Solution to Homework 1

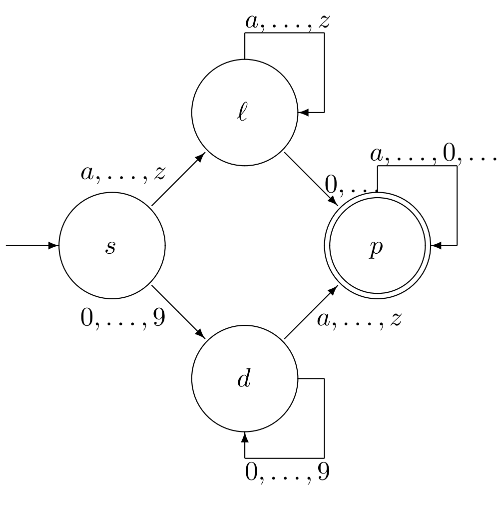
Task 1: general description. In class, we designed automata for recognizing integers and real numbers.

Task 1.1. Use the same ideas to describe an automaton for recognizing good passwords. A good password should contain at least one letter and at least one digit.

A natural idea is to have 4 states: start (s), has letters but not digits (ℓ), has digits but no letters (d), and has both (p). Start is the starting state, p is the only final state. The transitions are as follows:

- from s, any letter $a, \ldots, z$ leads to ℓ, any digit $0, \ldots, 9$ leads to d;
- from ℓ, any letter leads back to ℓ, every digit leads to p;
- from d, any digit leads back to d, every letter leads to p;
- from p, every symbol leads back to p.

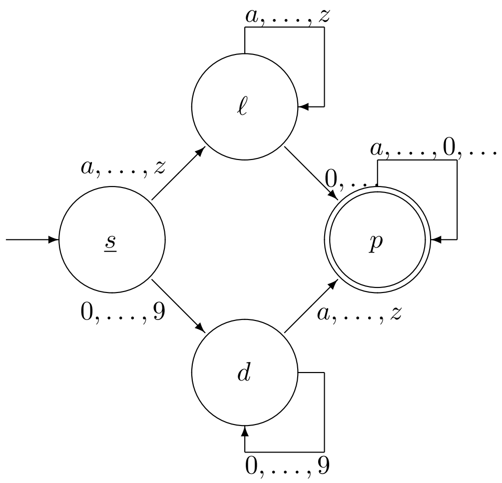
Solution. The desired automaton takes the following form:



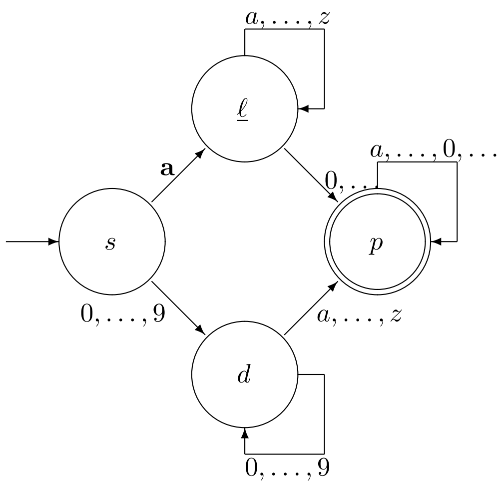
Task 1.2. Trace, step-by-step, how the finite automaton from Part 1.1 will check whether the following two words (sequences of symbols) are correct names for Java constants:

- the word $a12$ (which this automaton should accept) and
- the word 123 (which this automaton should reject).

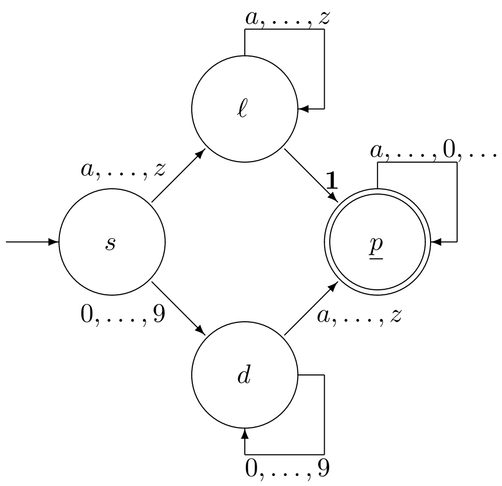
Solution. Let us trace how this automaton will accept the word $a12$ We are originally in the state s:



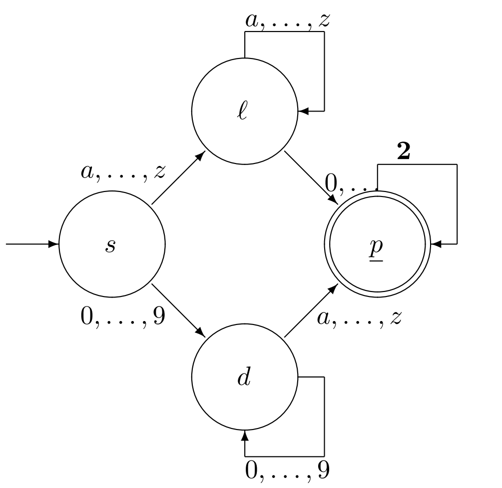
Then, we read the first letter a of the word **a**12, so we move to state ℓ:



Then, we read the second letter 1 of the word $a\mathbf{12}$, and we get to the state p:

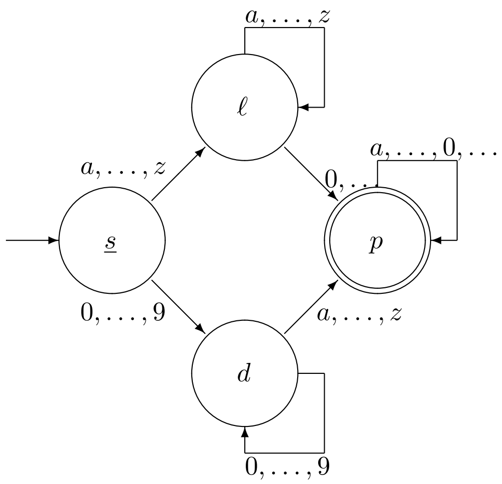


Then, we read the third symbol 1 of the word $a1\mathbf{2}$, and we stay in the state p:

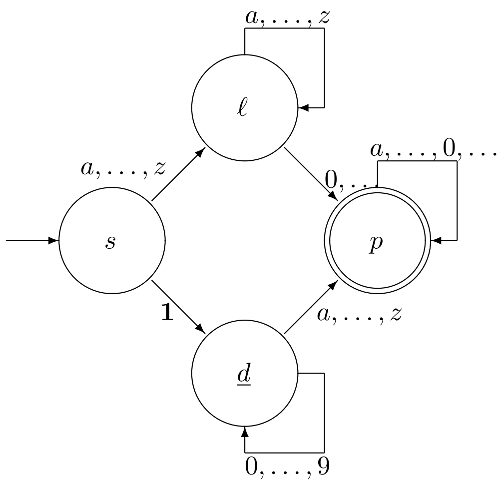


The word is read, we are in the final state, so the word $a12$ is accepted.

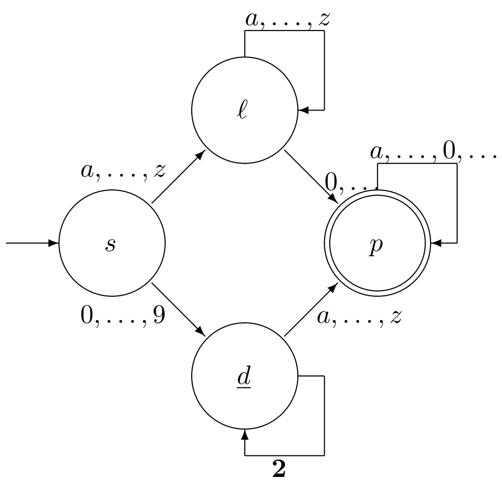
Let us now trace how the automaton will react to the word 123. We also start in the start state s:



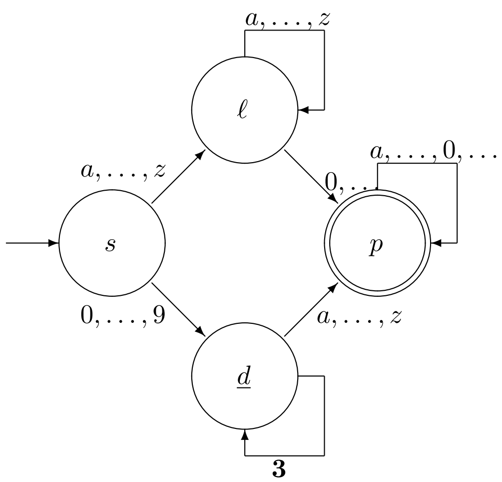
Then, we read the first letter 1 of the word **1**23, so we move to the state d;



After that, we read the second symbol 2 of the word 1**2**3 and we stay in the state d:



Then, we read the last symbol 3 of the word 12**3** and stay in the state d:



We have read all the symbols, we are in the state d which is not final, so the word 123 is not accepted.

Task 1.3. Write down the tuple $\langle Q, \Sigma, \delta, q_0, F\rangle$ corresponding to the automaton from Part 1.1:

- Q is the set of all the states,
- Σ is the alphabet, i.e., the set of all the symbols that this automaton can encounter; for simplicity, consider only three symbols: 0, 1, and the letter a;
- $\delta : Q \times \Sigma \to Q$ is the function that describes, for each state q and for each symbol s, the state $\delta(q, s)$ to which the automaton that was originally in the state q moves when it sees the symbol s (you do not need to describe all possible transitions this way, just describe two of them);
- q_0 is the staring state, and
- F is the set of all final states.

Solution. $Q = \{s, \ell.d, p\}$, $\Sigma = \{0, 1, a\}$, $q_0 = s$, $F = \{p\}$, and the transition function δ is described by the following table:

	0	1	a
s	d	d	ℓ
ℓ	p	p	ℓ
d	d	d	p
p	p	p	p

Task 1.4. Apply the general algorithm for union and intersection to:

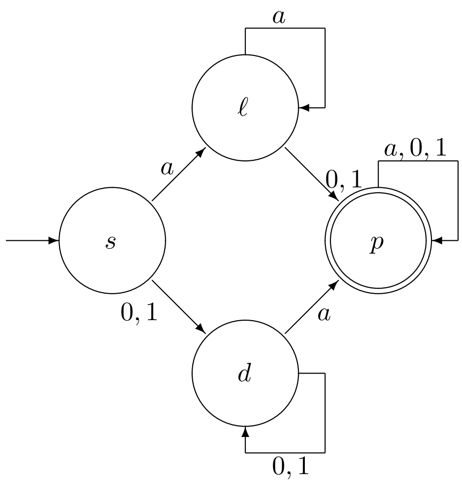
- the automaton from Part 1.1 as Automaton A and
- an automaton for recognizing Java names for variables as Automaton B.

In Java, a name of the variable should start with a letter, all other symbols can be letters or digits. A natural idea is to have 3 states: start (s), correct name (c), and error (e). Start is the starting state, c is the only final state. The transitions are as follows:

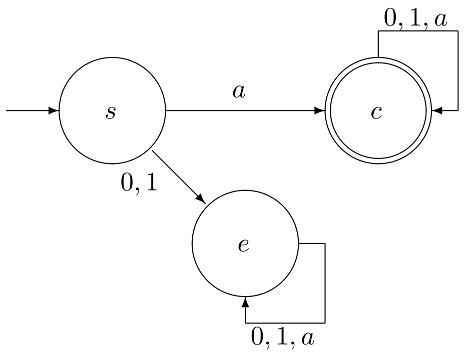
- from s, any letter lead to c, any digit leads to e;
- from c, any symbol leads back to c;
- from e, every symbol leads back to e.

For simplicity, in your automaton for recognizing the union and intersection of the two languages, feel free to assume that you only have symbols 0, 1, and a.

Solution. If we limit ourselves to these 3 symbols, then the Automaton A takes the following form:



The Automaton B has the following form:



In the beginning, before we see any symbols, both automata are in the state s, so the combined automaton is in the state (s, s). Then:

- if we read a, Automaton A goes into state ℓ and automaton B goes into state c, so we go into the state (ℓ, c);
- if we read 0 or 1, Automaton A goes into state d and automaton B goes into state e, so we go into the state (d, e).

We can similarly describe transitions from these three new states. As a result, we get the following automaton: