ISSN: 0975-766X
CODEN: IJPTFI
Available Online through ***Research Article***
www.ijptonline.com

REDUCING THE NUMBER OF STATES IN PUSHDOWN RECOGNIZERS BY MEANS OF EQUIVALENCE RELATION

Vladimir Mikhailovich Polyakov, Yuri Dmitrievich Ryazanov
Belgorod State Technological University named after V.G. Shukhov,
Russia, 308012, Belgorod, Kostyukov str., 46
Belgorod State Technological University named after V.G. Shukhov,
Russia, 308012, Belgorod, Kostyukov str., 46.

Received on: 15.10.2016 *Accepted on: 12.11.2016*

Abstract.

Pushdown recognizers are used for solving the tasks of context-free languages recognition. The paper considers a class of pushdown recognizers and a finite set of states, and deals with the solution of the task on reduction in a number of recognizer states. To solve this task, the relation over a set of recognizer states having a property of equivalence is introduced, such that "merging" of the equivalence class into one state gives a recognizer which is equivalent to the initial. The algorithm for splitting a set of states to equivalence classes and algorithm for building-up of a recognizer equivalent to the initial, but with smaller (not bigger) number of states are proposed. The method based on a consecutive splitting of a set of states so that nonequivalent states have fallen in various classes is used for splitting of a set of states to equivalence classes. The example for reduction of a number of recognizer states with the use of the proposed algorithm is represented. The proposed algorithm can be used by development of formal language processing programs.

Keywords: Context-free language, pushdown recognizer, state, transition, equivalent conversions.

Introduction. The important place in the theory of formal languages is occupied with context-free languages, the problem of recognition for which is solved with the help of pushdown recognizers (PD-recognizers) [1 – 6]. A class of PD-recognizers with a finite set of states is considered in the work. Two various PD-recognizers are named as equivalent if languages recognized by them are equal. The equivalent PD-recognizers may have a different number of states. In practice, the preference is given to the recognizers with a smaller number of states. The task of conversion of a given PD-recognizer into a recognizer equivalent to it with smaller (not bigger) number of states is solved in the work. To solve this task, the relation over set of states of a PD-recognizer having a property of equivalence is introduced, and it is shown that a conversion of a PD-recognizer comprising in "merging" of a pair of states belonging to the introduced relation into one state, does not change the recognized language, i.e. is the equivalent conversion. It follows that a "merging" of an equivalence class into one state is also an equivalent conversion. The algorithm for splitting a set of PD-recognizer states into equivalence classes and algorithm for build-up of a PD-recognizer equivalent to the initial, with not a great many of states than the initial PD-recognizer is proposed.

Definition of a pushdown recognizer with a finite set of states. We will consider in our work the PD-recognizers class [10 – 12] which can be built under Wirth diagrams [10] or syntactic diagrams with multiinput components [13]. A PD-recognizer from this class could be formally presented as follows:

$M = (Q, \mathrm{X}, \Gamma, I, S, P, W,$ *[delta]*, *[lambda]*, q_0, $\boldsymbol{q}$, *[gamma]* $_0$), where

Q – a finite set of states, $Q = \{q_0, q_1, \ldots, q_m, \boldsymbol{q}\}$;

X – a finite set of input symbols including an end mark ⊣ with which the input string is ended;

$\Gamma = Q \cup \{\nabla\}$ – a finite set of stack symbols (it is equal to the set of states added with a magazine base marker ∇);

I – a finite set of operations with a head, I = (*shift*, *hold*). Operation *shift* moves the head one position to the right, and *hold* does not change a head position;

S – a finite set of operations with a state, S = {*state* (q_0), *state* (q_1), …, *state* (q_m)}. Operation *state* (q_i) marks out transition into a state q_i;

P – set of operations with a magazine, P = {*push in* (q_0), *push in* (q_1), …, *push in* (q_m), *push out, not to change*}.

