elemeinek meglétét ellenõrizzük a munkamemóriában. Azokat az elemek, amelyek léteznek, kivesszük a hipotézis halmazból. A második lépés kiválasztja azokat a szabályokat, amelyek a hipotézis halmaz legalább egy elemét képesek elõállítani. Ezen kiválasztott szabályok némelyike a konfliktus halmazban lehet, a harmadik lépésben ezt ellenõrizzük. A konfliktus halmazban található szabályok alkalmazásával a hipotézis halmaz egyes elemei elõállíthatóak, ezeket az elemeket tehát kivesszük a halmazból. A negyedik lépés akkor következik, ha a hipotézis halmaz még mindig tartalmaz elemet. Ez a lépés elõállítja azokat a munkamemória elemeket, amelyeket a munkamemóriába helyezve a második lépésben kiválasztott szabályok bekerülnek a konfliktus halmazba. Ebben az utolsó lépésben elõállított munkamemória elemek alkotják az új hipotézis halmazt, amelyet a soron következõ ciklus fog feldolgozni.

Az egymás utáni ciklusok abban az esetben állnak le, ha a hipotézis halmaz kiürül, vagy egyik ciklusról a másikra nem változik.

A Rete illesztõ algoritmus bõvítése: a visszafelé következtetés támogatása

A Rete algoritmus egy hatékony mintaillesztõ az elõre következtetés alatti mintaillesztési feladat elvégzésére, de nem támogatja a szabályok kétirányú felhasználását, amely a szabályalapú rendszerek eredeti ötlete volt.

Célunk egy olyan Rete-alapú következtetõ gép kidolgozása volt, amely képes a Rete-háló használatára mind elõre, mind visszafelé következtetés alatt. Ez implicit módon azt is jelenti, hogy mindkét esetben ugyanazt a munkamemóriát használja.

A következõkben a fentebb leírt lépéseket ismertetjük Rete illesztõ alkalmazása esetén. Ezek röviden a következõek:

- ellenõrzés a jelenlegi munkamemóriában,
- azon szabályok kiválasztása, amelyek a megkívánt elemeket létrehozhatják,
- a kiválasztott szabályok keresése a konfliktus halmazban,
- azoknak a munkamemória elemeknek a keresése, amelyek lehetõvé teszik a kiválasztott szabályok legalább egyikének az alkalmazását, íly módon elõállíthatóvá válik egy hipotézis.

Keresés a munkamemóriában

A Rete illesztõ algoritmus egyik legmeghatározóbb tulajdonsága a koncentrált munkamemória hiánya. A munkamemória a Rete hálóban van "szétkenve". Az algoritmus lényeges tulajdonsága, hogy azok a munkamemória elemek, amelyekkel egyetlen szabály feltételrésze sem foglalkozik, sehol sem tárolódnak. A hálóban a munkamemória elemeket a kétbemenetû csomópontok tárolják a [Forgy79] irodalomban leírtaknak megfelelõen..

Az olyan elemek, amelyekkel egy részbõl álló feltétellel rendelkezõ szabályok foglalkoznak csak, nem tárolódnak a hálóban, mivel végighaladva a gráfban nem érintenek kétbemenetû csomópontot. Az ilyen elemek azonban megtalálhatóak a konfliktus halmazban. Annak érdekében, hogy a konfliktus halmaz-beli keresést elkerüljük, kiterjesztettük a lezáró csomópontokat is egy memóriával, amelyben minden, a csomóponthoz beérkezõ munkamemória elem tárolódik. Íly módon a hálózaton áthaladó minden munkamemória elem megtalálható lesz legalább egy csomópont memóriájában.

A megkivánt elem keresése a normális illesztési mûködéshez igen hasonlóan zajlik le. A keresendõ elemet elküldjük a bemeneti csomópontnak, amely elõállít egy tokent az elemmel. Az egyes csomóponti programok, amelyek megvizsgálják a csomóponthoz beérkezõ elemeket a következõk szerint módosulnak.
Az egybemenetû csomópontok programja nem módosul.
A kétbemenetû csomópontok a token beérkezési oldalán ellenõrzik a memóriájukban a tokent. Ha megtalálták, akkor a keresés véget ért. Amennyiben nem, úgy a keresés folytatódik, de a kétbemenetû csomópont már nem küldi tovább a tokent az õt követõ csomópontoknak, hiszen azok memóriáiban már nem szerepelhet a keresett token.
A lezáró csomópontok a kétbemenetûekhez hasonló módon mûködnek, ellenben csak egy memóriával rendelkeznek. Ezek a csomópontok csak abban az esetben jutnak szerephez, ha a gráfban köztük és a bemeneti csomópont között nincs kétbemenetû csomópont.
A keresés vagy a keresett elem megtalálásával, vagy a gráf teljes bejárásával ér véget.

