Automātu teorija - 1. mājas darbs

2023-10-30

Gunārs Ābeltiņš

1. uzdevums

(a) apakšuzdevums

Izvade: 001001001000010010001

(b) apakšuzdevums

Atrastā ievade: 000100100000110110110

Atrastā Izvade: 100010010000011011011

Heminga attālums: 3

2. uzdevums

Transformators (Q, X, Y, f, g, q_0) ir definēts sekojoši:

Q = {s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_10, s_11, s_12}

X = {0, 1}

Y = {0, 1, 2}

q_0 = s_1

Funkcijas f un g ir dotas ar tabulu:

Stāvoklis q	Ieeja x	f(q,x)	g(q,x)
s_1	0	s_2	0
s_1	1	s_1	0
s_2	0	s_3	0
s_2	1	s_1	0
s_3	0	s_6	1
s_3	1	s_1	0
s_4	0	s_5	1
s_4	1	s_4	1
s_5	0	s_6	1
s_5	1	s_4	1
s_6	0	s_9	2
s_6	1	s_4	1
s_7	0	s_8	2
s_7	1	s_7	2
s_8	0	s_9	2
s_8	1	s_7	2
s_9	0	s_3	0

s_9	1	s_7	2

3. uzdevums

(a) apakšuzdevums

Valoda ir 21 vārds ar garumu <= 5: 1, 000, 001, 101, 111, 0100, 0101, 00001, 00011, 00101, 00111, 01100, 01101, 10000, 10001, 10101, 10111, 11000, 11001, 11101, 11111

(b) apakšuzdevums

$(1 \vee (01^*(0 \vee 1)))((0 \vee 1)(1 \vee (01^*0(0 \vee 1))))^*$

4. uzdevums

Akceptors (Q, X, f, Q_A, q_0) ir definēts sekojoši:

$Q = \{s\_1, s\_2, s\_3\}$

$X = \{0, 1\}$

$Q_A = \{s\_2\}$

$q_0 = s\_1$

Funkcija f ir dota ar tabulu:

Stāvoklis q	Ieeja x	f(q,x)
s_1	0	s_2
s_1	1	s_3
s_2	0	s_2
s_2	1	s_1
s_3	0	s_3
s_3	1	s_3

```cpp
#include <iostream>

#define I 2 % 2
#define J 0 % 2
#define K 0 % 2
#define M 8 % 2

using namespace std;

string transform(int state, string input)
{
    if (input.size() == 0)
        return "";

    switch (state)
    {
    case 1:
        if (input[0] == '0') return (char)('1' - M) + transform(3, input.substr(1));
        else    return (char)('0' + J) + transform(3 - I, input.substr(1));
    case 2:
        if (input[0] == '0') return (char)('1') + transform(3 - M, input.substr(1));
        else return (char)('1' - J) + transform(1, input.substr(1));
    case 3:
        if (input[0] == '0') return (char)('0') + transform(3 - K, input.substr(1));
        else return (char)('0' + M) + transform(2, input.substr(1));
    }
}

int hemming_distance(string a, string b)
{
    int distance = 0;
    for (int i = 0; i < a.size(); i++)
        if (a[i] != b[i]) distance++;
    return distance;
}

string find_closest(string output)
{
    string ans;
    ans.resize(output.size(), '0');

    auto execute = [&output, &ans](auto &&execute, string str, int n)
    {
        if (n == 0)
        {
            if (hemming_distance(transform(1, str), output) < hemming_distance(transform(1, ans), output))
                ans = str;
            return;
        }

        execute(execute, str + "0", n - 1);
        execute(execute, str + "1", n - 1);
    };
    execute(execute, "", output.size());

    return ans;
}

int main()
{
    string input = "110010010000111111011";
    cout << "Input:  " << input << endl;
    cout << "Output: " << transform(1, input) << endl;
    cout << "---------------------------" << endl;
    string output = "110010010000111111011";
    string ans = find_closest(output);
    cout << "Given Output: " << output << endl;
    cout << "Found Output: " << transform(1, ans) << endl;
    cout << "Found Input:  " << ans << endl;
    cout << "Distance: " << hemming_distance(output, transform(1, ans)) << endl;
}
```

```cpp
    #include <iostream>

#define I 2 % 2
#define J 0 % 2
#define K 0 % 2
#define M 8 % 2

using namespace std;

bool accept(int state, string input)
{
    if (input.size() == 0)
        if (state == 2 - M || state == 2 - J)
            return true;
        else
            return false;

    switch (state)
    {
    case 1:
        if (input[0] == '0')
            return accept(3, input.substr(1));
        else
            return accept(2 - K, input.substr(1));
    case 2:
        if (input[0] == '0')
            return accept(1, input.substr(1));
        else
            return accept(I + 1, input.substr(1));
    case 3:
        if (input[0] == '0')
            return accept(4, input.substr(1));
        else
            return accept(3 - M, input.substr(1));
    case 4:
        if (input[0] == '0')
            return accept(J + 2, input.substr(1));
        else
            return accept(2, input.substr(1));
    }

    return false;
}

void print_accepted(int max_len = 5)
{
    auto execute = [](auto &&execute, string str, int n)
    {
        if (n == 0)
        {
            if (accept(1, str))
                cout << str << endl;

            return;
        }

        execute(execute, str + "0", n - 1);
        execute(execute, str + "1", n - 1);
    };

    for (int i = 1; i <= max_len; i++)
        execute(execute, "", i);
}

int main()
{
    print_accepted();
}
```