W – a finite set of output values, W = {*accept, reject*};

q_0 – an initial state, $q_0 \in Q$;

$\boldsymbol{q}$ – an accepting state, $\boldsymbol{q} \in Q$;

[gamma]$_0$ – initial contents of magazine, *[gamma]* $_0 = \nabla\boldsymbol{q}$ (the magazine contains a base marker and accepting state);

[delta]:$Q \times \mathrm{X} \times \Gamma \rightarrow I \times S \times P$ – a partial function of transitions which puts in correspondence to a state, to an input string symbol (being under a head) and to the upper symbol of magazine an operation with a head, a state and magazine, and the set of types of values for the triple (q_m, x, q_s) is equal to {(*shift*, *state* (q_n), *not to change*), (*hold*, *state* (q_n), *push in* (q_r)), (*hold*, *state* (q_s), *push out*)}. If the function value for the triple (q_m, x, q_s) is equal to (*shift*, *state* (q_n), *not to change*), we will designate such transition as (q_m, (x,→), q_n), and if the function value for this triple is equal to (*hold*, *state* (q_n), *push in* (q_r)), we will designate such transition as (q_m, (x, ↓(q_r)), q_n); if value for this triple is equal to (*hold*, *state* (q_s), *push out*), we will designate such transition as (q_m, (q_s,↑), q_s).

[lambda]:Q×X× $\Gamma \rightarrow W$ is a partial function of outputs which puts in correspondence to a state, an input string symbol (being under a head), and to the upper symbol of magazine the output value *accept* or *reject*. The function value for the triple (***q***, ⊣,∇) is equal to *accept*, and for all remaining where the function is determined –*reject*.

The intervals on which the functions *[delta]* and *[lambda]* are defined, have not intersected, and their combination is equal to sending area. The triple (q_m, α, ∇[gamma]) where q_m – a state, α – a part of an input string, starting with a symbol under the head and ending the end mark, [gamma] – magazine contents, is named as PD-recognizers configuration. An initial configuration is (q_0, α_0, ∇***q***), where α_0 – the entire input string (the head is over the first symbol). Let the PD-recognizer configuration is the triple (q_m, x α, ∇[gamma] q_s) where x – a symbol under the head, q_s – the upper symbol of magazine. If for the triple (q_m, x, q_s) the transitions function *[delta]* is defined, its value defines operations with the head, a state, and the magazine. In the course of execution of these operations the configuration changes. If for the triple (q_m x, q_s) the outputs function [lambda] is determined, the recognition process is ended with the result equal to the function [lambda] value. We will name such configuration as final. So, the PD-recognizer operation consists in changing of configurations. The final configuration is last where the result of recognition is determined. It is possible to present a PD-recognizer in the form of the weighed directed multigraph [6 – 9] which nodes match to states, and arcs to transitions. We will represent an initial state with an unmarked arrow, and a accepting state with a bold circle. Transition (q_m, (x,→), q_n) there matches to an arc from a node q_m to a node q_n, marked with (x,→), transition (q_m, (x, ↓(q_r)), q_n) to an arc from a node q_m to a node q_n, marked with (x, ↓(q_r)), and transition (q_m, (q_s,↑), q_s) to an arc from a node q_m to a node q_s marked with (q_s,↑). For the purpose to ensure compactness of representation of a PD-recognizer, we will represent "parallel arcs" from a state q_m in a state q_s on which various input symbols and equal operations are written, with one arc on which the set of input symbols and an operation was written. The example of the graph of a PD-recognizer is shown in fig. 1.

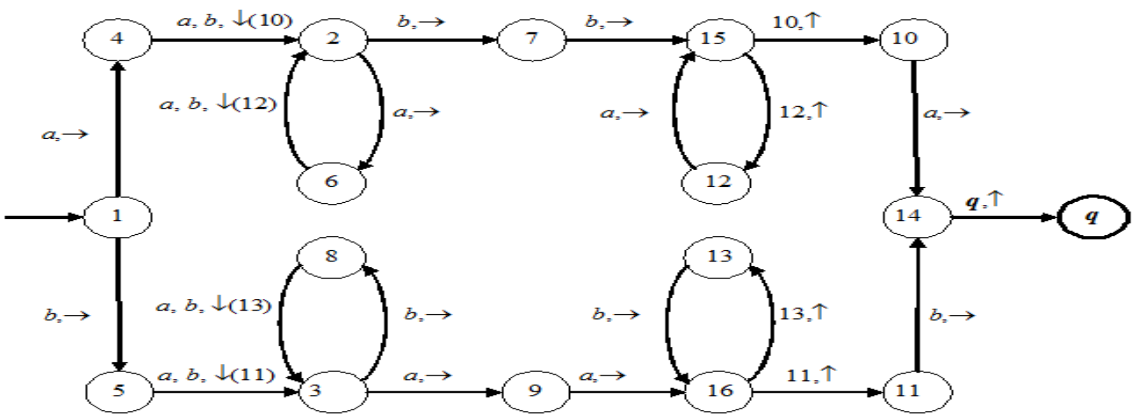

Fig. 1. The graph of a PD-recognizer.

