Magyar nyelvű szótárak tömör reprezentációja nemdeterminisztikus automatákkal

Kertész-Farkas Attila[1], Fülöp Zoltán[2], Kocsor András[3]

[1,3] Szegedi Tudományegyetem, MTA-SZTE Mesterséges Intelligencia Kutatócsoport, Aradi vértanúk tere 1, 6720 Szeged *
[2]Szegedi Tudományegyetem, Számítástudomány Alapjai Tanszék, Árpád tér 2, 6720 Szeged
[1]kfa@rgai.inf.u-szeged.hu
{[2]fulop, [3]kocsor}@inf.u-szeged.hu

Kulcsszavak: automata, minimális automata, automata-tömörítés

1. Bevezetés

Azokban a számítógépes alkalmazásokban, amelyek véges nyelvekkel, nyelvi korpuszokkal dolgoznak, – mint például a beszédfelismerésben, beszédszintézisben alkalmazott programok, – a nyelvek reprezentálására hatékonyságuk és egyszerű szerkezetük miatt automatákat érdemes alkalmazni. Egy nyelv felismerésére számos egymással ekvivalens automata megkonstruálható, melyek közül a lehető legkisebb méretűt, a legtömörebbet érdemes használni.

Ebben a cikkben véges nyelveket felismerő automatákat tömörítő heurisztikus algoritmusokkal foglalkozunk. Automata tömörítő algoritmuson olyan algoritmust értünk, amely egy (általában nemdeterminisztikus) $\mathcal{A}$ automatából kiindulva megkonstruál egy $\mathcal{A}$-val ekvivalens nem feltétlenül determinisztikus, de kisebb méretű automatát.

A minimális determinisztikus automata (MDFA) megkonstruálásával sokan foglalkoztak, lásd a [W94] összefoglaló munkát, viszont nemdeterminisztikus automaták (NFA) tömörítésére eddig kevesebb figyelem esett. Közismert, hogy megadható olyan k állapotú NFA, mellyel ekvivalens MDFA-nak 2^k állapota van [AJV99]. Ugyanakkor, a minimális állapotszámú NFA kiszámolása NP-nehéz probléma [JR93], ezért heurisztikus algoritmusok kidolgozására van szükség. [AJV99]-ben egy olyan heurisztikus tömörítő algoritmust dolgoztak ki, amely egyenlő hosszúságú szavakat tároló automatán működik, és az automata gráfján definiált biklikk lefedő rendszerek alapján végzi el a megfelelő tömörítést. Ezt a módszert általánosították tetszőleges nyelvet felismerő automatákra [CC03]-ban. Ez utóbbi heurisztikus algoritmus azonban nagy időigénye miatt a gyakorlatban nem igazán alkalmazható.

A DFA-t minimalizáló algoritmusok nemcsak az állapotszám, hanem az átmenetek száma szerint is az MDFA-t konstruálják meg. A NFA-k esetében azonban

* Ez a cikk az Oktatási Minisztérium 2001/055 számú IKTA pályázata támogatásával készült.

más a helyzet. Ha csak állapotszám szerint tömörítünk, mint például az [AJV99]-ben szereplő algoritmus is, akkor az átmenetek száma akár meg is sokszorozódhat és ezáltal az automata mérete növekszik. Ezt a tényt az eddigi tömörítő algoritmusok nem vették figyelembe, így például az [AJV99]-ben megadott algoritmus a kiindulási automatához képest több mint kétszer akkora helyen tárolható automatát is adott eredményül.

A cikkben, továbbfejlesztjük az [AJV99]-ben megadott tömörítési eljárást és megadunk egy olyan gyors heurisztikus algoritmust, amely nemcsak az állapotok száma szerint tömörít eredményesen, hanem korlátozza az átmenetek számának növekedését is. Így az eljárás minden esetben garantáltan kisebb méretű automatát eredményez.

A cikk felépítése a következő. A második fejezetben definiáljuk a szükséges fogalmakat, majd a harmadik fejezetben ismertetjük az említett automata tömörítő algoritmust. A negyedik fejezetben bemutatjuk az algoritmus futási eredményeit magyar nyelvű korpuszokon, végül az ötödik fejezetben összegezzük a cikk eredményeit.

