Porovnávanie regulárnych jazykov generovaných na základe regulárnych výrazov

Matúš Piroh

Slovensko

Úvod

Z teórie formálnych jazykov a automatov je známe, že každý regulárny výraz generuje nejaký jazyk. To je nejaká množina slov. Ak vezmeme dva regulárne výrazy,

ktoré hoci nie sú rovnaké, ale generujú rovnaký jazyk, hovoríme, že dané regulárne výrazy sú navzájom ekvivalentné. Pozrime sa na nasledujúcu dvojicu regulárnych výrazov:

$$a(a+b)$$

$$aa+ab$$

Obidva regulárne výrazy predstavujú jazyk $\{aa, ab\}$, teda sú navzájom ekvivalentné. Je to zrejmé aj tým, že druhý výraz dostaneme roznásobením zátvorky v prvom výraze.

Vezmime si ďalšie dva výrazy:

$$(a+ab)*aba*$$

$$a((ba)*a*)*ba*$$

Ak sa pozrieme na tieto výrazy a chceli by sme zistiť, či sú aj tieto dva ekvivalentné, vidíme, že tieto výrazy sú komplikovanejšie než predchádzajúce dva a nevieme to zistiť jednoduchým spôsobom „pozriem a vidím“. V skutočnosti aj tieto reťazce sú ekvivalentné. Ako to zistiť? Porovnanie dvoch regulárnych výrazov nie je až taká jednoduchá záležitosť. Štandardný postup ako spraviť toto porovnanie pozostáva zo štyroch krokov:

1. Prevod daných výrazov na automaty
2. Determinizácia automatov
3. Minimalizácia automatov
4. Porovnanie dvoch minimálnych automatov

Aj keď je tento algoritmus univerzálny, nie je veľmi praktický. Jeden z problémov môže spôsobovať samotná determinizácia. Je známe, že ak má regulárny výraz n znakov, nedeterministický automat bude mať $2n$ stavov, no po determinizácii môže počet stavov narásť až na hodnotu 2^{2n}. Teda ak by sme zobrali aj relatívne malý regulárny výraz, povedzme 20 znakový, nedeterministický automat by obsahoval 40 stavov a tak po determinizácii by automat mohol dosiahnuť až 2^{40} stavov a určite pracovať ručne s takýmito automatmi nie je jednoduchá záležitosť, práve naopak, je to veľmi nepraktické. Preto by bolo rozumné mať program, ktorý by dokázal riešiť tieto veci namiesto nás.

Ciele práce

1. Navrhnúť a implementovať vlastné riešenie na porovnávanie regulárnych výrazov
2. Preskúmať a zhodnotiť existujúce knižnice a programy určené na prácu s automatmi

Návrh a implementácia

Prevod výrazu na automat:

Implementovať prevod regulárneho výrazu na nedeterministický konečný automat nie je vôbec zložité. Stačí si len dobre navrhnúť štruktúru programu.

Ako abecedu sme zvolili všetky malé písmená, teda od *a* po *z*.

Celý automat si neskôr budeme pamätať v samostatnej triede, ktorá obsahuje zoznam stavov a referencie na počiatočný stav a na koncové stavy. Jednotlivé stavy budú obsahovať poradové číslo a prechody, teda zoznam stavov, kam sa dostanú na jednotlivé znaky.

Postupným čítaním daného výrazu sa rozhodujeme, o akú operáciu ide. Ak sú za sebou dva znaky, vieme že ide o zreťazenie a teda vo výslednom automate stavy ukladáme za sebou. Ak nájdeme znak +, vykonáme zjednotenie, teda v automate vzniknú dve nové vety. Ak sa vo výraze nachádza znak *, ide o uzáver, preto do automatu pridáme cyklus.

Takýmto spôsobom po prečítaní celého výrazu vznikne nedeterministický automat, ktorý prijíma jazyk generovaný regulárnym výrazom, ktorý sme dostali na vstupe.

