Институт информационных технологий

Кафедра: Математическое и программное обеспечение ЭВМ

Дисциплина: Теория автоматов и формальных языков

ЛАБОРАТОРНАЯ РАБОТА № 3-4

Тема: Регулярные множества и регулярные выражения

Выполнил:

студент гр. 1ПИб-02-2оп-23

Кринкин Олег Алексеевич

Проверил:

доцент, к.т.н. Ганичева Оксана Георгиевна

ЗАДАНИЕ

Выполнить задания из учебника Карпов Ю.Г. «Теория автоматов»:

1. Построить конечный автомат с входным алфавитом , распознающий:

е) все цепочки, заканчивающиеся кодом aabba;

ж) все цепочки, в которых за каждым a непосредственно следует b.

1. Построить конечный автомат с входным алфавитом , распознающий:

а) все цепочки, в которых за каждым a когда-нибудь в будущем следует b;

б) все цепочки, в которых две последние буквы не совпадают;

в) все цепочки, начинающиеся и заканчивающиеся различными символами;

г) все цепочки, включающие по крайней мере один символ a и один символ b.

1. Построить автоматы, распознающие языки, задаваемые регулярными выражениями:

.

1. Построить регулярное выражение, задающее множество всех таких слов над словарем , в которых за символом b

а) обязательно стоит символ c

б) не может стоять символ c. Построить конечные автоматы, распознающие соответствующие языки.

1. Построить регулярное выражение, которое определяет язык, соответствующий вашему варианту (см таблицу, вариант по списку группы). Для заданного регулярного выражения построить детерминированный конечный автомат. На основе конечного автомата написать программу для распознавания строк, принадлежащих языку, определяемому регулярным выражением.

|  |  |  |
| --- | --- | --- |
| 30 | 1, 0 | Множество слов, в которых присутствует по крайней мере один символ «0» и один символ «1» |

ХОД РАБОТЫ

Задание 1:

Конечный автомат, распознающий строки по регулярным выражениям, состоит из:

КА= (Х, S, S0, F, ∂)

X – конечное непустое множество входных символов.

S – конечное непустое множество состояний.

S0 – начальное состояние.

F <= множество выделенных заключительных состояний.

∂ - программа автоматов, функция переходов.

e) Все цепочки, заканчивающиеся кодом aabba:

X – {a, b}

S – {S0, S1, S2, S3, S4, S5}

S0 – начальное состояние.

F={S5}

∂ - (S, X) -> S’

(S0, a+b) -> S0, (S0, a) -> S1, (S1, a) -> S2, (S2, b) -> S2, (S2, b) -> S3,

(S3, b) -> S4, (S4, a) -> S5.

R = (a+b)\*aabba (рис. 1)

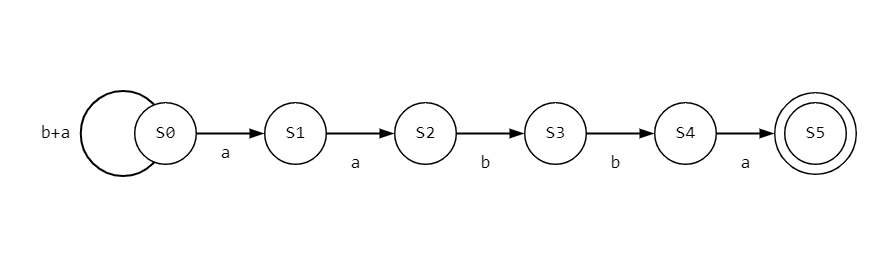


Рис. . Автомат R = (a+b)\*aabba

ж) Все цепочки, в которых за каждым a непосредственно следует b:

X – {a, b}

S – {S0, S1, S2}

S0 – начальное состояние.

F={S2}

∂ - (S, X) -> S’

(S0, b) -> S0, (S0, a) -> S1, (S1, b) -> S2, (S2, b) -> S0

R = (b\*+(ab)\*+b\*)\* (рис. 2)

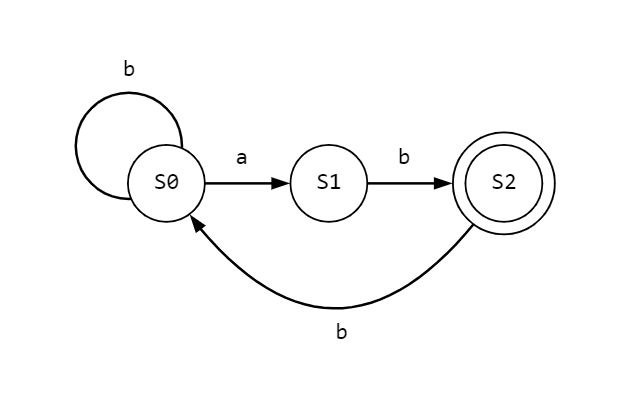


Рис. . Автомат R = (b\*+(ab)\*+b\*)\*

Задание 2:

а) Все цепочки, в которых за каждым a когда-нибудь в будущем следует b:

X – {a, b, c}

S – {S0, S1, S2, S3, S4}

S0 – начальное состояние.

F={S4}

∂ - (S, X) -> S’

(S0, a+b+c) -> S0, (S0, a) -> S1, (S1, a+b+c) -> S2, (S2, b) -> S3, (S3, a+b+c) -> S4, (S4, a+b+c) -> S4.

R = (a+b+c)\*a(a+b+c)\*b(a+b+c)\* (рис. 3)

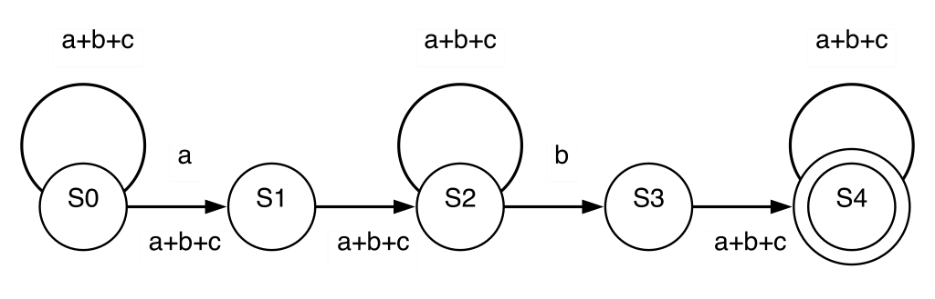


Рис. 3. Автомат R = (a+b+c)\*a(a+b+c)\*b(a+c+b)\*

б) все цепочки, в которых две последние буквы не совпадают:

X – {a, b, c}

S – {S0, S1, S2, S3, S4}

S0 – начальное состояние.

