CONSTRUCTION OF THE GENERALIZED FINITE AUTOMATON

Martin Šoka

Bachelor Degree Programme (4), FIT BUT

E-mail: xsokam00@stud.fit.vutbr.cz

Supervised by: Jan Kaštil

E-mail: ikastil@fit.vutbr.cz

Abstract: This paper deals with construction of a generalized automaton and with its problems. A generalized automaton (GA) is a finite automaton where single transitions are defined on words rather than on single letters. GA consists of a lesser number of states and transitions, which eventually saves memory used for storing the automaton.

Keywords: generalized automaton, finite automaton, regular expressions, graph, subgraph, python

1. ÚVOD

Deterministické zovšeobecnené automaty (DZA) slúžia ako model reprezentujúci regulárne jazyky. Rozdiel medzi konečnými automatmi a zovšeobecnenými automatmi spočíva v tom, že zovšeobecnené automaty rozširujú konečné automaty o možnosť definovať jednotlivé prechody slovami a nie len znakmi. DZA môže byť získaný z obyčajného deterministického konečného automatu, skrátením dlhých ciest grafu na jedinú hranu. Algoritmus na získanie spomínaného automatu je objasnený v tomto príspevku, ktorý vychádza z článku [1]. Vstupom algoritmu je minimálny deterministický konečný automat, ktorý je vytvorený z jedného alebo z viacerých regulárnych výrazov. Prevod regulárnych výrazov na konečné automaty (a následnú determinizáciu a minimalizáciu) zabezpečuje knižnica NetBench, ktorá je vyvíjaná výskumnou skupinou akcelerovaných sieťových technológii (Accelereated Network Technologies @ FIT). Táto knižnica, rovnako ako aj algoritmus, je napísaná v jazyku Python (verzia 2.6).

Nasledujúca kapitola podrobnejšie popisuje algoritmus na vytvorenie deterministického zovšeobecneného automatu.

2. POPIS ALGORITMU

Ako bolo spomenuté v úvode, vstupom algoritmu je minimálny deterministický konečný automat. Z takého automatu vznikne deterministický zovšeobecnený automat zredukovaním prebytočných stavov. Odstránenie prebytočného stavu z automatu sa nazýva *S*-redukcia. S-redukciou zanikajú pôvodné hrany, ktoré obsahujú prebytočný stav a vznikajú nové hrany, kde z pôvodných prechodov definovaných znakmi vzniknú nové prechody definované slovami. Tieto slová vzniknú konkatenáciou znakov z pôvodných prechodov.

Hlavným problémom konštrukcie algoritmu je nájdenie množiny prebytočných stavov, ktoré majú byť zredukované. Pre nájdenie tejto množiny je nutné vyhľadať v grafe automatu maximálny acyklický podgraf zložený z množiny všetkých prebytočných stavov. Hľadanie maximálneho acyklického podgrafu v grafe je NP-úplný problém. Vďaka tomu nebol nikdy algoritmus pre vytvorenie zovšeobecneného automatu na hľadanie vzorov v dátach paketov použitý, z čoho vyplýva, že tento model pre reprezentáciu regulárnych jazykov nebol testovaný pre výrazy používané v moderných IDS (Intrusion detection system).

2.1. PREBYTOČNÉ STAVY

Prebytočný stav je každý stav v automate, ktorý nie je ani počiatočný ani koncový stav a nemá žiadne slučky, ktoré v ňom aj začínajú, aj končia. Množinu všetkých prebytočných stavov si označme $Superf(Q)$, kde Q je množina všetkých stavov automatu. Odstránením jedného prebytočného stavu z automatu môže nastať situácia, kedy sa veľkosť množiny $Superf(Q)$ zmenší o *1* až *N* stavov, kde $N = |Superf(Q)|$. Preto je nutné nájsť podmnožinu stavov množiny $Superf(Q)$, ktorá bude v automate indukovať acyklický podgraf. Ak sa odstráni z automatu prebytočný stav z takejto podmnožiny všetky ostatné stavy v podmnožine zostanú naďalej prebytočné. Ak je množina $Superf(Q)$ prázdna, automat neobsahuje žiadne prebytočné stavy a je *S*-neredukovateľný.

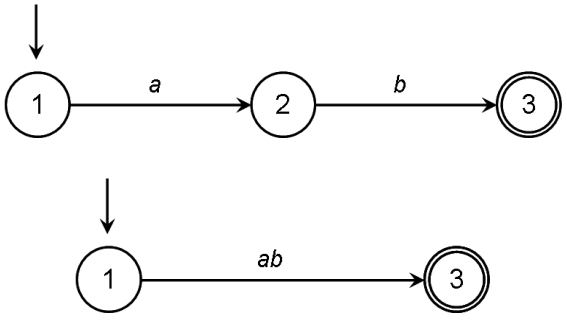
2.2. MAXIMÁLNY ACYKLICKÝ PODGRAF

Ak *S* je množina prebytočných stavov a indukuje v automate acyklický podgraf, potom je možné vytvoriť automat $S(A,S)$. Ak by množina *S* indukovala podgraf, v ktorom by bol jeden alebo viac cyklov, potom by mohlo nastať, že zredukovaním prebytočného stavu z množiny *S*, by ďalší prebytočný stav z tejto množiny prestal byť prebytočný. Podmienkou pre vytvorenie automatu $S(A,S)$, ktorý bude *S*-neredukovateľný je, že množina *S* musí indukovať maximálny acyklický podgraf. O množine *S* ďalej platí, že $S \subseteq Superf(Q)$.

Pre problém hľadania maximálneho acyklického podgrafu sme vytvorili heuristiku, podľa ktorej je implementovaná rekurzívna metóda na prechádzanie grafom automatu. Grafom sa prechádza po jednotlivých hranách, pričom sa vždy pred pridaním nového stavu do množiny *S* kontroluje acyklickosť podgrafu, ktorý indukuje množina *S*. Rekurzívnym zanorovaním metódy sa postupne kontrolujú všetky cesty, ktoré je možné dosiahnuť z vybraného počiatočného stavu. Ak sú z počiatočného stavu skontrolované všetky možné cesty v grafe a v množine $Superf(Q)$ sa nachádzajú stavy, ktoré nie sú ani v množine *S*, ani neboli testované, tak sa následne skontroluje možnosť pridania týchto stavov do množiny *S*. Maximálne zanorenie rekurzívnej metódy je rovné počtu stavov, ktoré obsahuje výsledná množina *S* indukujúca maximálny acyklický podgraf.

2.3. *S*-REDUKCIA

Táto transformácia redukuje stavy automatu určené množinou $Superf(Q)$. S-redukovaný automat označme $S(A,q)$, kde *A* je pôvodný automat a $q \in Superf(Q)$ automatu *A*. $S(A,q)$ sa získa z automatu *A* potlačením stavu *q* a znovu definovaním všetkých hrán, ktoré obsahovali stav *q*. *S*-redukcia je znázornená na obrázku 1.



Obrázok 1: *S*-redukcia

Automat na obrázku 1 hore obsahuje jeden prebytočný stav *2*. Aplikovaním *S*-redukcie vznikne automat $S(A,2)$ znázornený na obrázku 1 dole. Z hrán $(1,a,2)$ a $(2,b,3)$ vznikla nová hrana

$(1, ab, 3)$. Automat $S(A,2)$ už neobsahuje ďalšie prebytočné stavy, z čoho vyplýva, že je *S*-neredukovateľný.

2.4. ZHRNUTIE ALGORITMU

Algoritmus popisovaný v tejto kapitole je možné opísať nasledovne:

1. Vypočítanie množiny prebytočných stavov $Superf(Q)$;
2. Vypočítanie maximálnej množiny stavov *S* z množiny prebytočných stavov $Superf(Q)$ tak, že *S* indukuje maximálny acyklický podgraf;
3. Aplikovanie S-redukcie na všetky stavy z množiny S.

2.5. DIFERENCIA S PÔVODNÝM ČLÁNKOM

V článku [1] je opísaná nie len *S*-redukcia, ale aj *I*-redukcia, ktorá redukuje nerozlíšiteľné stavy. Algoritmus túto redukciu neobsahuje, lebo odstránenie nerozlíšiteľných stavov prebieha pri minimalizácii deterministického konečného automatu. Podľa článku môže niekedy vzniknúť zmenou poradia redukcii automat s menším počtom stavov.

3. ZÁVER

Obsahom práce bolo navrhnutie a implementovanie algoritmu na konštrukciu deterministického zovšeobecneného automatu v jazyku Python a skúmanie problémov pri tom vzniknutých. Z popisu *S*-redukcie aplikovanej na konečný automat vyplýva, že deterministický zovšeobecnený automat má nižší počet stavov ako minimálny deterministický konečný automat, z ktorého vznikol. Tým je splnený hlavný ciel konštrukcie algoritmu, ktorým je zníženie pamäťových nárokov na uloženie automatu.

Model deterministického zovšeobecneného automatu je možné porovnať s modelom Delayed Input DFA (deterministic finite automaton), prípadne s inými modelmi, ktoré sú implementované v knižnici NetBench. Hlavnými porovnávajúcimi prvkami bude efektivita automatu a pamäťové nároky na uloženie automatu.

ACKNOWLEDGEMENT

This paper was created with the support of TeamIT - Building Competitive Research Teams in IT, CZ.1.07/2.3.00/09.0067.

REFERENCES

[1] Giammarresi, D., Montalbano, R.: Deterministic generalized automata. In Theoretical Computer Scineta 215. 1999. P. 191-208

[2] Puš, V., Tobola, J., Košař, V., Kaštil J., Kořenek J.: Netbench: Framework for Evaluation of Packet Processing Algorithms. In: *Symposium On Architecture For Networking And Communications Systems*. Los Alamitos, CA, USA: IEEE Computer Society, 2011, s. 95-96. ISBN 978-0-7695-4521-9.

[3] Křivka, Z., Masopust, T.: Grafové algoritmy. 2011. Skriptum FIT BUT

[4] Eilenberg, S.: Automata, Languages and Machines, Vol. A (Academic Press, 1974)