Simulátor konečnostavového automatu

Róbert Novotný

Úvod

V tomto článku podáme návod na konštrukciu simulátora konečnostavového automatu – t. j. vytvoríme program, ktorý bude mať na vstupe inštrukcie konečnostavového automatu a vstup pre tento konečnostavový automat; na výstupe bude mať zase symboly, ktoré dá na výstup tento konečnostavový automat počas svojho behu. Simulátor naprogramujeme v skriptovacom jazyku PHP, najmä pre jeho jednoduchosť a niektoré vhodné vlastnosti (napr. slabé typovanie).

Teória

Na úvod uvedieme definíciu konečnostavového automatu.

Definícia. *Konečnostavovým automatom nazývame usporiadanú šesticu*

$$M = (Q, \Sigma, \Delta, f, g, q_I),$$

kde

$$\begin{aligned} Q &\dots \textit{konečná množina stavov,} \\ \Sigma &\dots \textit{vstupná abeceda,} \\ \Delta &\dots \textit{výstupná abeceda,} \\ f : Q \times \Sigma \to Q &\dots \textit{prechodová funkcia,} \\ g : Q \times \Sigma \to \Delta &\dots \textit{výstupná funkcia,} \\ q_I &\dots \textit{počiatočný stav} \end{aligned}$$

Konečnostavový automat si teda môžeme predstaviť ako zariadenie skladajúce sa z pásky, ktorá obsahuje vstup, riadiacej jednotky (alebo procesora) a hlavy, ktorá z pásky načítava jednotlivé symboly. Načítavanie symbolov prebieha výhradne zľava doprava, pričom je vždy načítaný len jeden symbol. Riadiaca jednotka sa môže nachádzať v rozličných stavoch (zodpovedá im množina Q). Medzi jednotlivými stavmi sa môže prepínať jednak podľa práve načítavaného symbolu a jednak podľa aktuálneho stavu.

Príklad. *Zostrojme konečnostavový automat, ktorý pri načítaní symbolu a dá na vstup symbol $-$ a pri každom druhom načítanom b dáva na výstup symbol $+$.*

Za vstupnú abecedu môžeme zobrať množinu $\Sigma = \{a, b\}$. Automat bude mať dva stavy, teda $Q = \{q_0, q_1\}$. Ďalej množina výstupných symbolov $\Delta = \{+, -\}$. Za počiatočný stav ešte vezmeme $q_I = q_0$.

Prechodovú funkciu f a výstupnú funkciu g definujeme pomocou tabuliek:

f	a	b
q_0	q_0	q_1
q_1	q_1	q_0

g	a	b
q_0	$-$	$+$
q_1	$-$	$+$

Konečnostavový automat môžeme znázorniť aj pomocou grafu – do jednotlivých vrcholov umiestnime stavy a hrany zorientujeme podľa toho, z ktorého stavu sa môžeme prepnúť do ktorého. Hrany navyše označíme symbolom, pri ktorého načítaní je možné prepnutie vykonať. Navyše pri hrane označíme symbol, ktorý bude zapísaný na výstup.

Prax

Konečnostavový automat sme implementovali ako objekt v jazyku PHP. Funkcie f a g sme implementovali pomocou jednej dvojrozmernej tabuľky (konkrétne ako poľa polí). Riadkové indexy tejto tabuľky tvoria symboly pásky, stĺpcové indexy sú tvorené stavmi. V políčku tabuľky sa nachádza usporiadaná dvojica $\langle \text{symbol}, \text{stav} \rangle$ (na tomto mieste sme v podstate tabuľky zlúčili).

Algoritmus je v podstate veľmi jednoduchý:

Typeset by plainTEX.

1. načítaj ďalší symbol z pásky a uvedom si, v akom stave sa práve nachádzaš (na začiatku je potrebné špecifikovať počiatočný stav q_I.
2. v tabuľke nájdi riadok, ktorého indexom je symbol pásky a stĺpec, ktorého indexom je názov aktuálneho stavu (jazyk PHP povoľuje aj reťazcové indexy polí).
   a) ak sa také políčko v tabuľke nenachádza, zastav (simulácia skončila)
   b) v opačnom prípade zober z usporiadanej dvojice, ktorá sa v políčku nachádza stav – do tohto stavu prepneme; a symbol – tento symbol vypíš na výstup. Vráť sa na krok 1.

Zdrojový kód v PHP

```
1  //-----------------------------------------------------
2  // Simulator konecnostavoveho automatu
3  //-----------------------------------------------------
4  // (C) Robert Novotny, novotnyr@skmi.science.upjs.sk
5  // 11. 5. 2003
6  //
7  // Vytvorene pre PrirF UPJS KE, KMI/LIN1.
8  //
9  // S problematikou automatov a formalnych jazykov
10 // sa mozno oboznamit vo vplyvnej monografii [Hu]
11 // (Hopcroft - Ullman: Formalne jazyky a automaty)
12
13 <?php
14   class fs_automaton {
15     //data
16     var $data;
17     //vystupna funkcia, implicitne echovany vystup
18     var $output_function = "";
19
20     //konstruktor, zatial prazdny
21     function fs_automaton() {
22     }
23
24     //implicitna vystupna funkcia, mozno predefinovat v clene $outputfunction
25     function default_output_function($symbol) {
26       echo $symbol;
27     }
28
29     //osetrovac vystupu, podla toho, ci uzivatel zadal vlastnu vystupnu funkciu
30     function output($symbol) {
31       if($this->output_function == "") {
32         call_user_method("default_output_function", $this, $symbol);
33       } else {
34         call_user_func($this->output_function, $symbol);
35       }
36     }
37
38     //vyhlada nasledovnu instrukciu
39     function find_instruction($symbol, $state) {
40      //vrati dvojicu (vystupny_symbol, dalsi_stav) podla dat
41       return array(
42         "symbol" => $this->data[$symbol][$state][1],
43         "state" => $this->data[$symbol][$state][0]
44       );
45     }
46
```

```
47      //spusti zadany automat na danom vstupe $input, pricom $init_state
48      //je pociatocny stav
49      function run($input, $init_state) {
50        $state = $init_state;
51        //nacitavame vstup
52        for($i = 0; $i < strlen($input); $i++) {
53          //poznacime si aktualne citany symbol
54          $symbol = $input[$i];
55          //najdeme dalsi stav a vystupny symbol (find_instruction
56          //zodpoveda funkcii <i>f</i>
57          //v definicii KSA
58          $next_instruction = $this->find_instruction($symbol, $state);
59          //vypiseme symbol
60          $this->output($next_instruction["symbol"]);
61          //sme v dalsom stave, v dalsom kole nacitame znak atd...
62          $state = $next_instruction["state"];
63        }
64      }
65    }
66
67    //priklad pouzitia (na <i>a</i> dava minus, a pri kazdom druhom <i>b</i> dava +
68    $ksa = new fs_automaton();
69    $ksa->data["a"]["q_0"] = array("q_0", "-");
70    $ksa->data["a"]["q_1"] = array("q_1", "-");
71    $ksa->data["b"]["q_0"] = array("q_1", "-");
72    $ksa->data["b"]["q_1"] = array("q_0", "+");
73
74    $ksa->run("aaababaa", "q_0");
75  ?>
```

Literatúra

1. Hopcroft, Ullmann – Formálne jazyky a automaty, Alfa, Bratislava 1978
2. Oficiálna stránka jazyka PHP – `http://www.php.net`