F={S4}

∂ - (S, X) -> S’

(S0, a) -> S0, (S0, c) -> S0, (S0, b) -> S2, (S1, a) -> S4, (S2, a) -> S4

(S0, b) -> S0, (S0, c) -> S1, (S0, a) -> S3, (S1, b) -> S4, (S2, c) -> S4

(S3, b) -> S4, (S3, c) -> S4

R = (a+b+c)\*(a(b+c)+b(a+c)+c(a+b)) (рис. 4)

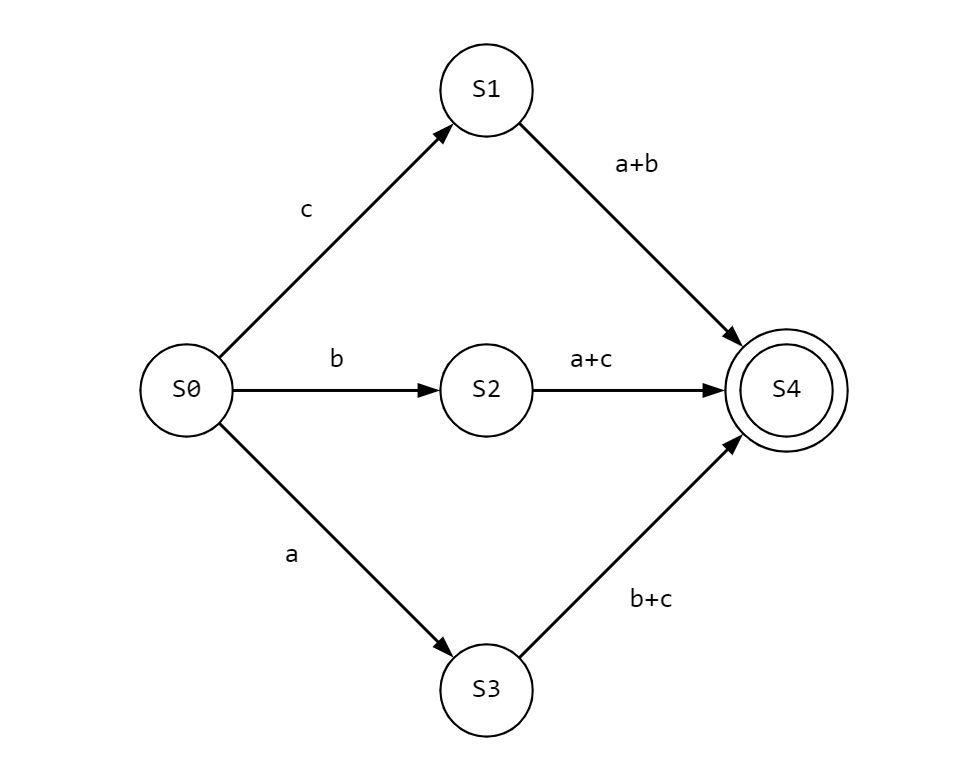


Рис. 4. Автомат R = (a+b+c)\*(a(b+c)+b(a+c)+c(a+b))

в) Все цепочки, начинающиеся и заканчивающиеся различными символами:

X – {a, b, c}

S – {S0, S1, S2, S3, S4}

S0 – начальное состояние.

F={S4}

∂ - (S, X) -> S’

(S0, a) -> S0, (S0, c) -> S0, (S0, b) -> S0, (S1, b) -> S1, (S1, a) -> S1, (S1, c) -> S1, (S0, a) -> S1, (S0, b) -> S2, (S0, c) -> S3, (S1, b) -> S4, (S1, c) -> S4, (S2, b) -> S2, (S2, a) -> S2, (S2, c) -> S2, (S2, a) -> S4, (S2, c) -> S4, (S3, b) -> S3, (S3, a) -> S3, (S3, c) -> S3, (S3, b) -> S4, (S3, a) -> S4

R = (a(a+b+c)\*(b+c)+b(a+b+c)\*(a+c)+c(a+b+c)\*(a+b)) (рис. 5)

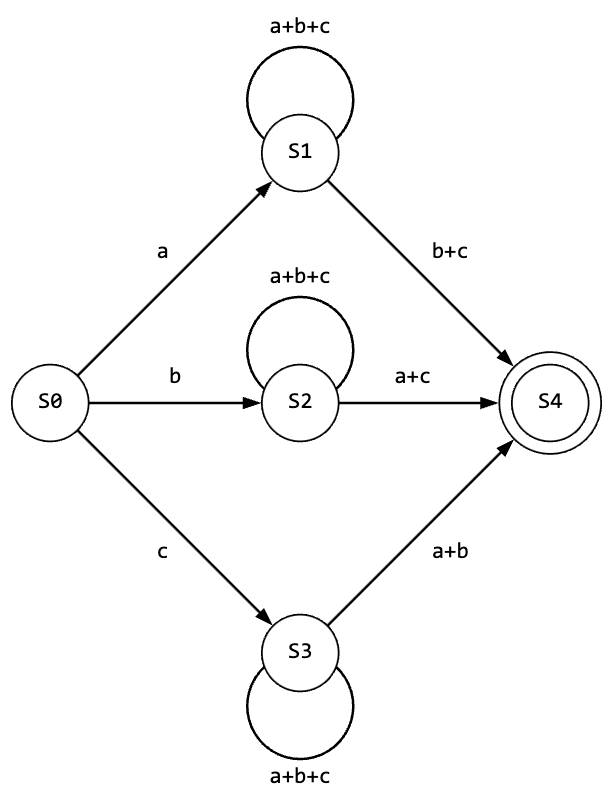


Рис. 5. Автомат R = (a(a+b+c)\*(b+c)+b(a+b+c)\*(a+c)+c(a+b+c)\*(a+b))

г) Все цепочки, включающие по крайней мере один символ a и один символ b:

X – {a, b, c}

S – {S1, S2, S3, S4}

S1 – начальное состояние.

F={S3}

∂ - (S, X) -> S’

(S1, a+b+c) -> S1, (S1, a) -> S2, (S1, b) -> S4, (S2, a+b+c) -> S2, (S4, a+b+c) -> S4, (S4, a) -> S3, (S2, b) -> S3, (S3, a+b+c) -> S3.

R=((a+b+c)\*a(a+b+c)\*b(a+b+c)\*)+((a+b+c)\*b(a+b+c)\*a(a+b+c)\*) (рис. 6):

