Solution to Homework 9

Background. In Problem 7, we considered a grammar with rules

$$S \to \varepsilon; \quad S \to ABS; \quad S \to ASB; \quad S \to SAB; \quad S \to SBA; \quad S \to BSA;$$

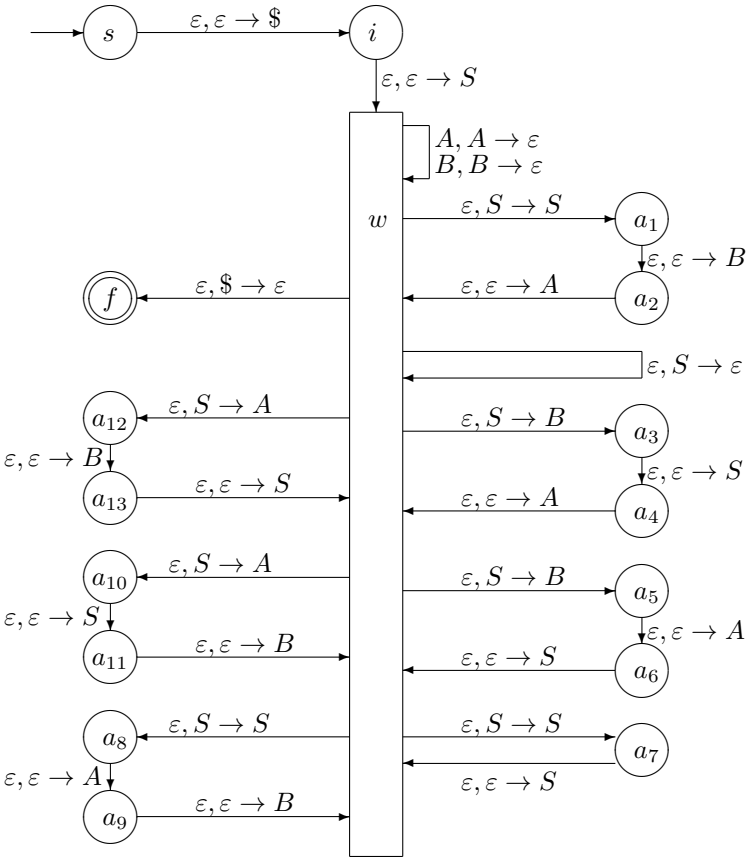
$$S \to SBA; \quad S \to SS,$$

where A and B are terminal symbols.

Tasks:

1. Use a general algorithm to construct a (non-deterministic) pushdown automaton that corresponds to context-free grammar described in Problem 7.
2. Show, step by step, how the word $ABAB$ will be accepted by this automaton.

Solution to Task 1. By using the general algorithm, we get the following pushdown automaton:



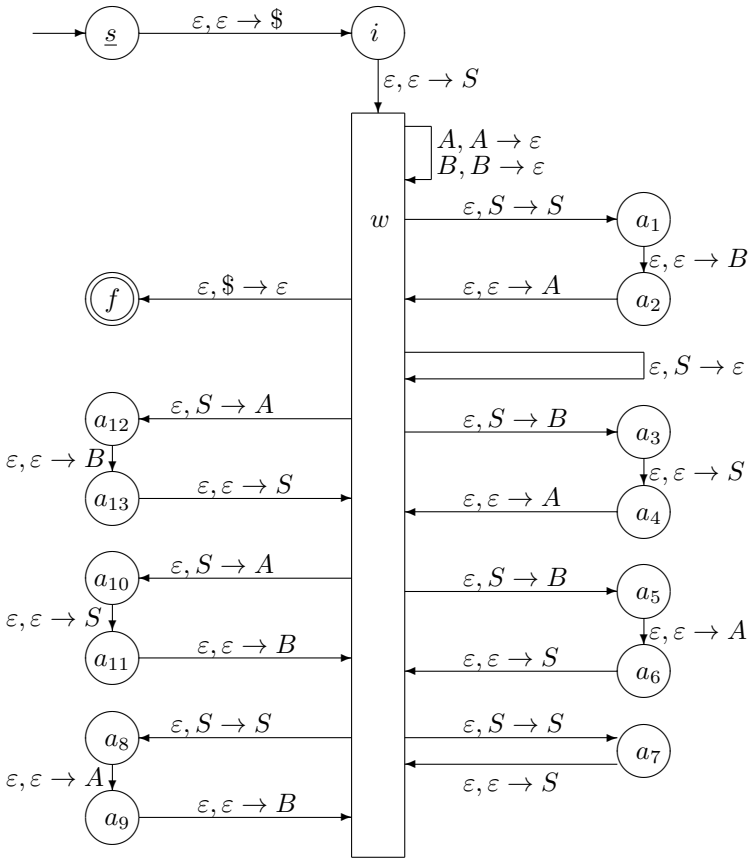
Solution to Task 2. Let us show how this is done on the example of the word $ABAB$ generated by the above automaton:

$$\underline{S} \to AB\underline{S} \to ABAB\underline{S} \to ABAB.$$

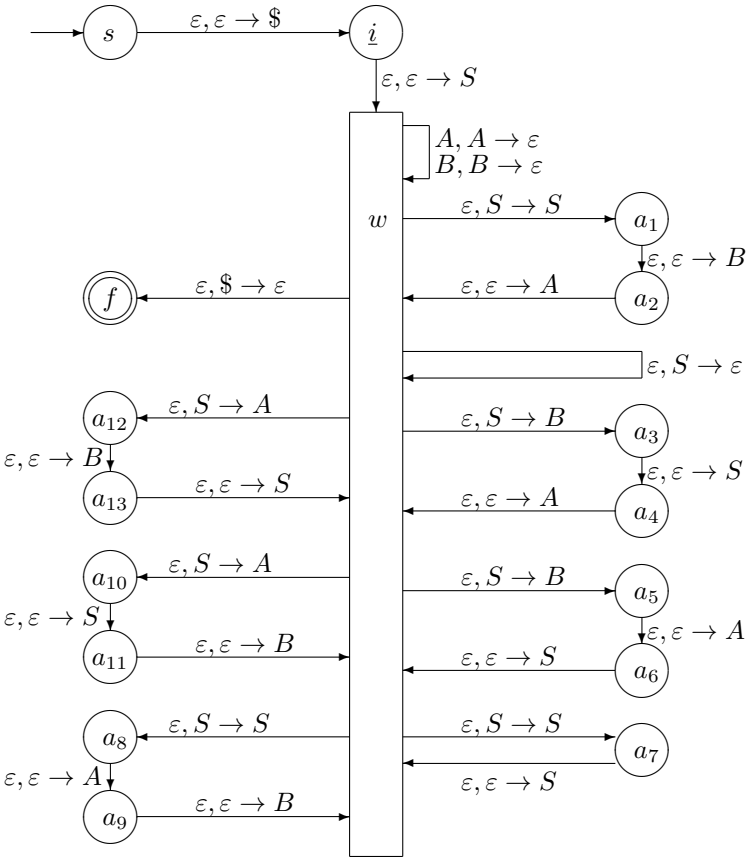
To make this derivation clearer, let us mark the variable S corresponding to different transitions by subscripts:

- we start with the first occurrence S_1 of the variable S;
- we then use the rule $S_1 \to ABS_2$ whose right-hand side contains the second occurrence S_2 of the variable S;
- this occurrence, in its turn, gets transformed into $S_2 \to ABS_3$ for yet another occurrence S_3 of the same variable S; so far, the derivation takes the form
$$S_1 \to ABS_2 \to ABABS_3;$$
- finally, to the occurrence S_3, we apply the rule $S_3 \to \varepsilon$, resulting in the desired derivation:
$$S_1 \to ABS_2 \to ABABS_3 \to ABAB.$$

Let us now trace what our pushdown automaton will do. We start in the state s with an empty stack:

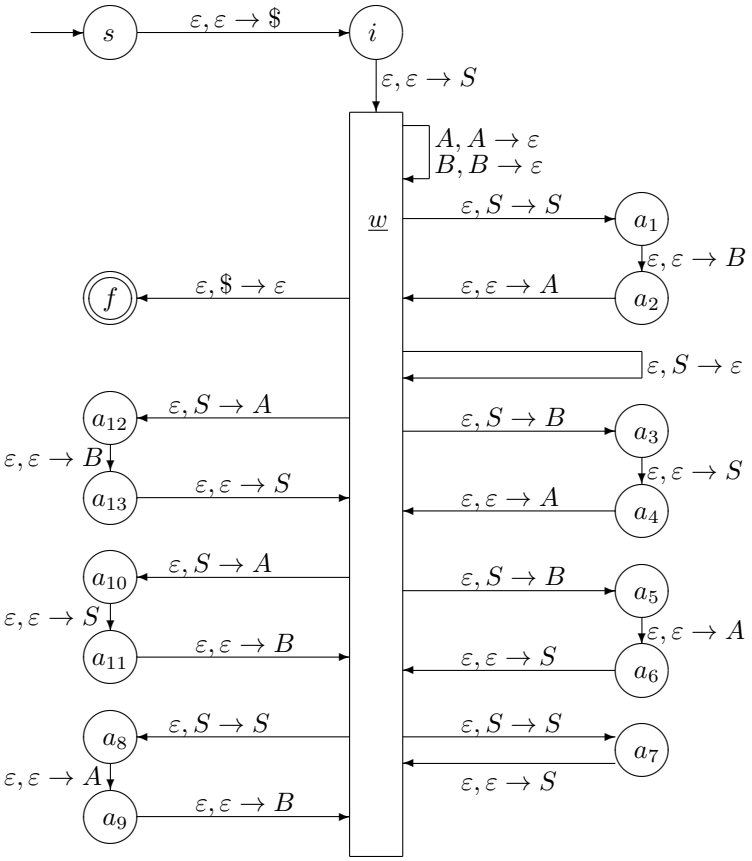


The only thing we can do when in the state s is push the dollar sign into the stack and get to the intermediate state i:



The contents of the stack is as follows:

When we are in the state i, the only thing we can do is push the starting variable S (which corresponds to the first occurrence S_1 of this variable) into the stack and go into the working state w;



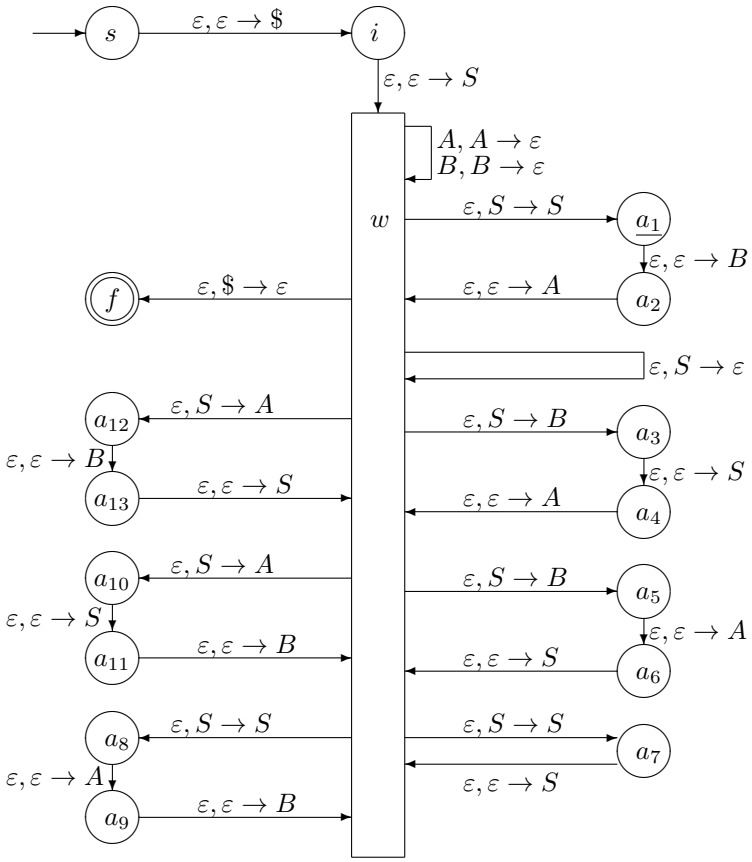
Now, the stack contains the starting variable on top of the dollar sign:

Now that we are in the working state, we can start following the rules that were used to derive the word $ABAB$. The first rule was $S \to ABS$, or, to be precise, $S_1 \to ABS_2$. As we have mentioned, this rule is implemented in three steps:

- first, we pop S (that corresponds to the first occurrence S_1) and push the last symbol of the right-hand side – in this cases, the symbol S (that corresponds to the second occurrence S_2) – into the stack, getting into the auxiliary state a_1;
- then, we push B into the stack, getting into the auxiliary state a_2;
- finally, we push A into the stack, and go back to the working state w.