Szabálykeresés

E lépés célja azon szabályok megkeresése, amelyek képesek elõállítani a megkívánt munkamemória elemet. Mindezt a szabályok jobb oldalainak vizsgálatával érhetjük el. A lépés eredménye azon szabályok listája, amelyek következmény részükben tartalmaznak olyan munkamemória elem elõállító utasítást (MAKE), amely éppen a megkívánt elemet hozza létre.
A MAKE utasítás meghatározott típusú és néhány adott tulajdonsággal rendelkezõ munkamemória elemet állít elõ. A feladat ezeknek a mintáknak és a megkívánt munkamemória elemnek az összevetése. Erre a mintaillesztési feladatra a Rete algoritmust alkalmazzuk. A MAKE utasításokban szereplõ mintákból építünk egy ún MAKE-Rete hálót, amely elvégzi ezt a keresési feladatot. A MAKE-Rete háló nem tartalmaz kétbemenetû csomópontokat, mivel a MAKE utasítások mindig egy munkamemória elemet állítanak elõ. A háló lezáró csomópontjai választják ki a szabálybázisból azokat a szabályokat, amelyek a bemenõ csomópontra érkezõ munkamemória elemet képesek elõállítani.

A konfliktus halmaz ellenõrzése

E lépés célja annak ellenõrzése, hogy az elõzõ lépésben kiválasztott szabályok valamelyike megtalálható-e a konfliktus halmazban.

A konfliktus halmaz szabálypéldányokat tartalmaz, melyek általában több megkötést tartalmaznak a szabályoknál. Egy szabály jobb oldalán lehetnek változók, melyek a szabály bal oldaláról nyerik el kötésüket a szabálypéldány létrehozása során (azaz a szabály két oldalán ugyanazzal a névvel szerepelnek). Ezek a változók információt visznek át a szabály feltételrészéből a következmény részébe. Ennek megfelelõen, ha a konfliktus halmazban találunk egy megfelelõ szabálypéldányt, ellenõriznünk kell ezeket a változókötéseket is. Amennyiben azok nem felelnek meg az elõírt munkamemória elem tulajdonságainak, akkor a kiválasztott szabálypéldány nem képes a megkívánt munkamemória elem elõállítására.

Mivel a Rete illesztõ algoritmus nem foglalkozik a konfliktus halmaz kialakításával, e keresési mechanizmus leírása kívül esik e cikk célkitûzésein.

A hiányzó munkamemória elemek keresése

Ez a lépés akkor következik, ha az elõzõk nem érték el céljukat, azaz nem sikerült a bizonyitandó hipotézist megtalálni a munkamemóriában, illetve a konfliktus halmazban nincs olyan szabály, amely az elemet elõállíthatná. Ebben az esetben azokat az elemeket kell megkeresnünk, amelyek még hiányoznak a munkamemóriából ahhoz, hogy egy, a megkívánt elemet elõállítani képes szabálypéldány bekerüljön a konfliktus halmazba. Erre a célra az elõrekövetkeztetés alatti Rete hálót alkalmazzuk módosított csomóponti programokkal, mivel mind a szabályok feltételei, mind a már meglévõ munkamemória elemek ebben találhatóak.

A feladat tehát hiányzó munkamemória elemek elõállítása és azok tulajdonságainak megfelelõ beállítása. A típust és a tulajdonságokat a teszt csomópontokban elvégzett vizsgálatok határozzák meg. A gráfbejárás kezdõpontja a hálóban nem a normál mûködés során megszokott bemenõ csomópont, hanem a MAKE-Rete háló által kiválasztott lezáró csomópontok. A hálót e lezáró csomópontoktól a bemenõ csomópont felé haladva járjuk be. A lezáró csomópontok elõállítanak egy (egyelõre) nem definiált tulajdonságokkal és típussal rendelkezõ munkamemória elemet (illetve elemeket) tartalmazó tokent, amelyben kitöltik a feltétel és következmény rész közötti változókötésbõl adódó tulajdonságértékeket. Ez az ún. biankó token.