Determinizácia automatu

Implementácia determinizácie je trochu zložitejšia než samotný prevod na automat. Celý princíp spočíva v tom, že začneme v počiatočnom stave, pozrieme sa na prechody a v novom automate vytvárame stavy, ktoré získame zjednotením stavov v starom automate. To opakujeme pre každý novovytvorený stav.

Pred determinizáciou si potrebujeme stavy nejako „pomenovať“, aby sme vedeli z ktorých stavov v nedeterministickom automate vznikli. Na to využijeme tzv. bitkódy. To znamená, že každý stav v pôvodnom automate si označíme bitovým číslom, ktoré obsahuje práve jednu jednotku, a pozícia tejto jednotky v bitkóde nám určuje poradové číslo daného stavu.

Toto označenie nám značne zjednoduší proces determinizácie a to tak, že nové stavy budeme vytvárať bitovým súčtom.

Ak už nepribudne žiadny nový stav a každý stav má vytvorený prechod na každý symbol abecedy, nad ktorou pracujeme, potom je determinizácia na konci a získame výsledný deterministický automat.

Minimalizácia automatu

Podstatou minimalizácie je nájsť ekvivalentné stavy. Začneme tým, že vytvoríme dve skupiny stavov: S1, obsahuje koncové stavy, S2 obsahuje nekoncové stavy.

Následne potrebujeme tabuľku, ktorá bude obsahovať informácie o prechodoch pre dané skupiny. Danú tabuľku môžeme chápať ako mapu, kde kľúče budú skupiny a hodnoty predstavujú zoznam stavov, ktorý je dlhý ako abeceda, nad ktorou pracujeme, teda každý prvok znamená prechod na príslušný symbol.

Začneme tým, že si stavy označíme podľa skupín, nie podľa poradových čísel. Postupnou iteráciou cez všetky skupiny pridávame každý stav do mapy podľa príslušnej skupiny a následne začneme mapu deliť. Potom nám stačí už len skontrolovať veľkosť mapy, a ak sa zmenila, znamená to, že pribudli nové skupiny. Preoznačíme všetky stavy podľa nových skupín, a opäť naplňujeme mapu. Tento cyklus opakujeme, kým veľkosť mapy neostane nezmenená.

Na koniec z výslednej mapy vytvoríme nový automat, a dodáme referencie na počiatočný a koncový stav z pôvodného automatu.

Porovnanie dvoch automatov

Pred samotným porovnaním je potrebné implementovať spomínaný proces normalizácie. Ten sa správa podobne ako prehľadávanie grafu. Na začiatku vynulujeme všetky označenia stavov v celom automate. Vojdeme do automatu počiatočným stavom, postupne prechádzame po hranách a zároveň si pamätáme, aké označenie sme použili naposledy. Ak sa dostaneme do stavu, v ktorom sme ešte neboli, označíme ho nasledujúcim číslom.

Ak sú už všetky stavy označené, normalizácia je na konci.

Nakoniec je potrebné porovnať automaty a to tak, že pre každý stav v prvom automate skontrolujeme, či majú stavy, do ktorých sa dostane na každý symbol rovnaké označenia ako stavy v druhom automate. Ak áno, môžeme prehlásiť, že automaty sú ekvivalentné a tým pádom sú ekvivalentné aj zadané regulárne výrazy, z ktorých sme dostali tieto automaty.

Záver

Po implementácii programu bolo vytvorené jednoduchú používateľské rozhranie, ktoré obsahuje textové políčka na zadanie regulárnych výrazov, výstup a zároveň zobrazenie automatov prislúchajúcich daným výrazov.

Po porovnaní s inými knižnicami určenými na prácu s automatmi pracuje náš program, resp. jednotlivé kroky algoritmu vždy v lepšom čase.

Zdroje

- J.E. Hopcroft: Formálne jazyky a automaty (2001)
- http://www.matematika.cz/regular-na-automat
- https://www.cambridge.org/core/journals/rairo-theoretical-informatics-and-applications/article/conver
- https://www.brics.dk/automaton/

Ústav informatiky

Prírodovedecká fakulta, Univerzita Pavla Jozefa Šafárika v Košiciach

Jesenná 5, 040 01 Košice,