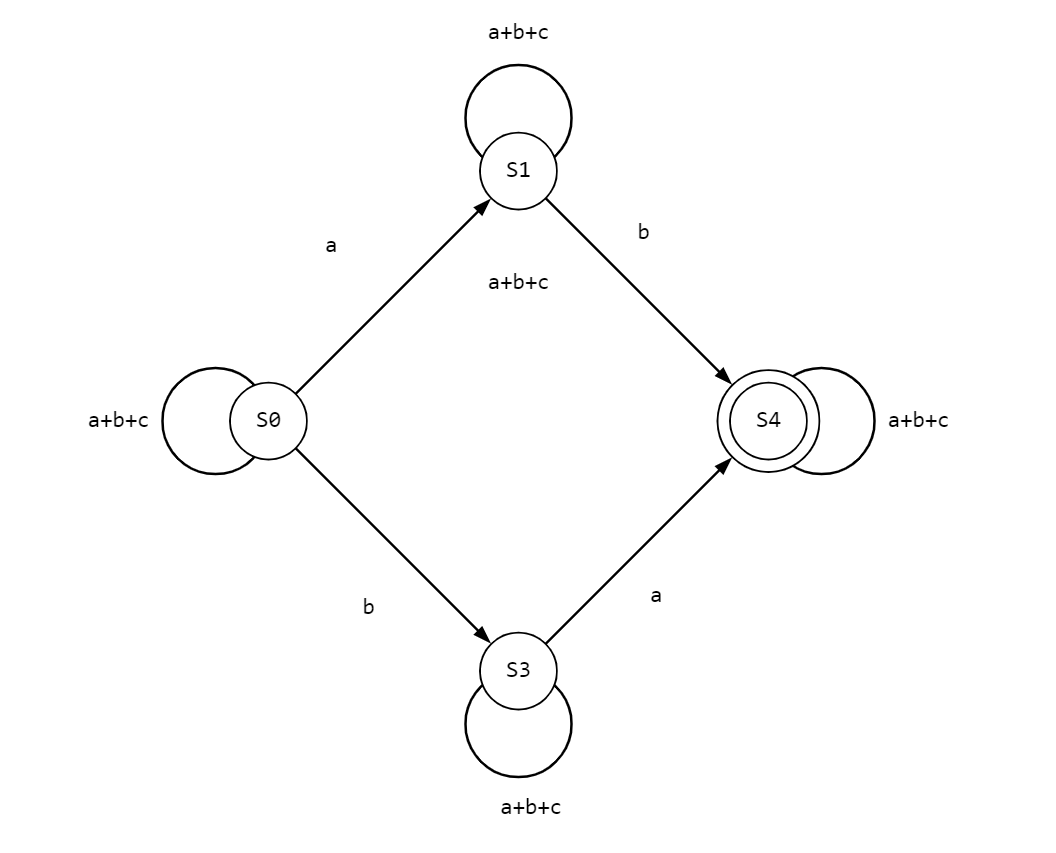


Рис. 6. Автомат

R = ((a+b+c)\*a(a+b+c)\*b(a+b+c)\*)+((a+b+c)\*b(a+b+c)\*a(a+b+c)\*)

Задание 3:

X – {a, b}

S – {S0, S1}

S0 – начальное состояние.

F={S1}

∂ - (S, X) -> S’

(S0, a) -> S0, (S0, b) -> S1

R = a\*b (рис. 7)

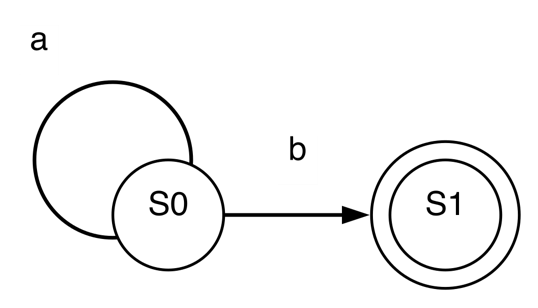


Рис. . Автомат R = a\*b

X – {a, b}

S – {S0}

S0 – начальное состояние.

F={S0}

∂ - (S, X) -> S’

(S0, a) -> S0

R = a\*a\* (рис. 8)

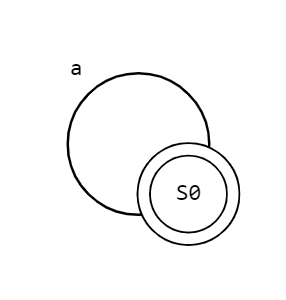


Рис. 8. Автомат R = a\*a\*

X – {a, b}

S – {S0, S1}

S0 – начальное состояние.

F={S1}

∂ - (S, X) -> S’

(S0, a) -> S0, (S0, b) -> S1, (S1, b) -> S1

R = a\*+b\* (рис. 9)

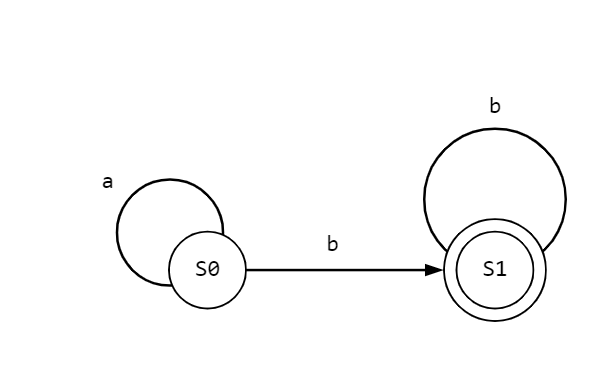


Рис. . Автомат R = a\*+b\*

X – {a, b}

S – {S0, S1}

S0 – начальное состояние.

F={S1}

∂ - (S, X) -> S’

(S0, a) -> S0, (S0, b) -> S1

R = (a+b)\* ; R = (a\*b\*)\* ; R = (a\*+b\*)\* (рис. 10)

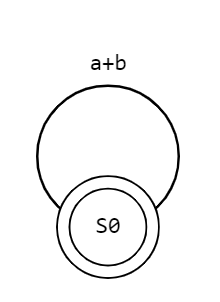


Рис. 10. Автомат R = (a+b)\*; R = (a\*b\*)\* ; R = (a\*+b\*)\*\*

X – {a, b}

S – {S0, S1}

S0 – начальное состояние.

F={S1}

∂ - (S, X) -> S’

(S0, a) -> S0, (S0, b) -> S1, (S1, b) -> S1,

(S1, a) -> S2

R = a\*b+b\*a (рис. 11)

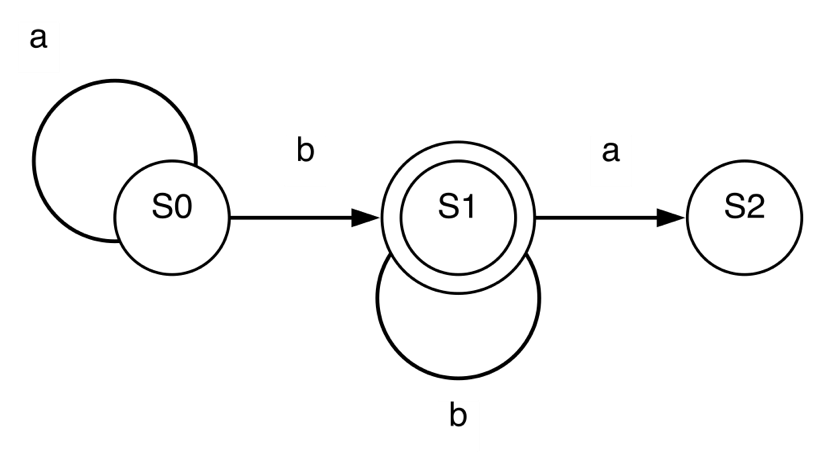


Рис. 11. Автомат R = a\*b+b\*a

Задание 4:

а) X – {a, b, c}

S – {S0, S1}

S0 – начальное состояние.

F={S1}

∂ - (S, X) -> S’

(S0, a) -> S0, (S0, с) -> S0, (S0, с) -> S1, (S1, b) -> S0

R = (a+c+bc)\* (рис. 12)

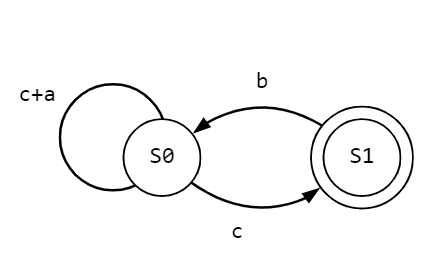


Рис. 12. Автомат R = (a+c+bc)\*

б) X – {a, b, c}

S – {S0, S1}

S0 – начальное состояние.

F={S1}

∂ - (S, X) -> S’

(S0, a) -> S0, (S0, с) -> S0, (S0, b) -> S1, (S1, b) -> S1, (S1, a) -> S1,

(S1, a) -> S0

R = (a\*c\*)\*(b\*a\*)\* (рис. 13)

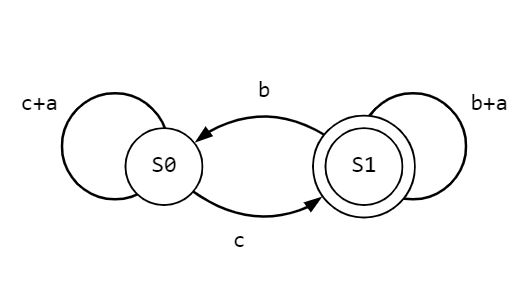


Рис. 13. Автомат R = (a\*c\*)\*(b\*a\*)\*

Задание 5:

1. По заданию из варианта построено регулярное выражение:

X – {0, 1}

S – {S0, S1, S2, S3}

S0 – начальное состояние.

F={S3}

∂ - (S, X) -> S’

(S0, 0) -> S1, (S0, 1) -> S2, (S1, 0) -> S1, (S1, 1) -> S3, (S2, 0) -> S3,

(S2, 1) -> S2, (S3, 1) -> S3, (S3, 0) -> S3

R = (0+1)\*(0+1)(0+1)\*(0+1)(0+1)\* (рис. 14)

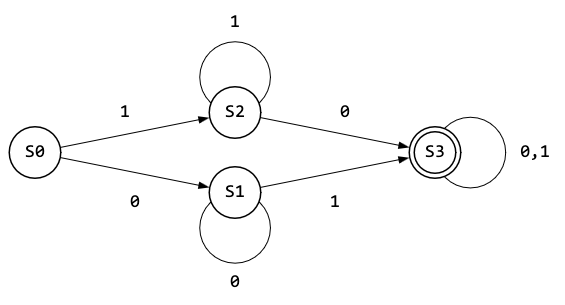


Рис. 14. Автомат R = (0+1)\*(0+1)(0+1)\*(0+1)(0+1)\*

1. По описанному автомату спроектирована и написана программа, принимающая на вход последовательность и анализирующая её на правильность, попутно строя на экране таблицу протокола.
2. Для написанной программы составлена следующая структурная схема (рис. 15):

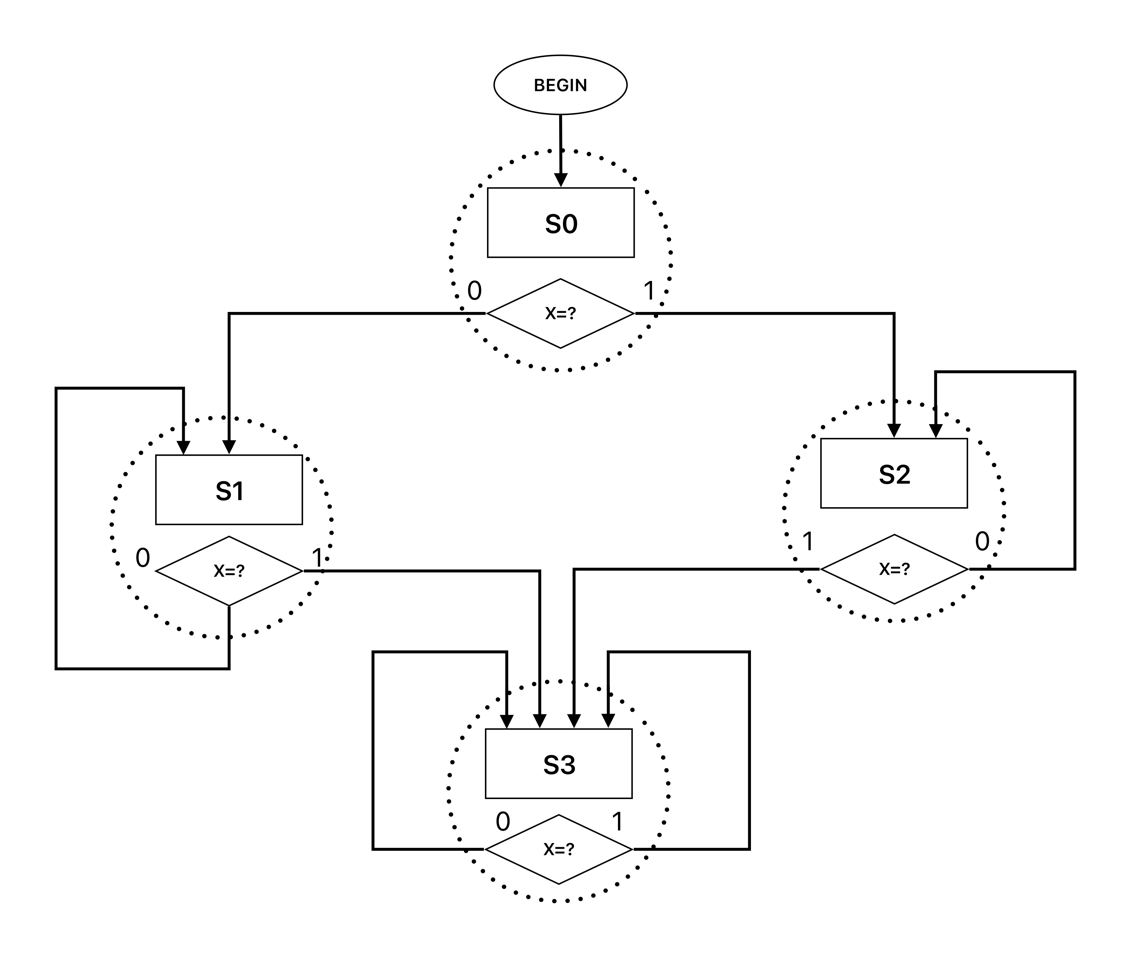


Рис. 15. Структурная схема программы

1. При вводе слова 11100 программа определяет его как верное, т.к. моделируемый автомат перешёл в состояние S3 (рис. 16):

