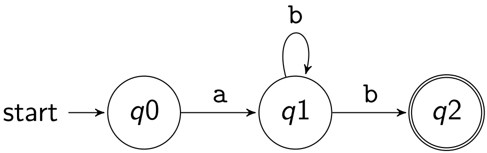
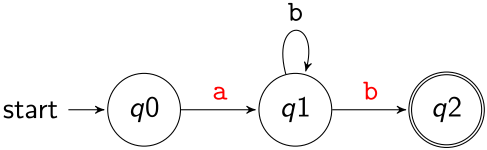
Finite automata



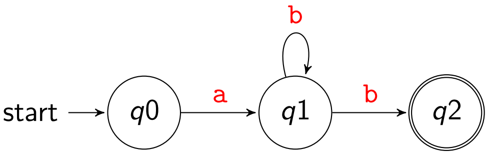
Finite automata



accepting runs $q0 \xrightarrow{\texttt{a}} q1 \xrightarrow{\texttt{b}} q2 \in F$

ab

Finite automata

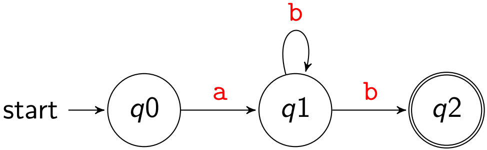


accepting runs $q0 \xrightarrow{\mathtt{a}} q1 \xrightarrow{\mathtt{b}} q2 \in F$

$q0 \xrightarrow{\mathtt{a}} q1 \xrightarrow{\mathtt{b}} q1 \xrightarrow{\mathtt{b}} q2 \in F$

ab, abb

Finite automata



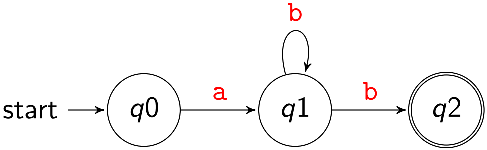
accepting runs $q0 \xrightarrow{\texttt{a}} q1 \xrightarrow{\texttt{b}} q2 \in F$

$q0 \xrightarrow{\texttt{a}} q1 \xrightarrow{\texttt{b}} q1 \xrightarrow{\texttt{b}} q2 \in F$

$\vdots$

regular expression $\texttt{ab}^*\texttt{b}$ denoting the regular language $\{\texttt{ab}, \texttt{abb}, \dots\}$

Finite automata and regular grammars: example



accepting runs $q0 \xrightarrow{\texttt{a}} q1 \xrightarrow{\texttt{b}} q2 \in F$

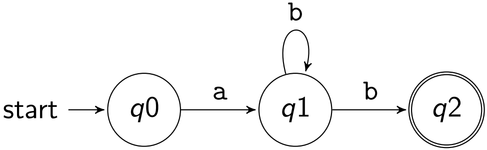
$$q0 \xrightarrow{\texttt{a}} q1 \xrightarrow{\texttt{b}} q1 \xrightarrow{\texttt{b}} q2 \in F$$

$$\vdots$$

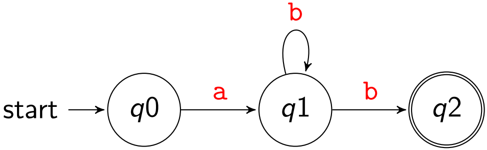
regular expression $\texttt{ab}^*\texttt{b}$ denoting the regular language $\{\texttt{ab}, \texttt{abb}, \ldots\}$

```
q0 --> [a], q1.        % tran(q0,a,q1).
q1 --> [b], q2.        % tran(q1,b,q2).
q1 --> [b], q1.        % tran(q1,b,q1).
q2 --> [].             % final(q2).
```

Strings accepted by a finite automaton

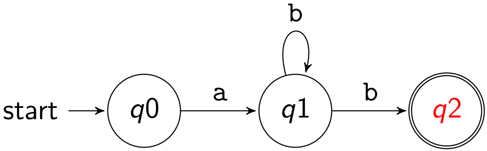


Strings accepted by a finite automaton



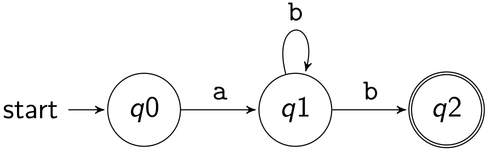
```
tran(q0,a,q1).
                tran(q1,b,q2).
        tran(q1,b,q1).
```

Strings accepted by a finite automaton



```
tran(q0,a,q1).
          tran(q1,b,q2).
     tran(q1,b,q1).
                final(q2).
```

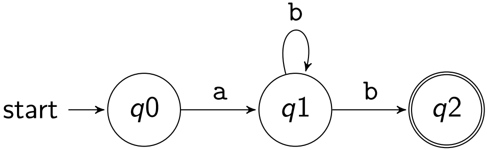
Strings accepted by a finite automaton



```
tran(q0,a,q1).
          tran(q1,b,q2).
     tran(q1,b,q1).
                final(q2).
```

```
accept(String) :- steps(q0,String,Q), final(Q).
```

Strings accepted by a finite automaton

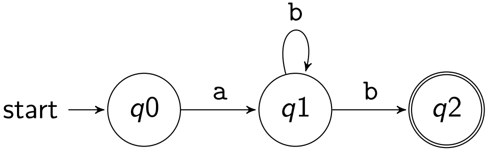


```
tran(q0,a,q1).
          tran(q1,b,q2).
     tran(q1,b,q1).
                 final(q2).
```

```
accept(String) :- steps(q0,String,Q), final(Q).
```

```
steps(Q,[H|T],N) :- tran(Q,H,Qn), steps(Qn,T,N).
```

Strings accepted by a finite automaton



```
tran(q0,a,q1).
                    tran(q1,b,q2).
          tran(q1,b,q1).
                                 final(q2).
```

```
accept(String) :- steps(q0,String,Q), final(Q).

steps(Q,[],Q).

steps(Q,[H|T],N) :- tran(Q,H,Qn), steps(Qn,T,N).
```

Regular expressions over a finite alphabet A

- $\emptyset$ denoting the empty set
- ϵ denoting the set $\{\epsilon\}$ containing only the empty string, which has no characters at all
- a in A denoting the set $\{a\}$

Regular expressions over a finite alphabet A

- $\emptyset$ denoting the empty set
- ϵ denoting the set $\{\epsilon\}$ containing only the empty string, which has no characters at all
- a in A denoting the set $\{a\}$

and for any regular expressions E and F,

- concatenation (EF) denoting the set of strings made by concatenating a string from E with a string from F
- alternation $(E + F)$ denoting the union of sets described by E and by F (also written $E|F$)
- Kleene star (E^*) denoting the set of all strings made by concatenating any finite number (including zero) of strings from E.