2. Definíciók

Egy H halmaz számosságát $|H|$-val jelöljük. Egy Σ ábécé feletti szavak halmazát Σ^*-gal jelöljük, Σ^* tetszőleges L részhalmazát pedig Σ *feletti nyelvnek*, vagy röviden csak *nyelvnek* nevezzük. *Nemdeterminisuztikus automatának* (NDFA-nak) nevezünk egy $\mathcal{A} = (Q, \Sigma, \delta, I, F)$ rendszert, ahol Q az állapotok véges, nemüres halmaza, Σ az input ábécé, $I \subseteq Q$ a kezdő-, $F \subseteq Q$ a végállapotok nemüres halmaza és $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ az átmenetfüggvény. Ha I egy elemű és minden $a \in \Sigma$-ra és $q \in Q$-ra $\delta(q, a)$ legfeljebb egy elemű, akkor $\mathcal{A}$ *determinisztikus* (röviden DFA). Tetszőleges $q \in Q$-ra és $E \subseteq Q$-ra legyen $\gamma^+(q) = \{(a, q') \in \Sigma \times Q \mid q' \in \delta(q, a)\}$, $\gamma^+(E) = \bigcup_{q \in E} \gamma^+(q)$ és $\gamma^-(q) = \{(q', a) \in Q \times \Sigma \mid q \in \delta(q', a)\}$, továbbá $q^+ = \bigcup_{a \in \Sigma} \delta(q, a)$ és $q^- = \{q' \in Q \mid (\exists a \in \Sigma)\, q \in \delta(q', a)\}$.

A δ függvényt kiterjesztjük $\mathcal{P}(Q) \times \Sigma^* \to \mathcal{P}(Q)$ típusú leképezéssé úgy, hogy minden $w \in \Sigma^*$-ra és $a \in \Sigma$-ra $\delta(q, wa) = \delta(\delta(q, w), a)$, majd $\delta(E, w) = \bigcup_{q \in E} \delta(q, w)$. Az $\mathcal{A}$ automata által felismert nyelven az $L(\mathcal{A}) = \{w \in \Sigma^* \mid \delta(I, w) \cap F \neq \emptyset\}$ nyelvet értjük.

Azt mondjuk, hogy $\mathcal{A}$ *egyértelmű* (röviden UFA), ha minden $w \in L(\mathcal{A})$ szóra $\mathcal{A}$ gráfjában pontosan egy út vezet valamely I-beli kezdőállapotból valamely F-beli végállapotba. Minden DFA egyben UFA is. Az $\mathcal{A}$ automata transzponáltján az $\overleftarrow{\mathcal{A}} = (Q, \Sigma, \delta', F, I)$ automatát értjük, ahol $\delta'(q, a) = \{p \mid q \in \delta(p, a)\}$. A cikkben csak olyan automatákkal foglalkozunk, amelyeknek a gráfja nem tartalmaz kört, tehát amelyek véges nyelveket tárolnak, ismernek fel. Feltesszük továbbá, hogy egy automata minden állapota elérhető valamely kezdőállapotból és minden állapotból eljuthatunk valamely végállapotba. Szükség esetén további részletek [AJV99]-ben találhatók.

3. Automata tömörítés

Ebben a fejezetben az $\mathcal{A}$ automatán az $\mathcal{A} = (Q, \Sigma, \delta, I, F)$ automatát értjük.

Definíció 1. Legyen $q \in Q$ és $S \subseteq Q$. A (q, S) párt *$\mathcal{A}$-beli egyesítéses tömörítésnek* nevezzük, ha $q \notin S$, $\gamma^+(q) = \gamma^+(S)$, a $\{q\} \cup S$ halmaz mindegyik eleme végállapot vagy egyike sem az, és teljesül, hogy ha $q \in I$ akkor $S \subseteq I$.

A (q, S) egyesítéses tömörítés $\mathcal{A}$-ra való alkalmazása azt jelenti, hogy a q-ba érkező átmeneteket átirányítjuk minden $p \in S$ állapotba, majd q-t és a belőle induló átmeneteket töröljük. Minden egyesítéses tömörítés elvégzése után az automatában az állapotok száma eggyel csökken.

A (q, S) egyesítéses tömörítés *diszjunkt*, ha minden különböző $s, s' \in S$-re $\gamma^+(s) \cap \gamma^+(s') = \emptyset$ és $\{q\} \cup S$ egyik eleme sem végállapot. Ha $\mathcal{A}$ UFA és egy (q, S) diszjunkt egyesítéses tömörítést alkalmazunk rá, akkor a kapott automata ugyancsak UFA lesz. Természetesen az $\mathcal{A}$-beli diszjunkt egyesítéses tömörítések száma általában kevesebb mint az egyesítéses tömörítések száma.

Ha a q-ba érkező átmenetek száma olyan nagy, hogy a tömörítés elvégzése után több élt kapunk, mint amennyit sikerül megspórolni, akkor a tömörítés növeli az automata méretét. Ennek kezelésére vezetjük be a következő fogalmat, mely cikkünk egyik legfontosabb eleme. A (q, S) egyesítéses tömörítés *valódi*, ha teljesül, hogy

$$|q^-| * (|S| - 1) < |q^+| + 1$$

Látható, hogy az egyenlőtlenség bal oldalán a tömörítés után kapott új átmenetek száma áll, míg a jobb oldal mutatja azt, hogy a tömörítéssel mennyi átmenettel és állapottal lesz kisebb az automata mérete. Tehát egy egyesítéses tömörítés akkor tömörít valóban, ha a fenti egyenlőtlenség teljesül.

Egyesítéses tömörítések egy $T = \{(q_1, S_1), (q_2, S_2), \ldots, (q_l, S_l)\}$ halmazát *megengedettnek* hívjuk, ha minden $1 \leq i \leq l$-re teljesül $q_i \notin \bigcup_{i=1}^{l} S_i$.

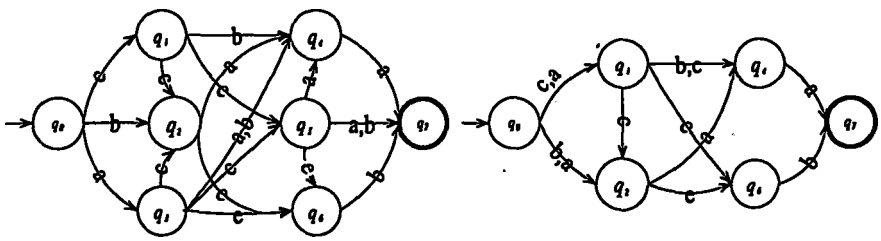
Definíció 2. Az $\mathcal{A}$ automatából egyesítéses tömörítések egy megengedett $T = \{(q_1, S_1), (q_2, S_2), \ldots, (q_l, S_l)\}$ halmazával kapott automatán azt az $\mathcal{A}' = (Q', \Sigma, \delta', I', F')$ automatát értjük, melyre

- $Q' = Q \setminus \{q_1, q_2, \ldots, q_l\}$, $I' = I \cap Q'$, $F' = F \cap Q'$,
- $\forall\, q \in Q'$ és $a \in \Sigma$ esetén $\delta'(q, a) = (\delta(q, a) \setminus \{q_{i_1}, \ldots, q_{i_k}\}) \cup S_{i_1}, \cup \cdots \cup S_{i_k}$, ahol $\{i_1, \ldots, i_k\} = \{1 \leq i \leq l \mid (q, a) \in \gamma^-(q_i)\}$.

A fenti definícióban a T egyesítéses tömörítésre vonatkozó megengedett feltétel szükséges ahhoz, hogy $\mathcal{A}'$-t egyértelműen definiálni tudjuk. A továbbiakban egyesítéses tömörítések halmazán mindig megengedett halmazt értünk, ezért a megengedett jelzőt el is hagyjuk.

Az 1. ábrán egy példa látható egyesítéses tömörítések egy halmazának alkalmazására, ahol a jobb oldali automatát a bal oldali automatából a $T = \{(q_3, \{q_1, q_2\}), (q_5, \{q_2, q_4, q_6\})\}$ halmazzal kaptuk, és így 66%-os tömörítést sikerült elérni.

Egy $\mathcal{A}$ NFA és a belőle tömörítéssel kapott $\mathcal{A}'$ NFA között a következő kapcsolat állapítható meg.



1. ábra. Példa egyesítéses tömörítések alkalmazására

Tétel 1. *Ha $\mathcal{A}'$ automatát az $\mathcal{A}$ automatából egyesítéses tömörítések egy T halmazával kapjuk, akkor $L(\mathcal{A}) = L(\mathcal{A}')$. Továbbá, ha $\mathcal{A}$ UFA és T elemei diszjunktak, akkor $\mathcal{A}'$ is UFA.*

Bizonyítás. A bizonyítás az [AJV99]-ben található hasonló állítás (Propozíció 2) bizonyításának általánosításával végezhető el.

Látható, hogy ha T valódi egyesítéses tömörítések halmaza, akkor a kapott $\mathcal{A}'$ automata állapot és átmenet száma kisebb, mint a kiindulási $\mathcal{A}$ automatáé, és így kisebb helyen tárolható.

Most megadjuk az általunk javasolt heurisztikus automata tömörítő algoritmust. Az algoritmus annyival több a [AJV99]-ben szereplőtől, hogy nemcsak azonos hosszúságú szavakat felismerő automatákat tud tömöríteni és, hogy csak valódi tömörítéseket enged meg.

3. Algoritmus. *ReductionAutomata*
input: *$\mathcal{A}$ automata*
output: *a kiindulási automatával ekvivalens, tömörített automata*

1 $n \leftarrow |Q|$;
2 *ismételjük meg kétszer:*
3 *keressük meg az összes valódi diszjunkt egyesítéses tömörítést, majd végezzük el a tömörítéseket a 2. definíció szerint;*
4 $\mathcal{A} \leftarrow \overleftarrow{\mathcal{A}}$;
5 **ha** $n \neq |Q|$ **akkor** *ugorjunk 1-re;*
6 **return** $\mathcal{A}$;

Ha az algoritmus 3. sorában nemcsak diszjunkt egyesítéses tömörítéseket keresünk, akkor az eredményül kapott automata még kisebb lesz, viszont az így módosított algoritmus nem őrzi meg az UFA tulajdonságot.

Az algoritmus úgy gyorsítható fel jelentősen, hogy egy (q, S) egyesítéses tömörítés meghatározása esetén a q állapothoz az S halmazba nem az egész automatában keresünk állapotokat, hanem csak a q állapotból elérhető állapotok őseit vizsgáljuk. Így egy, a gyakorlatban gyors algoritmust kapunk.

4. Futási eredmények

A teszteléshez néhány internetes portál szavaiból készítettünk adatokat. A tömörítő algoritmust egy Pentium III 700 MHz-es számítógepen futtatuk. A kiindulási automaták konstruálása úgy történt, hogy először standard algoritmussal egy fa gráfú DFA-t készítettünk, majd alkalmaztuk az eddig ismert leggyorsabb minimalizáló algoritmust [H71]. Ezután az így kapott MDFA-ra alkalmaztuk a cikkben javasolt automata tömörítő algoritmust. Négy automatát vizsgáltunk, a futási eredmények az 1. táblázatban láthatók.

szavak száma	kiindulási MDFA átmenetek és állapotok száma	futási idő (s)	tömörített automata átmenetek és állapotok száma	futási idő (s)	tömörítés aránya
8213	21432	2	20356	1	94.98%
112584	241463	1517	222961	226	92.34%
26852	63561	82	58618	15	92.22%
433367	794245	22947	738129	3517	92.93%

1. táblázat. A valódi diszjunkt egyesítéses tömörítések alkalmazásával kapott eredmények

A nyelvi korpuszok vizsgálatában gyakran előfordul, hogy további adatok – például súlyok vagy kimeneti szimbólumok – tárolására is szükség van. Ha tömören szeretnénk az automatát reprezentálni [K99] vagy ha az átmeneteket költségesen tudjuk tárolni, akkor érdemes korlátozni az átmenetek számának növekedését. A csak átmenetszám szerinti tömörítés esetén olyan esetek is adódhatnak, amikor egy új állapot akalmas hozzáadásával csökkenteni lehet az automata méretén. Ez újabb kutatási irányt vethet fel. Viszont, ha az automata állapotaihoz súlyokat rendelünk, vagy az átmeneteket alacsony költséggel tudjuk tárolni, akkor elvégezhetjük a nem valódi egyesítéses tömörítéseket is. Ha olyan rendszert implementálunk, amiben nem fontos, hogy az automata többször fogad el egy szót, vagyis nem UFA, akkor megengedhetjük a nem diszjunkt egyesítéses tömörítéseket is, és így tovább csökkenthetjük az állapotok számat. Ezért az algoritmust nemcsak valódi és diszjunkt egyesítéses tömörítések keresésére is lefuttattuk.

szavak száma	kiindulási MDFA állapotok száma	futási idő (s)	tömörített automata állapotok száma	futási idő (s)	tömörítés aránya
8213	7870	2	7235	1	91.93%
112584	81718	1517	65785	706	80.50%
26852	21995	82	18122	36	82.39%
433367	254140	22947	199045	8574	78.32%

2. táblázat. A nem csak valódi egyesítéses tömörítések alkalmazásával kapott eredmények

5. Konklúzió

A cikkben továbbfejlesztjük az [AJV99]-ben megadott tömörítési algoritmust és bemutattunk egy véges nyelveket felismerő, nemdeterminisztikus automatákra alkalmazható gyors heurisztikus tömörítő algoritmust. A tömörítés lényege, hogy az automatában (q, S) alakú egyesítéses tömörítéseket keresünk, ahol q egy állapot, S pedig állapotok egy halmaza, majd a q-ba érkező átmeneteket átirányítjuk az S-beli állapotokba és törljük q-t. Csak valódi egyesítéses tömörítéseket engedünk meg, ami biztosítja azt, hogy az automata mérete, amit az állapotok és az átmenetek együttes száma határoz meg, valóban kisebb lesz. Az algoritmust implementáltuk, majd alkalmaztuk négy automatára. Az algoritmus gyakorlati alkalmazhatóságát támasztja alá, hogy az elvégzett összehasonlító tesztek alapján a minimális determinisztikus automatánál 15-25%-kal kisebb (nemdeterminisztikus) automatát konstruál meg. Ezért érdemes lehet az algoritmust továbbfejleszteni, általánosítani tetszőleges reguláris nyelvet felismerő automtákra vagy megvizsgálni az átmenetek száma szerinti tömörítés lehetőségeit.

Hivatkozások

[AJV99] J. Amilhastre, P. Janssen and M. C. Vilarem, FA Minimization Heuristics for a Class of Finite Languages, *In: Proc. of WIA 99, Lecture Notes in Computer Science (Szerk. O. Boldt és H. Jürgensen), Vol. 2214, pp. 1-13, Springer-Verlag, 2001.*

[CC03] J.-M. Champarnaud, F. Coulon, NFA Reduction Algorithms by Means of Regular Inequalities, *In: Proc. of DLT 03, Lecture Notes in Computer Science (Szerk. Ésik Z. és Fülöp Z.), Vol. 2710, pp. 194-205, Springer-Verlag, 2003.*

[H71] Hocproft, J.E, An $n * log(n)$ algorithm for minimizing states in a finite automaton, *In: Theory of Machines and Comutations (Szerk. Y. Kohavi and A. Paz), Academic Press, New York, 1971, pages 189-196*

[JR93] Tao Jiang, B. Ravikumar, Minimal NFA problems are hard, *SIAM Journal of Computation, 22: 1117-1141, 1993.*

[K99] George Anton Kiraz, Compressed Storage of Sparse Finite-State Transducers, *In: Proc. of WIA 99, Lecture Notes in Computer Science (Szerk. O. Boldt és H. Jürgensen), Vol. 2214, pp. 109-122, Springer-Verlag, 2001.*

[W94] Bruce W. Watson, A taxonomy of finite automata minimization algorithms, *http://www.cs.up.ac.za/~watson/publications.html, 1994*