Minden csomópont az elõre következtetés alatti kimenete felõl kapja meg ezeket az biankó tokeneket, majd a bemenetén át továbbítja az õt megelõzõ csomópontoknak. Amikor egy csomópont kap egy biankó tokent, akkor az általa elõrekövetkeztetés alatt végrehajott teszt alapján kitölti a még nem definiált tulajdonságok egyikét oly módon, hogy ha a token az elõre következtetés alatt áthaladna a csomóponton, akkor kielégítené annak tesztjét. A típustesztelõ csomópontok a biankó tokenben található munkamemória elem típusának értékét állítják be. A kétbemenetû csomópontok mûködése kicsit összetettebb. Ezek olyan biankó tokent kapnak, amely két (vagy több) munkamemória elemet tartalmaz (mivel elõre következtetés alatt ezek a csomópontok tokeneket fûznek egybe). Elsõ lépésként a tokent két megfelelõ méretû tokenre bontja szét a kétbemenetû csomópont visszafelé következtetés alatt futó csomóponti programja. A két résztoken struktúrája megfelel az elõre következtetés alatt a csomóponthoz beérkezõ két token szerkezetének. Ezután mindkét oldal csomóponti memóriájában megkeresi a megfelelõ oldali tokent. Ha egyik oldalon sem talált tokent, akkor a két tokent változtatás nélkül elõreküldi a megfelelõ oldalakon. Amennyiben az egyik oldali memóriájában talált egyezõ tokent, úgy csak a másik oldalon küldi elõre a tokent. Ennek az elõreküldött tokennek az elõreküldés elõtt néhány tulajdonságát kitölti a csomópont által elõrekövetkeztetés alatt vizsgált, a feltételrészek közötti változókötéseknek megfelelõen. Amennyiben egy oldalon több tokent is talál, úgy minden egyes megtalált token alapján elvégzi ezt a tulajdonságkitöltést, majd elõreküld egy-egy tokent a másik oldalon. Ha mindkét oldalon talált tokent, akkor ezen eljárást mindkét oldalra elvégzi. Egy token beérkezése egy kétbemenetû csomópont kimenetén át tehát több token továbbküldését is eredményezheti a csomópont két bemenetén keresztül.

Konklúzió

A cikkben ismertetett munka megmutatta, hogy a Rete algoritmus kiterjeszthetõ volt a visszafelé következtetés támogatására. Mivel az algoritmus alkalmazása esetén a munkamemória a Rete hálóban elosztva tárolódik, ezért egy szükségszerû kényszer az, hogy visszafelé következtetés alatt is ugyanezt a hálót alkalmazzuk. A cikk leírt egy olyan módszert, amely megfelel ennek a követelménynek. A leírt megoldás ugyanazt a Rete gráfot alkalmazza kétirányú módon.

Ez a kibõvített Rete illesztõ algoritmus egy teljes következtetõ gép részeként került megvalósításra. Ez a megvalósítás rugalmas eszközöket nyújt más problémamegoldó technikák számára, a teljes rendszer az objektum-orientált filozófiát követve épült fel oly módon, hogy a munkamemória elemekre nem tesz megkötést. Hatékonysági szempontokat figyelembe véve a rendszer C++ nyelven készült el [Mészáros93].

Hivatkozások

Forgy79 Forgy, C.L.: On the Efficient Implementation of
Production Systems, doktori értekezés, Carnegie-Mellon University, 1979.

Forgy82 Forgy, C.L.: Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, Artificial Intelligence 19, pp. 17-32, 1982.

Mészáros93 Mészáros, T.: Rete Match alapú interpreter objetum-orientált környezetben, diplomaterv, Budapesti Mûszaki Egyetem, Mûszer- és Méréstechnika Tanszék, 1993.

Perlin90 Perlin, M.: Topologically Traversing the Rete Network,
Applied Artificial Intelligence 4, pp.155-177, 1990.