Relation *E* on a set of the states having a property of equivalence. The pair of states q_m and q_m' belongs to a relation E if:

1) for some x, transition (q_m, (x,→), q_n) exists when and only when there is a transition (q_m', (x,→), q_n'), and (q_n, q_n') $\in E$;

2) for some x, transition (q_m, (x, ↓(q_s)), q_n) exists when and only when there is a transition (q_m', (x, ↓(q_s'), q_n') and (q_n, q_n') $\in E$ and (q_s, q_s') $\in E$;

3) transition (q_m, (q_n,↑), q_n) exists when and only when there is a transition (q_m', (q_s,↑), q_s);

This relation is reflexive, symmetrical and transitive.

The pair (q_m, q_m') $\in E$ will be named as a pair of equivalent states or there will be said that states q_m and q_m' are equivalent.

It follows from the third state equivalence requirement that if there exist only transitions from states q_m and q_m' with execution of the operation *push out*, the states q_m and q_m' are equivalent. In remaining cases, equivalence of other pairs of states is necessary for equivalence of states q_m and q_m'.

The pair of states containing accepting states ***q*** does not belong to the relation E.

The equivalent conversion of a pushdown recognizer reducing the number of states. We will consider a PD-recognizer M_1 in which there is a pair of the equivalent states (q_m, q_m').

Let's build a PD-recognizer M_2 as follows:

1) replace each transition of a type (q_t, d, q_m) in the PD-recognizer M_1 to the transition (q_t, d, q_m')

2) if for the transition from the state q_m' operation *push out* is executed, the PD-recognizer M_2 transits to a state which is pushed out from the magazine (the upper symbol of the magazine).

It is conventionally possible to name such conversion of the PD-recognizer M_1 as "merging" of states q_m and q_m' into one state.

Let PD-recognizers M_1 and M_2 process a string α, and on some step M_1 has occurred into the state q_m, and M_2 into the state q_m'. By this moment the PD-recognizers M_1 and M_2 have processed an equal number of symbols of the string α and contents of magazines of these recognizers is equal. It follows from the definition of equivalence of states q_m and q_m' that:

1) if M_1 will execute *shift* and will transit into the state q_n, then M_2 will execute *shift* and will transit into the state q_n', equivalent to the state q_n;

2) if M_1 will execute *push in* (q_s) and will transit into the state q_n, then M_2 will execute *push in* (q_s'), and will transit into the state q_n', where q_s is equivalent to q_s', and q_n is equivalent to q_n'; resulting in that at the top of magazines there will be the equivalent states, and recognizers will be in the equivalent states;

3) if M_1 will execute *push out* and will transit into the state q_n, then M_2 will execute *push out* and will transit into the state q_n' equivalent to the state q_n; and after completion of the pushing out operation, equivalent (or equal) states will be at the top of magazines;

4) if M_1 will reject the string α, then M_2 will also reject this string.

Thus, on each step of operation of PD-recognizers M_1 and M_2:

1) the part of the string α processed by the recognizer M_1 is equal to the part of a string α processed by the recognizer M_2;

2) recognizers M_1 and M_2 are in the equivalent states;

3) the number of symbols in the magazine of the recognizer M_1 is equal to the number of symbols in the magazine of the recognizer M_2, and *i*-th symbol in the magazine of the recognizer M_1 is equivalent (or is equal) to *i*-th symbol in the magazine of the recognizer M_2.

Therefore, if the string α is accepted (rejected) by one of the recognizers, then it is accepted (rejected) by another recognizer, i.e. recognizers M_1 and M_2 recognize the same language.

If there is a class of equivalent states in a PD-recognizer where a number of states is more than two, then a consecutive "merging" of pairs of the equivalent states will lead to "merging" of a class of the equivalent states into one state. Replacement of each class of equivalent states by one state allows to reduce a number of states of the PD-recognizer.