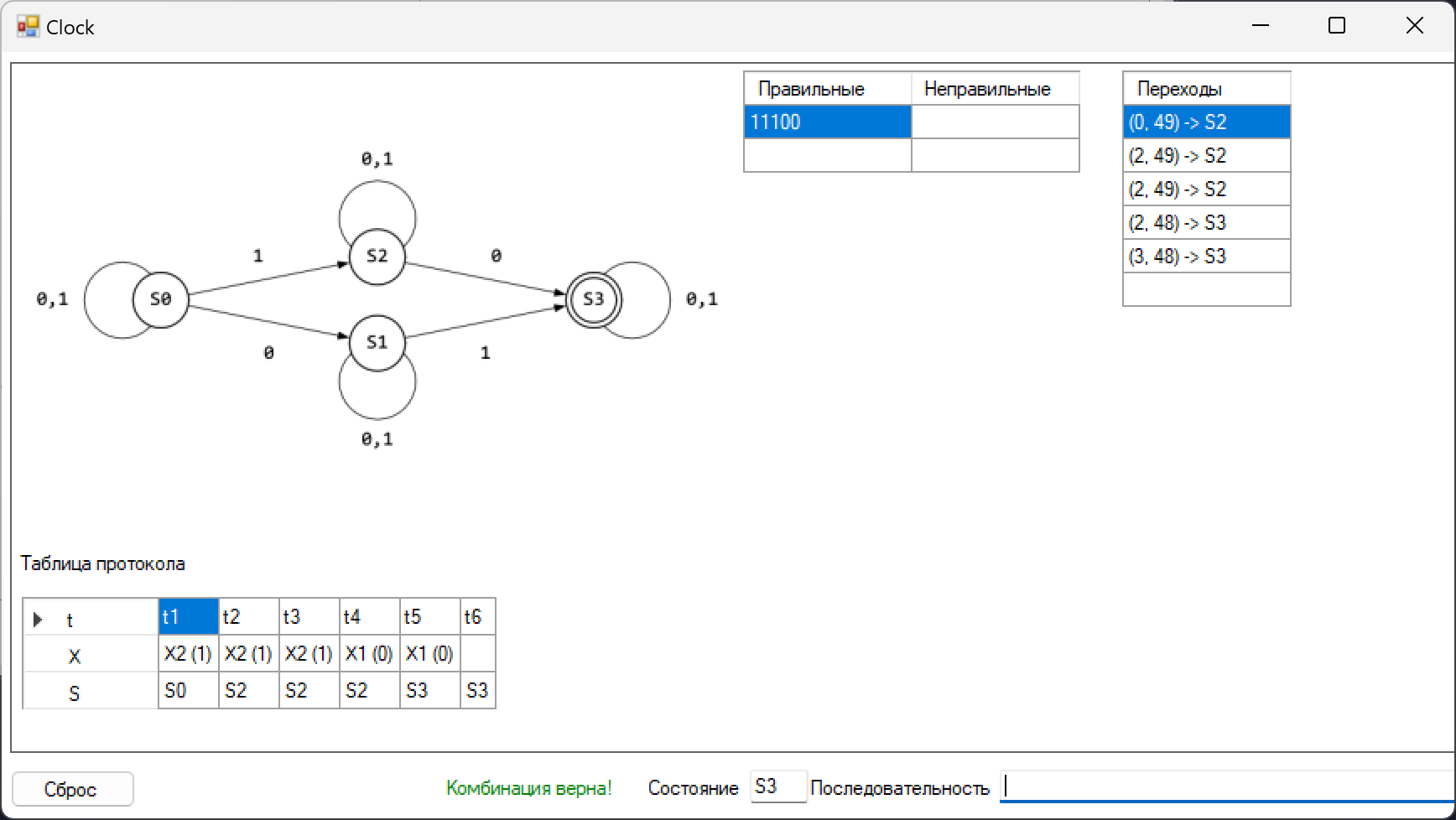


Рис. 16. Результат работы комбинации 11100

1. При вводе слова 111 программа определяет его как неверное, т.к. моделируемый автомат остался в состоянии S2 не встретив символ 0 (рис. 17):

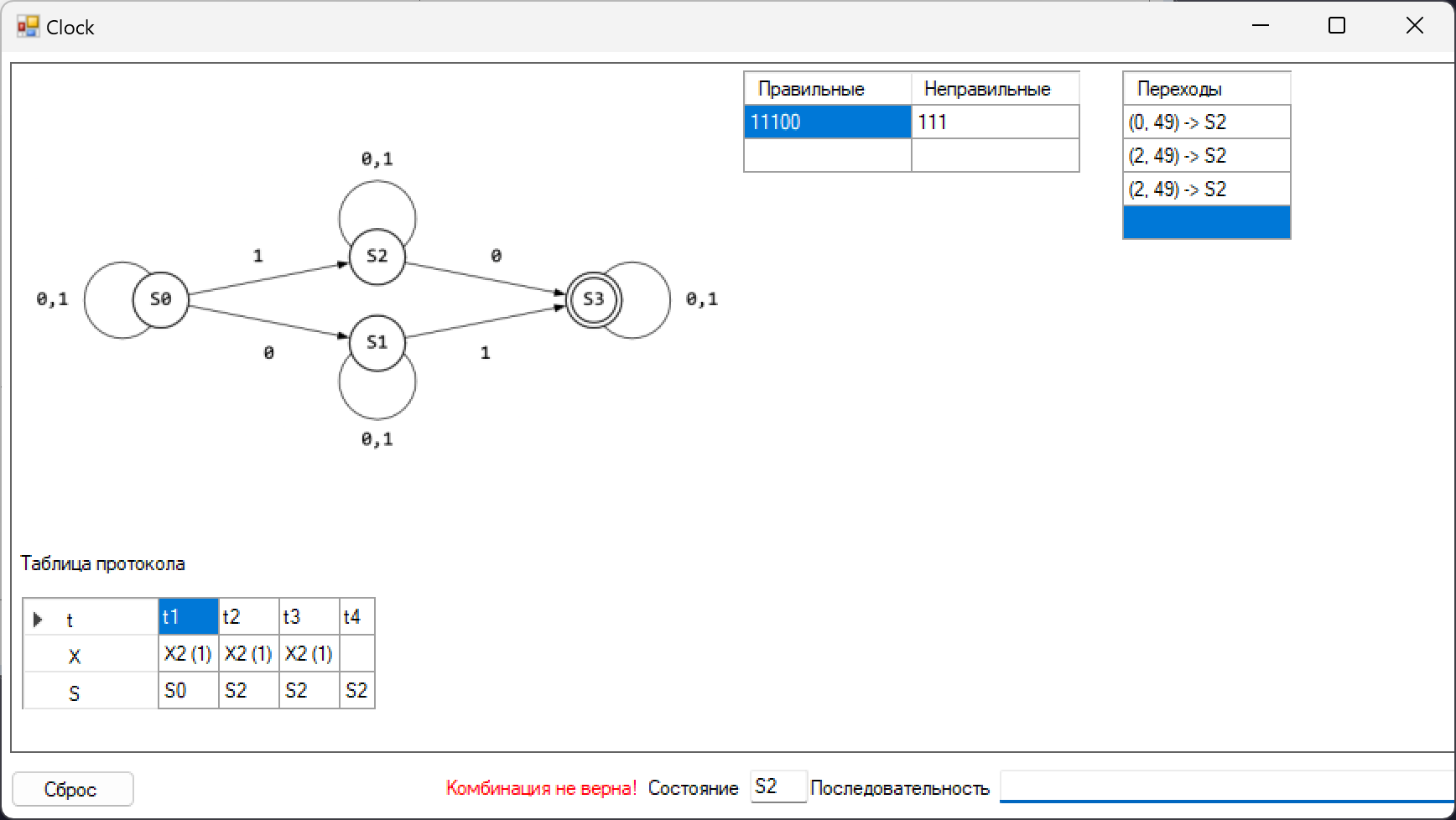


Рис. 17. Результат работы при комбинации 111

1. При вводе слова 10 программа определяет его как верное, т.к. моделируемый автомат перешёл в состояние S3 (рис. 18):