Let us illustrate this step by step.

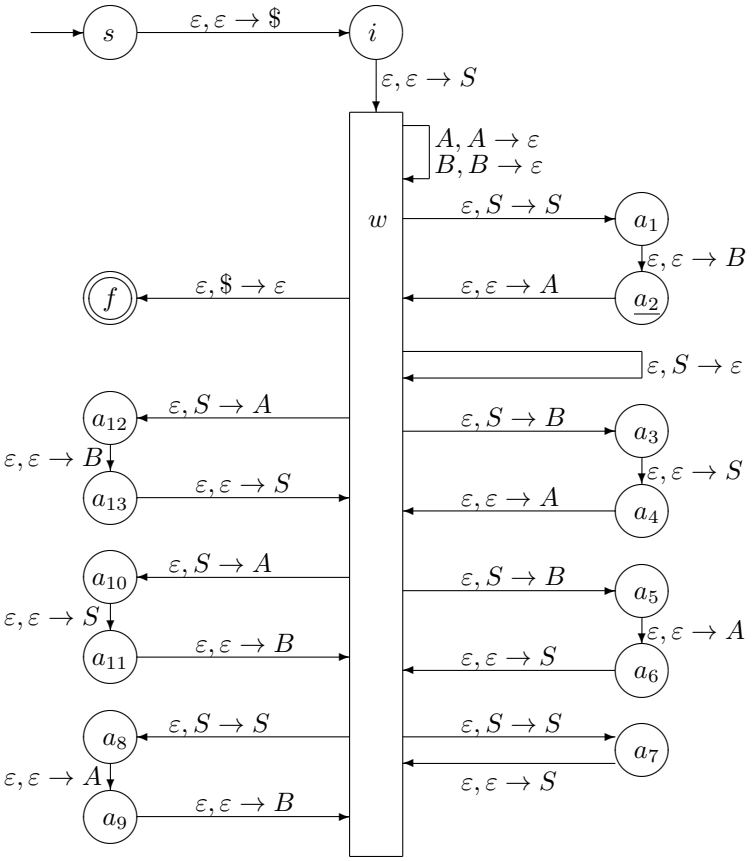
First, we pop S, push S, and go into the state a_1:



The stack will now have the new S instead of the original S:

S
\$

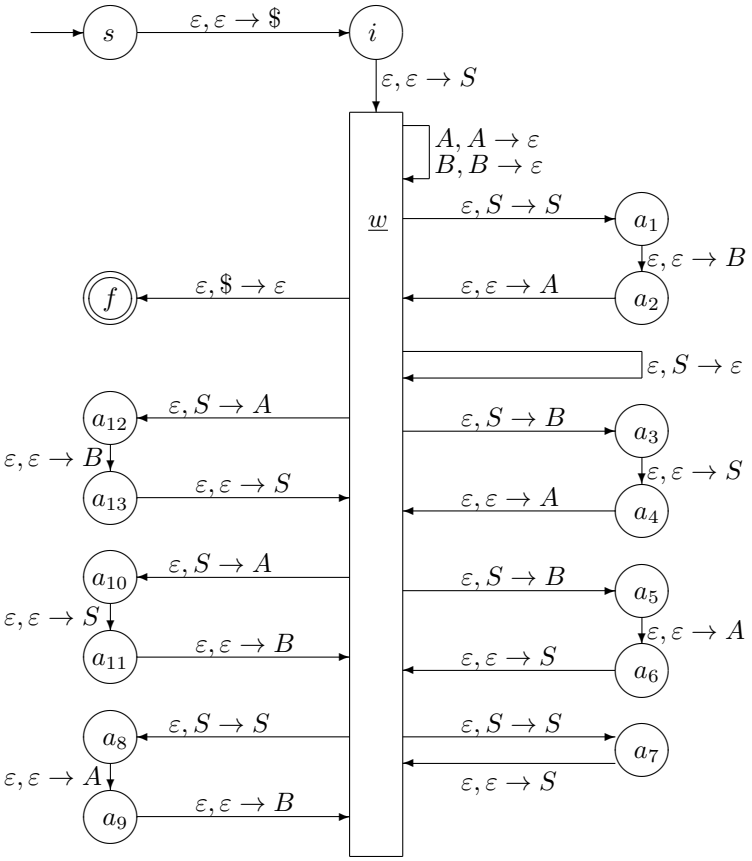
Then, we push B into the stack and go into the state a_2:



The stack will now have B on top of its previous contents:

B
S
\$

Finally, we push A into the stack, and go back to the working state:



The stack will now have symbol A at the top:

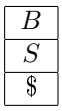
A
B
S
\$

Now, the symbol A is top of the stack. The only thing we can do if a terminal symbol is on top of the stack is use one of the rules of the type $x, x \to \varepsilon$ where x stands for the corresponding terminal symbol.

In our case:

- since the terminal symbol on top of the stack is the symbol A,
- we need to use the rule $A, A \to \varepsilon$,

i.e., we read the symbol A from the original word $ABAB$ and pop the top symbol A from the stack. We remain in the same state w, but the stack changes. The stack now has the following form:



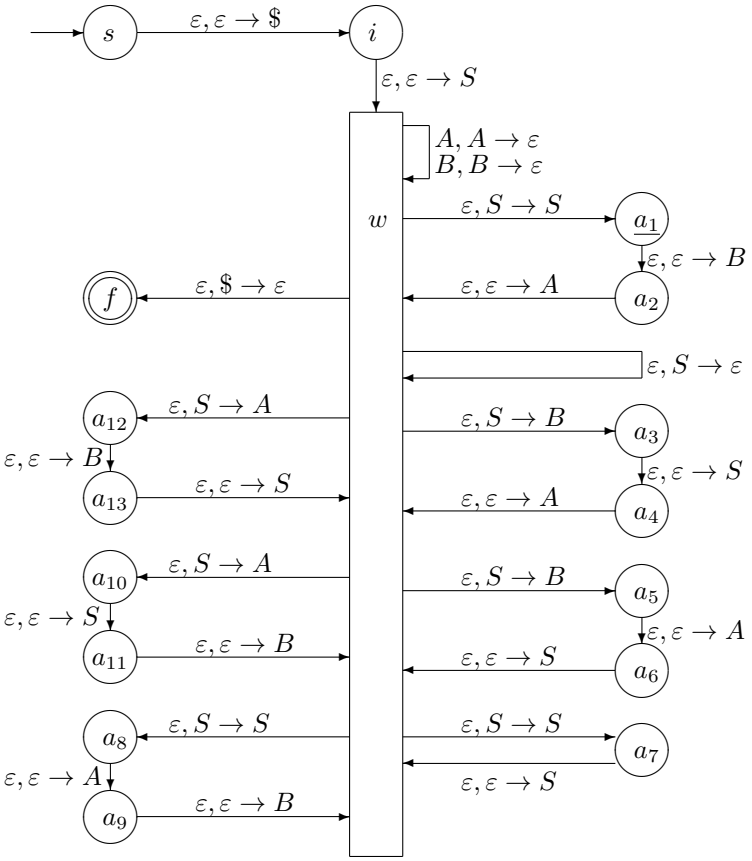
Now, we have a terminal symbol B on top of the stack. To delete it from the stack, we need to use the rule $B, B \to \varepsilon$, i.e., read the symbol B and pop B from the stack. The state w remains the same, but the stack changes to

Now, we have the symbol S_2 on top of the stack, so we use the rule $S_2 \to ABS_3$. As before, this rule is implemented in three steps:

- first, we pop S (that corresponds to the first occurrence S_1) and push the last symbol of the right-hand side – in this cases, the symbol S (that corresponds to the second occurrence S_2) – into the stack, getting into the auxiliary state a_1;
- then, we push B into the stack, getting into the auxiliary state a_2;
- finally, we push A into the stack, and go back to the working state w.