Algorithm for splitting of a set of states to equivalence classes under relation *E*. Relation *E* has a property of equivalence and determines splitting of a set of PD-recognizer states to equivalence classes. The equivalence class of an accepting state ***q*** contains only this state, therefore this class and the state ***q*** would not considered further. The following method is applicable for a splitting of a set of states to equivalence classes.

Taking into account requirements for a membership of pair of states in the relation *E,* we will sequentially split a set of states into subsets so that nonequivalent states have fallen into various subsets.

Let's consider in more details the requirements for a membership of pair of states (q_m, q_m',) in the relation *E*.

The first requirement.

The first requirement consists of two parts:

1 part: for some x the transition (q_m, (x,→), q_n) exists when and only when there is a transition (q_m', (x,→), q_n');

2 part: (q_n, q_n') $\in E$.

This requirement is true, if both parts are true.

The second requirement.

The second requirement consists of two parts:

1 part: for some x the transition (q_m, (x, ↓(q_s)), q_n) exists when and only when there is a transition (q_m', (x, ↓(q_s'), q_n');

2 part: (q_n, q_n') $\in E$ and (q_s, q_s') $\in E$.

This requirement is true, if both parts are true.

The third requirement.

This requirement is true, if in each of two states q_m and q_m' there is at least one transition with operation *push out*, or in one of two states q_m and q_m' there is no such transition.

At first we will split the states into subsets so that for any two states of a subset the third requirement were true, and only the first parts of the first two requirements. For a PD-recognizer (see fig. 1) there will be the following subsets:

Q_1 = {1,2,3} – there is two arcs from each state of this subset, one with the mark (a,→) and the second with the mark (b,→);

Q_2 = {4,5,6,8} – there is one arc with the mark (a, b, ↓(q_s)) from each state of this subset. It is not important what was exactly pushed into magazine for the truth of the first part of the second requirement;

Q_3 = {9,10,12} – there is one arc with the mark (*and*,→) from each state of this subset;

Q_4 = {7,11,13} - there is one arc with the mark (b,→) from each state of this subset;

Each arc which comes out any state of this subset has an operation *push out*, therefore states of this subset are pair-wise equivalent and it forms an equivalence class;

Q_5 = {14,15,16} - Each arc which comes out any state of this subset has an operation *push out*, therefore states of this subset are pair-wise equivalent and it forms an equivalence class;

Which form a splitting R = {Q_1, Q_2, Q_3, Q_4, Q_5}.

The states belonging to various subsets, are explicitly nonequivalent, but two states from one subset may be nonequivalent, as when splitting we did not consider the second parts of the first and second requirement for a membership of pair of states in the relation E.

To take into account the second parts of requirements, we will build the table T in which columns match to PD-recognizer states, and strings to transitions of type (q_m, (x,→), q_n) (we will name the string matching to such transition as (x,→)), and to transitions of type (q_m, (x, ↓(q_s)), q_n) (we will name the string matching to such transition as (x,↓)). The strings matching to "parallel" transitions will be merged into one string.

We will designate the table cell T at the string l and the column c as T [l, c].

The transition (q_m, (x,→), q_n) there matches to the cell T [(x,→), q_m]. If $q_n \in Q_i$, we will write i in the cell T [(x,→), q_m]. If states q_m and q_m' belong to one subset and there are transitions (q_m, (x,→), q_n) and (q_m', (x,→), q_n') and T [(x,→), q_m] ≠T [(x,→), q_m'] it means that (q_n, q_n',) $\notin E$, therefore, the second part of the first requirement is false, (q_m, q_m',) $\notin E$ and it is necessary to include states q_m and q_m' in different subsets of new splitting R'.

The transition (q_m, (x, ↓(q_s)), q_n) there matches to the cell T [(x,↓), q_m]. If $q_s \in Q_i$ and $q_n \in Q_j$ we will write (i, j) in the cell T [(x,↓), q_m]. If states q_m and q_m' belong to one subset and there are transitions (q_m, (x, ↓(q_s)), q_n) and (q_m', (x, ↓(q_s')), q_n') and T [(x,↓), q_m] ≠T [(x,↓), q_m'] it means that (q_n, q_n',) $\notin E$ or (q_s, q_s',) $\notin E$, therefore, the second part of the second requirement is false, (q_m, q_m',) $\notin E$ and it is necessary to include states q_m and q_m' in different subsets of new splitting R'.

The new splitting R' is formed after analysis of the table T. Subset Q_i' of the splitting R' is formed from units of some subset Q_j of the splitting R. Q_i' is a maximum potency subset of the set Q_j such that for each pair of states q_m and q_m' belonging to subset Q_i' for all strings l of the table T, the expression T [l, q_m] = T [l, q_m'] is true.

If $R' \neq R$, it is accepted that R = R', then we build the table T anew by the rules presented above and form a new splitting.

If R' = R, R' represents a set of equivalence classes.

First table T for a PD-recognizer (see fig. 1) is presented in table 1.

Table 1:

	Q_1			Q_2				Q_3			Q_4			Q_5		
	1	2	3	4	5	6	8	9	10	12	7	11	13	14	15	16

a,→	2	2	3					5	5	5						
b,→	2	4	2								5	5	5			
a b,↓				1 3	1 4	1 3	1 4									

The table shows that the subset {1,2,3} is split into one-element subsets {1}, {2} and {3}, the subset {4,5,6,8} to {4,6} and {5,8}, and remaining subsets are not split and as a result we gain the splitting {{1}, {2}, {3}, {4,6}, {5,8}, {9,10,12}, {7,11,13}, {14,15,16}}. We build new table *T* according to this splitting (see tab. 2).

Table 2:

	Q_1	Q_2	Q_3	Q_4		Q_5		Q_6			Q_7			Q_8		
	1	2	3	4	6	5	8	9	10	12	7	11	13	14	15	16
a,→	4	4	6					8	8	8						
b,→	5	7	5								8	8	8			
a b,↓				2 6	2 6	3 7	3 7									

This table does not include different columns matching to states belonging to any single subset. Therefore the gained splitting is a splitting to equivalence classes.

Algorithm for reduction of a number of states for a pushdown recognizer. Using the relation E on a set of states of a PD-recognizer M_1, it is possible to build a PD-recognizer M_2 equivalent to the recognizer M_1 which will not have more states than the recognizer M_1. Reduction of a number of states is attained at the expense of "merging" of all states belonging to one equivalence class, into one state.

We build the PD-recognizer M_2 as follows.

Let's take one state at a time from each equivalence class and we will gain a set of states of the PD-recognizer M_2.

Now it is necessary to determine transitions for each state of the recognizer M_2.

Let's suppose that q_m is the state of the PD-recognizer M_2.

If the PD-recognizer M_1 has a transition (q_m, (x,→), q_n), the state q_n belongs to equivalence class Q_i, and the state q_n' is chosen from the class Q_i in the capacity of the state of the PD-recognizer M_2, it would be necessary to add a transition (q_m, (x,→), q_n') into the PD-recognizer M_2. If the PD-recognizer M_1 has a transition (q_m, (x, ↓(q_s)), q_n), $q_n \in Q_i$, $q_s \in Q_j$ and the state q_n' is chosen from Q_i, and the state q_n' from Q_i, it is necessary to add a transition (q_m, (x,

$\downarrow(q_s'))$, q_n') into the PD-recognizer M_2. If the PD-recognizer M_1 has a transition (q_m, (q_n,↑), q_n), and $q_n \in Q_i$, and the state q_n' is chosen from Q_i, it is necessary to add a transition (q_m, (q_n',↑), q_n') into the PD-recognizer M_2. If $q_m \in Q_j$ and there is still one state in Q_j, for example q_s, and there is a transition (q_s, (q_p,↑), q_p), $q_p \in Q_k$ in the PD-recognizer M_1, and the state q_p' is chosen from Q_k, it is necessary to add a transition (q_m, (q_p',↑), q_p') into the PD-recognizer M_2.

The splitting on equivalence classes {{1}, {2}, {3}, {4,6}, {5,8}, {9,10,12}, {7,11,13}, {14,15,16}} is built on the set of states of a PD-recognizer (see fig. 1). Taking into account an accepting state ***q***, we will add into this splitting the class {***q***}. Let's select from each class the state with the smaller number and let's determine transitions of a new PD-recognizer by the rules presented above. As a result we will gain a PD-recognizer which graph is presented on fig. 2. In this recognizer there are twice less states than in an initial recognizer (see fig. 1).