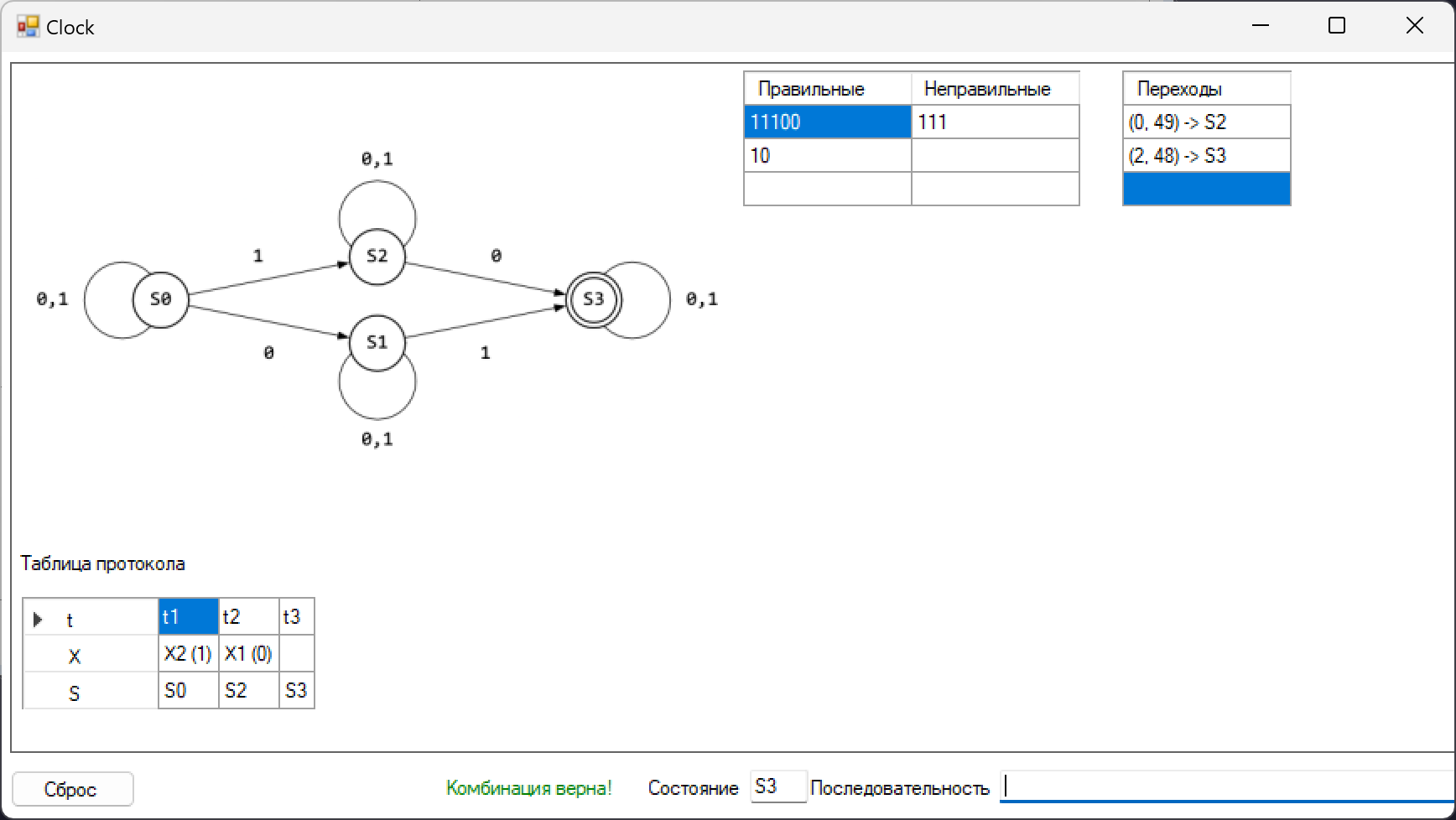


Рис. 18. Результат работы при комбинации 10

1. При вводе последовательности слов 110 1 0 10 011 программа переходит в автоматический режим и распределяет правильные и неправильные слова по таблицам (рис. 19):

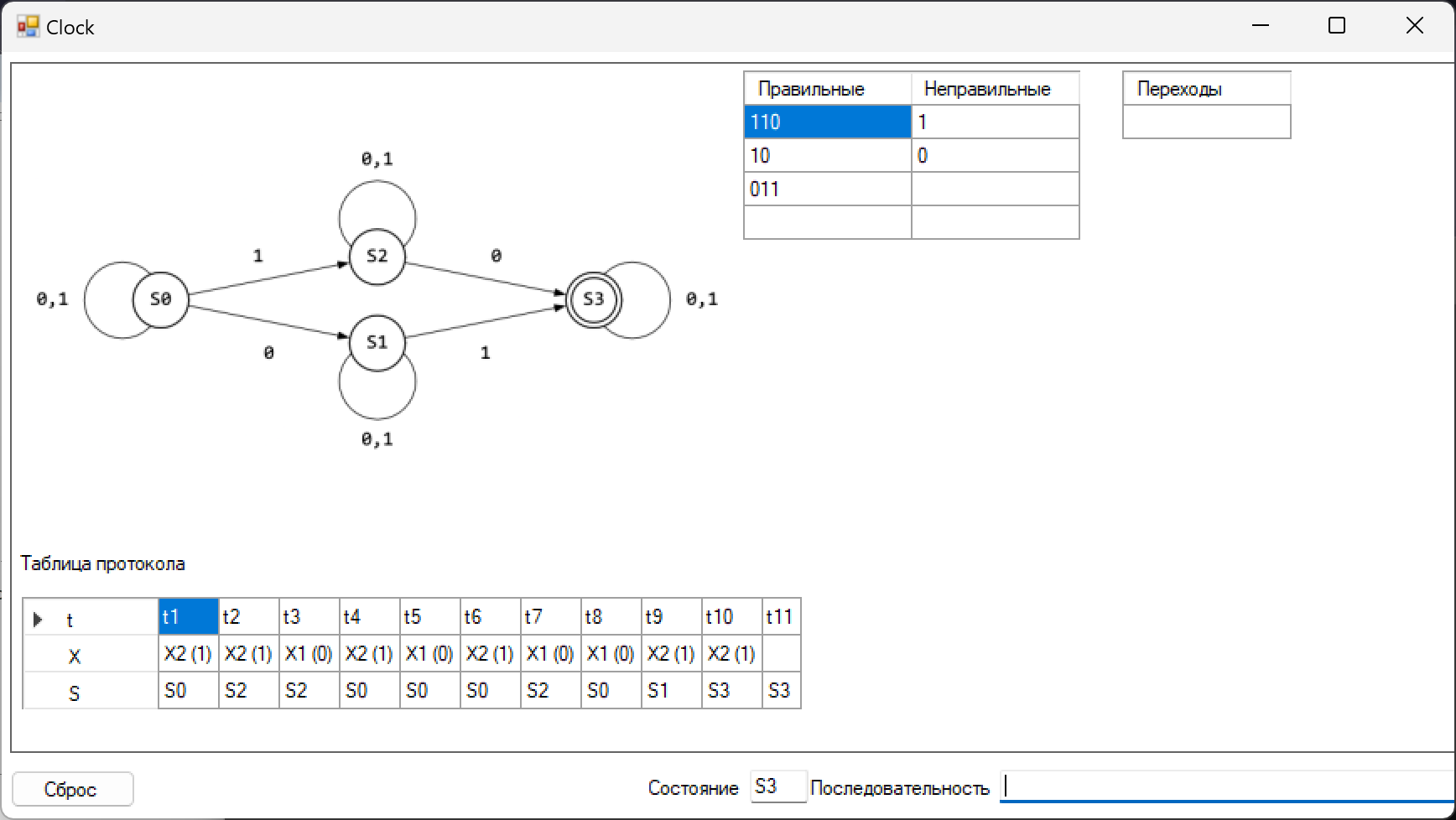


Рис. 19. Результат работы в автоматическом режиме

1. Для программы построена следующая блок-схема (рис. 20):