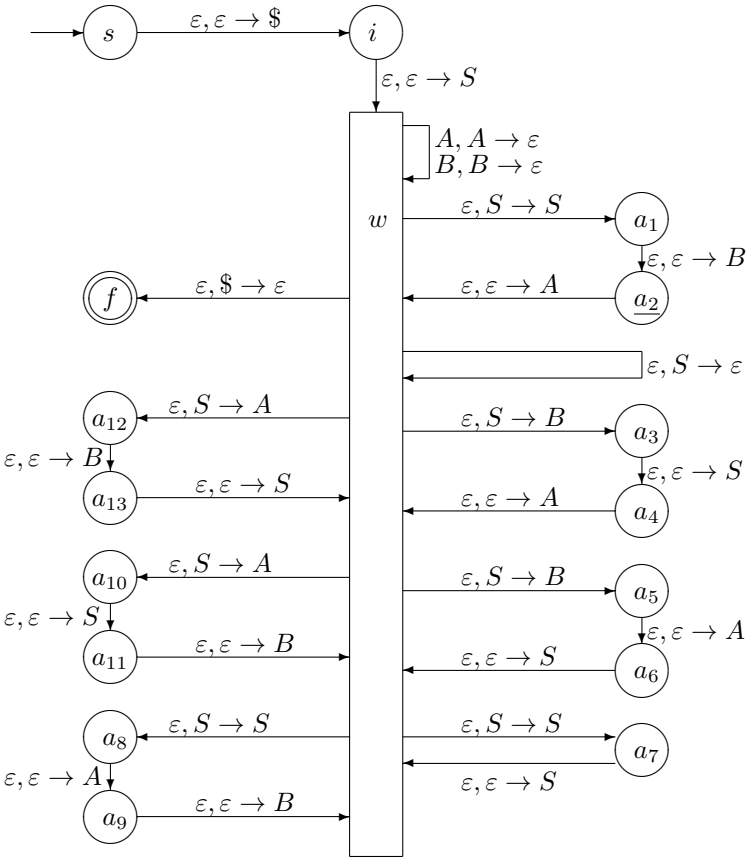
Let us illustrate this step by step.

First, we pop S, push S, and go into the state a_1:



The stack will now have the new S instead of the original S:

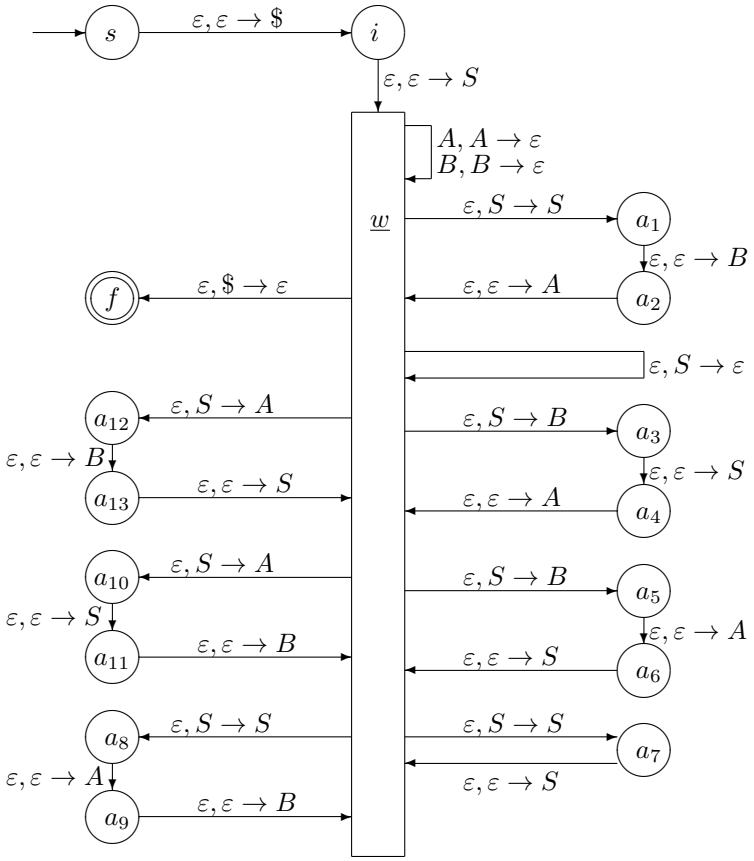
Then, we push B into the stack and go into the state a_2:



The stack will now have B on top of its previous contents:

B
S
\$

Finally, we push A into the stack, and go back to the working state:



The stack will now have symbol A at the top:

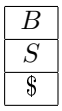
A
B
S
\$

Now, the symbol A is top of the stack. The only thing we can do if a terminal symbol is on top of the stack is use one of the rules of the type $x, x \to \varepsilon$ where x stands for the corresponding terminal symbol.

In our case:

- since the terminal symbol on top of the stack is the symbol A,
- we need to use the rule $A, A \to \varepsilon$,

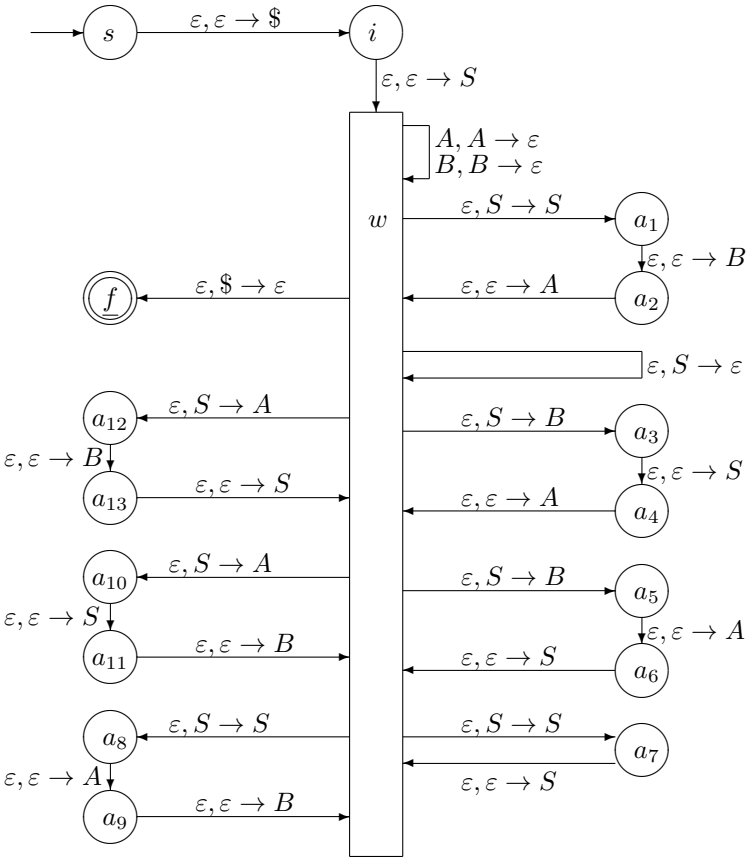
i.e., we read the symbol A from the original word $ABAB$ and pop the top symbol A from the stack. We remain in the same state w, but the stack changes. The stack now has the following form:



Now, we have a terminal symbol B on top of the stack. To delete it from the stack, we need to use the rule $B, B \to \varepsilon$, i.e., read the symbol B and pop B from the stack. The state w remains the same, but the stack changes to

To eliminate S from the top of the stack, we use the rule $S_3 \to \varepsilon$ that corresponds to the transition $\varepsilon, S \to \varepsilon$. So, we pop S_3 and return to the same state w. Now, we are in the same state, but the stack contains only the dollar sign:

Finally, we use the rule $\varepsilon, \$ \to \varepsilon$ to pop the dollar sign and go to the final state:



Now, we are in the final state f with the empty stack. This means that the word $++--$ is accepted by this pushdown automaton.

A graphical description of the transitions.

read							A	B
state	s	i	w	a_1	a_2	w	w	w
stack		\$	S \$	S \$	B S \$	A B S \$	B S \$	S \$

read				A	B		
state	a_1	a_2	w	w	w	w	f
stack	S \$	B S \$	A B S \$	B S \$	S \$	\$	