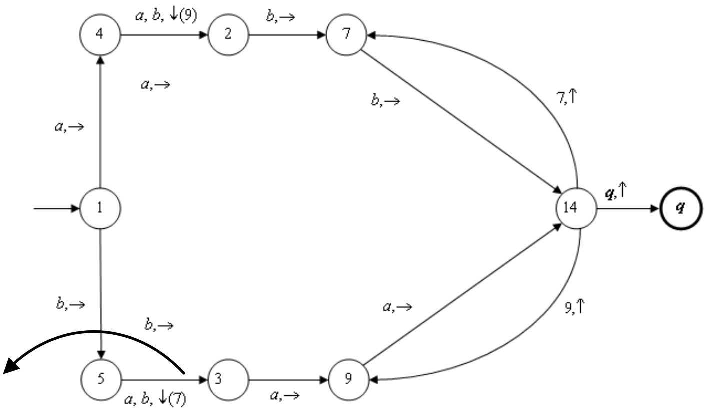

Fig. 2. The graph of a PD-recognizer

Resume. The paper solves the task of conversion of an initial PD-recognizer into a recognizer equivalent to it with smaller (not bigger) number of states on the basis of an introduced equivalence relation. The number of states of a PD-recognizer which is gained as a result of conversion, should be determined by an number of equivalence classes.

Outputs. For further reduction of a number of states, there can be used any other equivalence relation on a set of states of a recognizer that splits the set of states in a smaller number of equivalence classes such that a "merging" of the equivalence class in one state would leads to obtaining a recognizer equivalent to the initial.

Acknowledgments. The study has been executed with the support of the Russian Foundation of Fundamental Research (the grant # 16-07-00487).

References

1. Schutzenberger, M.P., 1963. On context-free languages and pushdown automata. Information and Control 6:3: 246–264.

2. Aho, A. V. and J.D. Ullman, 1972. The Theory of Parsing, Translation and Compiling: v. 1. PrenticeHall, pp: 576.
3. Lewis, P.M., D.J. Rosenkrantz and R. E. Stearns, 1976. Compiler Design Theory. Addison-Wesley Publishing Company, pp: 647/
4. Martynenko, B.K., 2004. Syntax-controlled data processing. 2nd supplemented edition. St. Petersburg: Publishing House of the St. Petersburg University, p. 317.
5. Aho, A. V., M.S. Lam, R. Sethi and J.D. Ullman, 2013. Compilers: Principles, Techniques, and Tools (Pearson New International Edition). Pearson Education Limited, pp: 952.
6. Hopcroft, J.E., R. Motwani and J.D. Ullman, 2013. Introduction to Automata Theory, Languages, and Computation (3rd ed.). Pearson.,pp: 496.
7. Belousov, A. I. and S. B. Tkachev, 2012. Presentation of push-down automation as oriented multigraphs. Science and Education of Bauman MSTU, 9: http://technomag.bmstu.ru/doc/460973.html
8. Stanevichene, L.I., 1993. Graphs of pushdown automata // The founding conference of the Russian Association "Women in Mathematics". Abstracts. M., p. 50
9. Vylitok, A.A., 1996. On the Construction of the Graph Pushdown Automaton. Moscow University Computational Mathematics and Cybernetics, 3: 68–73.
10. Ryazanov, Yu.D., 2014. Synthesis of Pushdown Recognizers from Deterministic Syntax Diagrams.Proceedings of Voronezh State University. Series: Systems analysis and information technologies, 1: 138–145.
11. Ryazanov, Yu.D. and I.N. Savelova, 2015. Transforming of the Pushdown Recognizer with One State into Recognizer with Finite Set of States. Modeling, Optimization and Information Technology, 4(11): https://moit.vivt.ru/wp-content/uploads/2015/12/RyazanovSavelova_4_15_1.pdf
12. Ryazanov, Yu.D., 2016. Transforming of the Pushdown Recognizer with Finite Set of States into Recognizer with One State. Bulletin of BSTU named after V.G. Shukhov, 1: 194–199.
13. Polyakov, V.M. and Yu.D. Ryazanov, 2015. VIRT Charts To Multiport Component Syntactic Charts Transformation. Global Journal of Pure and Applied Mathematics, 11(5): 3939–3952.