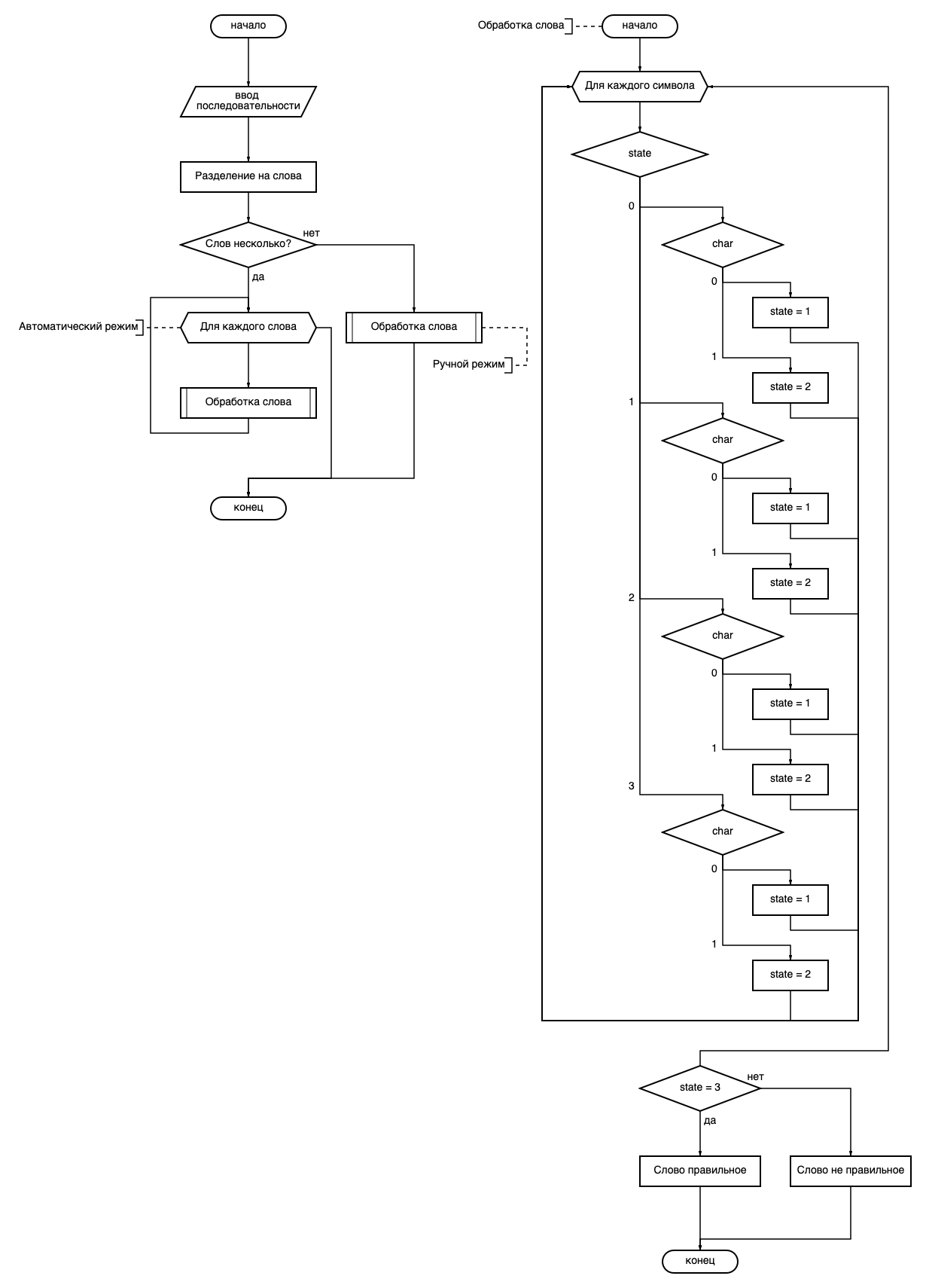


Рис. 20. Блок-схема программы

РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ РАБОТЫ

В результате выполнения лабораторной работы получен конечный автомат, обрабатывающий слова в соответствии с регулярным выражением. Также была разработана программа (рис. 21), моделирующая работу автомата.

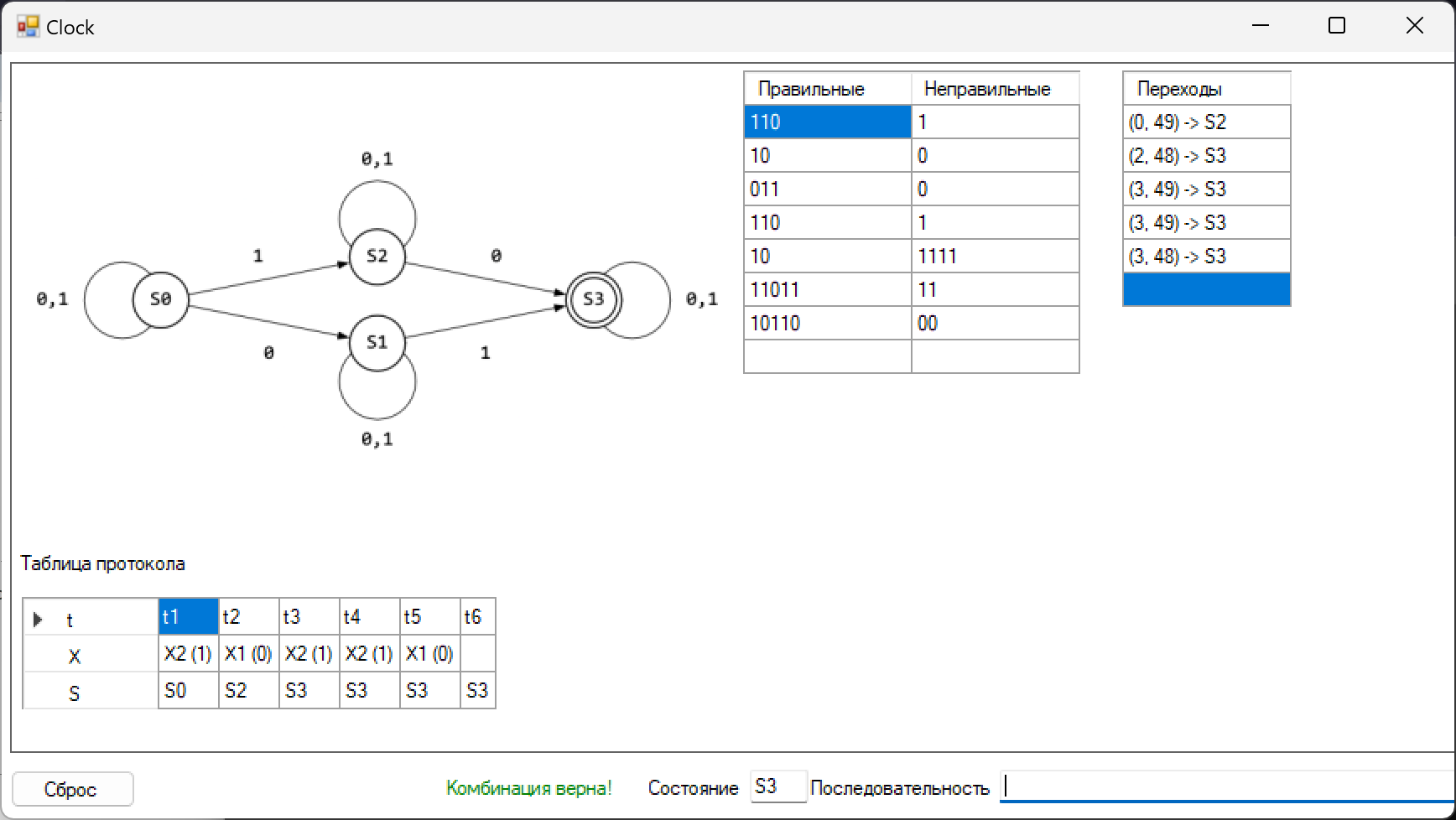


Рис. 21. Окно программы

Часть программного кода спроектированной программы:

private:

int s = 0;

int t = 0;

int right = 0;

int n\_right = 0;

bool manual = false;

System::String^ GetS() {

return "S" + s;

}

System::Void Reset() {

/\*

\* Функция, возвращающая программу в первоначальное состояние

\*/

s = 0; t = 0; right = 0; n\_right = 0;

stateBox->Text = "S0";

commandBox->Text = "";

combTextBox->Text = "";

ClearTable(this->dataGridViewProtocol);

ClearTable(this->dataGridViewSequence);

this->dataGridViewProtocol->ColumnCount = 1;

this->dataGridViewSequence->RowCount = 1;

}

System::Void ClearTable(DataGridView^ table) {

/\*

\* Функция, очищающая ячейки таблицы

\*/

for (int i = 0; i < table->RowCount; i++)

for (int j = 0; j < table->ColumnCount; j++)

table->Rows[i]->Cells[j]->Value = "";

}

System::Void AddCell(char command, int state, int proto\_len) {

/\*

\* Функция, добавляющая новую ячейку в каждую таблицу по команде

\*/

switch (command) {

case '0': { this->dataGridViewProtocol->Rows[1]->Cells[proto\_len - 1]->Value = "X1 (0)"; break; }

case '1': { this->dataGridViewProtocol->Rows[1]->Cells[proto\_len - 1]->Value = "X2 (1)"; break; }

}

// Выполняется если символ допустимый

this->dataGridViewProtocol->Rows[0]->Cells[proto\_len - 1]->Value = "t" + t;

this->dataGridViewProtocol->Rows[2]->Cells[proto\_len - 1]->Value = "S" + state;

this->dataGridViewProtocol->Rows[0]->Cells[proto\_len]->Value = "t" + (t + 1);

this->dataGridViewProtocol->Rows[2]->Cells[proto\_len]->Value = GetS();

if (manual) {

int count = this->dataGridViewTrans->RowCount++;

this->dataGridViewTrans->Rows[count - 1]->Cells[0]->Value = "(" + state + ", " + command + ") -> " + GetS();

}

}

System::Void ProcessCommand(char command) {

/\*

\* Функция, обрабатывающая следующее состояние автомата по команде

\*/

int prev\_s = s;

switch (s) {

case 0: {

switch (command) {

case '0': { s = 1; break; };

case '1': { s = 2; break; };

}

break;

}

case 1: {

switch (command) {

case '0': { s = 1; break; };

case '1': { s = 3; break; };

}

break;

}

case 2: {

switch (command) {

case '0': { s = 3; break; };

case '1': { s = 2; break; };

}

break;

}

case 3: {

switch (command) {

case '0': { s = 3; break; };

case '1': { s = 3; break; };

}

break;

}

}

t++;

int proto\_len = this->dataGridViewProtocol->ColumnCount++;

AddCell(command, prev\_s, proto\_len);

stateBox->Text = "S" + s;

}

System::Void ProcessLexem(String^ lexem) {

for (int i = 0; i < lexem->Length; i++) {

if (System::Char::ToLower(lexem[i]) == '0' ||

System::Char::ToLower(lexem[i]) == '1') {

// Вызов функции обработки каждого символа последовательности

ProcessCommand(System::Char::ToLower(lexem[i]));

}

}

if (s == 3) {

right++;

dataGridViewSequence->RowCount = std::max({ right, n\_right })+1;

dataGridViewSequence->Rows[right-1]->Cells[0]->Value = lexem;

}

else {

n\_right++;

dataGridViewSequence->RowCount = std::max({ right, n\_right })+1;

dataGridViewSequence->Rows[n\_right-1]->Cells[1]->Value = lexem;

}

}

System::Void Clock\_Load(System::Object^ sender, System::EventArgs^ e) {

/\*

\* Функция, вызывающаяся при запуске программы

\*/

this->pictureBox2->Image = Image::FromFile("scheme.png");

this->dataGridViewProtocol->ColumnCount = 1;

this->dataGridViewProtocol->Rows->Add(3);

this->dataGridViewProtocol->Rows[0]->HeaderCell->Value = "t";

this->dataGridViewProtocol->Rows[1]->HeaderCell->Value = "X";

this->dataGridViewProtocol->Rows[2]->HeaderCell->Value = "S";

this->dataGridViewSequence->ColumnCount = 2;

this->dataGridViewSequence->Columns[0]->Name = "Правильные";

this->dataGridViewSequence->Columns[1]->Name = "Неправильные";

this->dataGridViewTrans->ColumnCount = 1;

this->dataGridViewTrans->Columns[0]->Name = "Переходы";

}

bool error = false;

System::Void commandBox\_Enter(System::Object^ sender, System::Windows::Forms::KeyPressEventArgs^ e) {

/\*

\* Функция, вызывающаяся при нажатии клавиши Enter/Return на поле ввода.

\*/

if (e->KeyChar == (char)Keys::Return) {

ClearTable(this->dataGridViewTrans);

this->dataGridViewTrans->RowCount = 1;

if (this->commandBox->Text->Split(' ')->Length > 1) {

manual = false;

for each (String ^ lexem in this->commandBox->Text->Split(' ')) {

ProcessLexem(lexem);

s = 0;

}

}

else {

s = 0; t = 0; manual = true;

ClearTable(this->dataGridViewProtocol);

this->dataGridViewProtocol->ColumnCount = 1;

ProcessLexem(this->commandBox->Text);

if (s == 3) {

this->combTextBox->Text = "Комбинация верна!";

this->combTextBox->ForeColor = Color::Green;

}

else {

this->combTextBox->Text = "Комбинация не верна!";

this->combTextBox->ForeColor = Color::Red;

}

}

this->commandBox->Text = "";

}

}

System::Void buttonReset\_Click(System::Object^ sender, System::EventArgs^ e) {

// Нажатие кнопки Reset

Reset();

}

};

}

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы было выполнено построение регулярного выражения и конечного автомата по нему, при этом регулярное выражение построено лишь для частичного решения задачи, ибо построение полноценного регулярного выражения для данной задачи невозможно из-за отсутствия возможности построения конечного регулярного множества.

Для описанного автомата спроектирована и написана программа, имитирующая его работу. На вход программа принимает строку из некоторого алфавита и определяет соответствует ли она регулярному выражению или нет. При этом в программе строится таблица протокола работы.

Для автомата построен граф, для программы построена блок-схема и структурная схема. Программа протестирована на некотором множестве слов для отображения правильной её работы.