Little Languages

Lecture 23:

Constructing NFAs

Paper and pencil day!

© Kris Jordan 2019 - All Rights Reserved

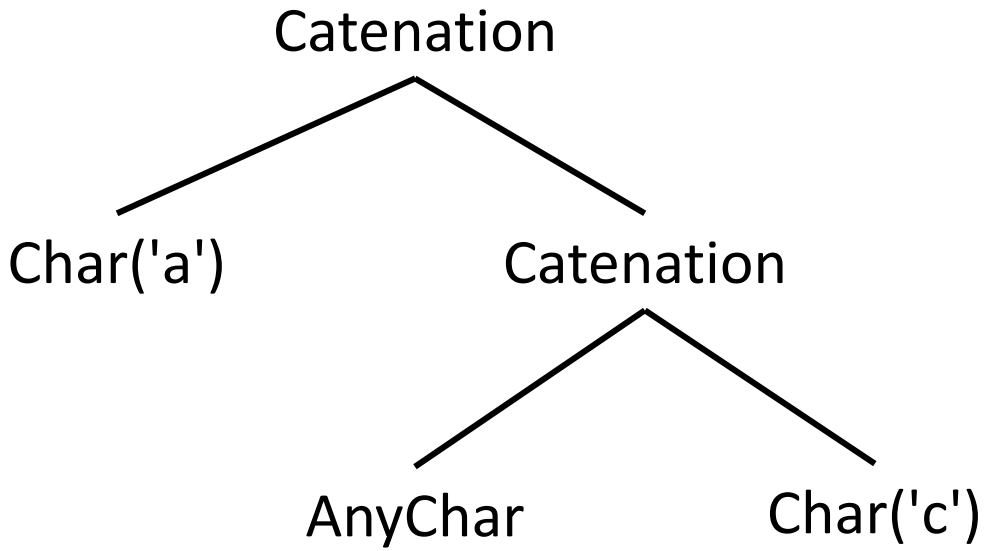
The Grammar and AST of `thegrep`

$$
\begin{aligned}
\langle \mathit{RegExpr} \rangle &::= \langle \mathit{Catenation} \rangle \ (\texttt{UnionBar} \ \langle \mathit{RegExpr} \rangle)? \\
\langle \mathit{Catenation} \rangle &::= \langle \mathit{Closure} \rangle \ \langle \mathit{Catenation} \rangle? \\
\langle \mathit{Closure} \rangle &::= \langle \mathit{Atom} \rangle \ \texttt{KleeneStar}? \\
\langle \mathit{Atom} \rangle &::= \texttt{LParen} \ \langle \mathit{RegExpr} \rangle \ \texttt{RParen} \mid \texttt{AnyChar} \mid \texttt{Char}
\end{aligned}
$$

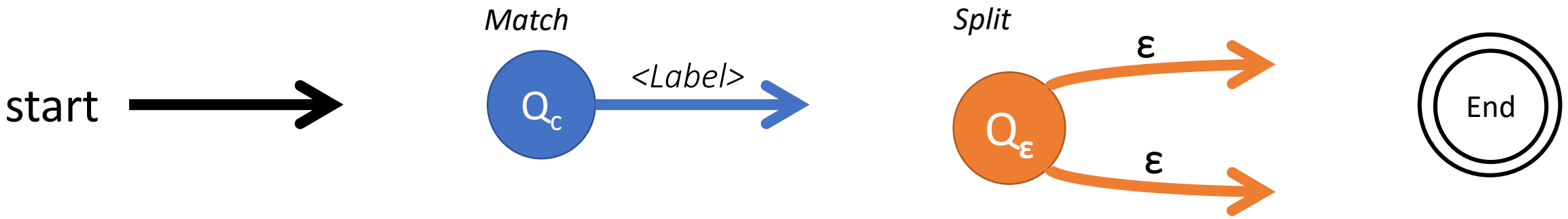
```
pub enum AST {
    Alternation(Box<AST>, Box<AST>),
    Catenation(Box<AST>, Box<AST>),
    Closure(Box<AST>),
    Char(char),
    AnyChar
}
```

Produce the AST for: a.c

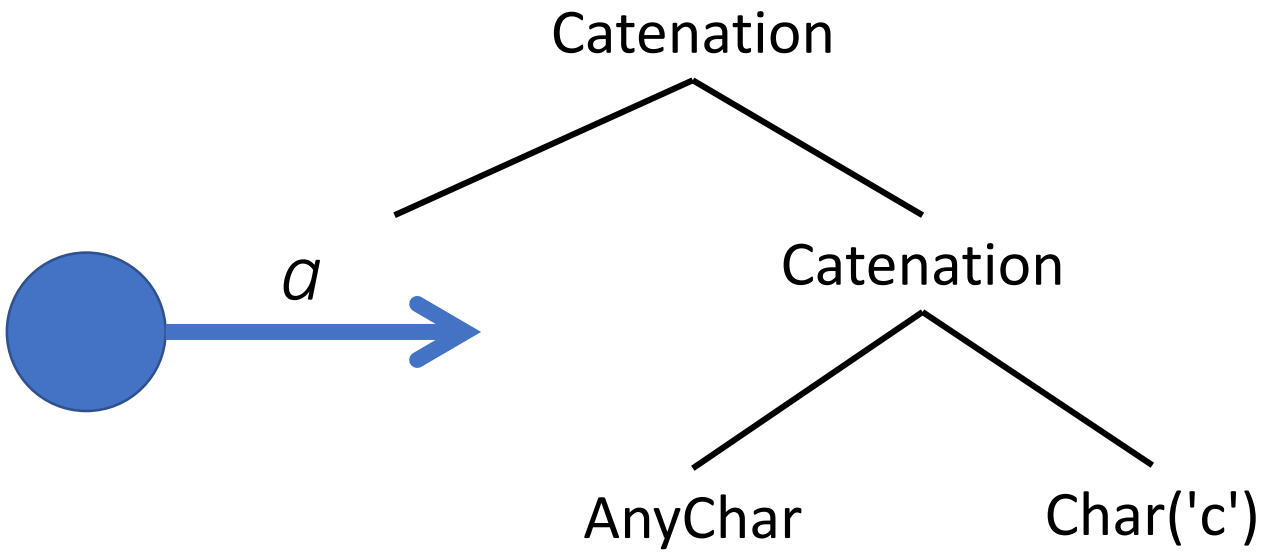
The AST for: a.c



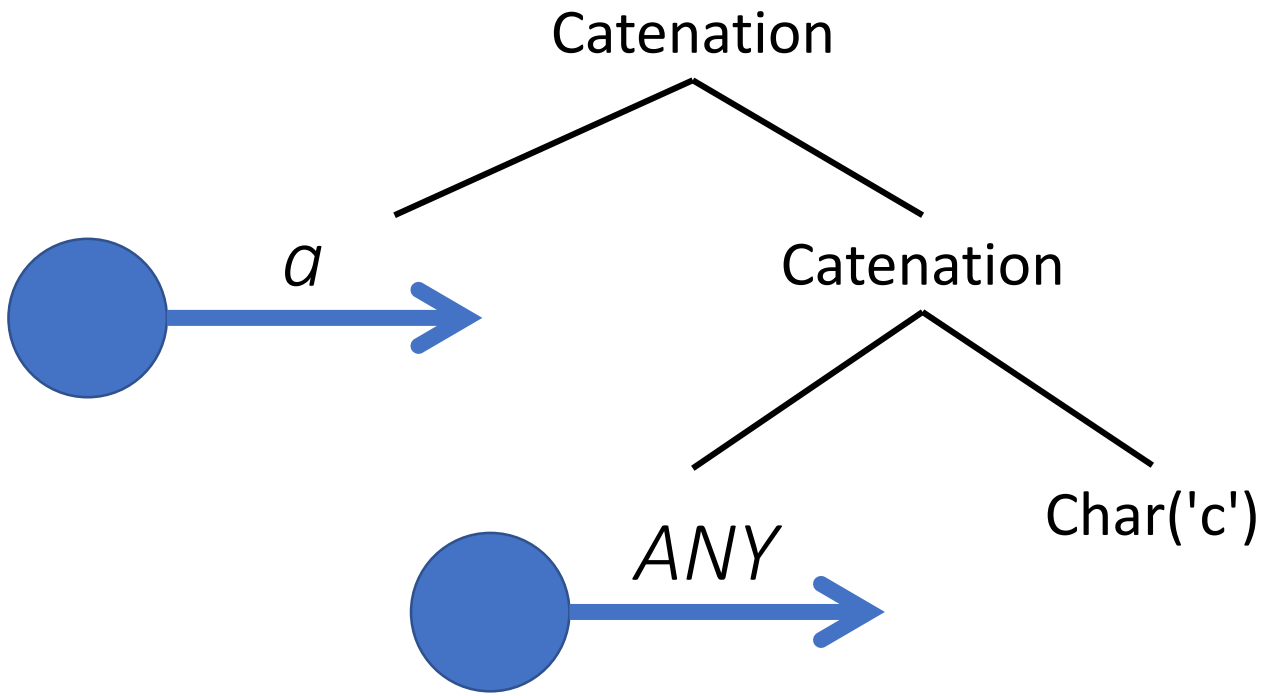
Hands-on: Draw an NFA transition diagram for **a.c** using only these State primitives:



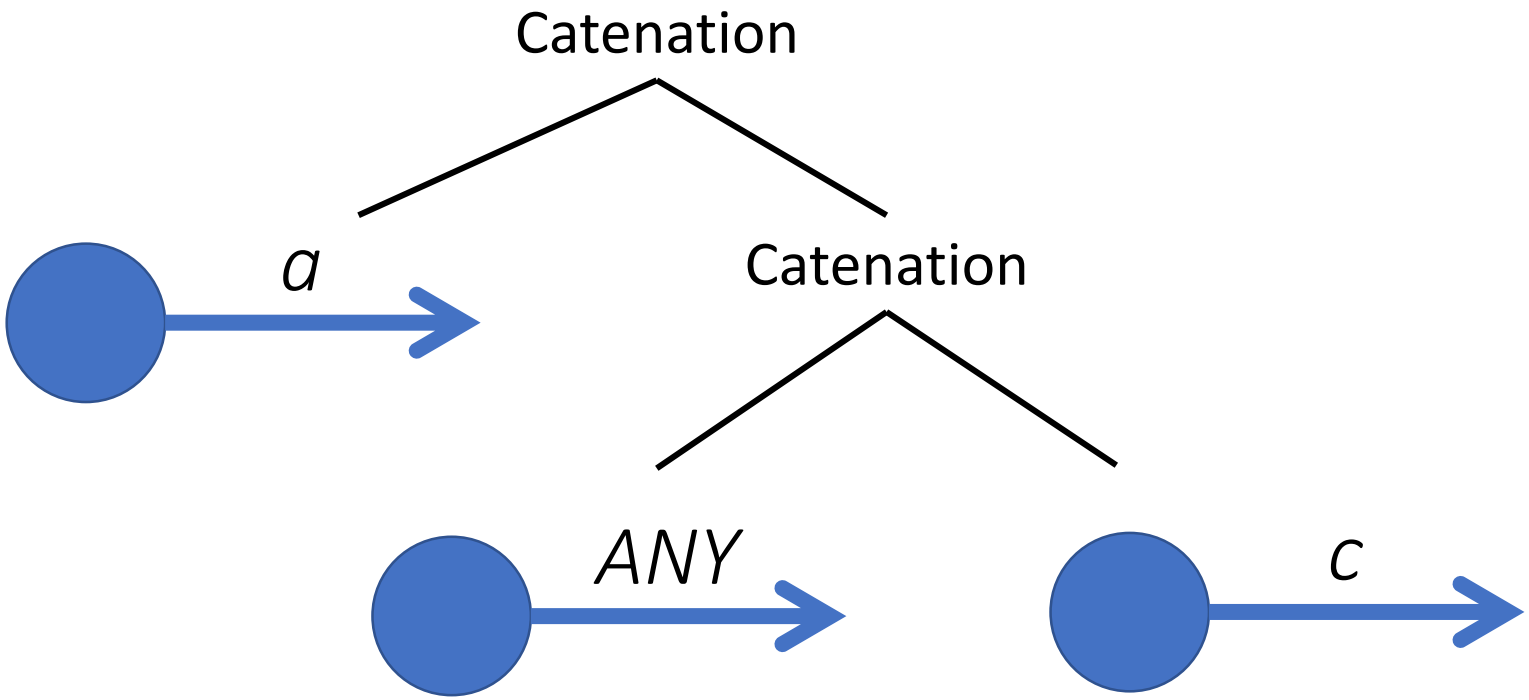
Produce the NFA for: a.c



Produce the NFA for: a.c



Produce the NFA for: a.c



Fragments

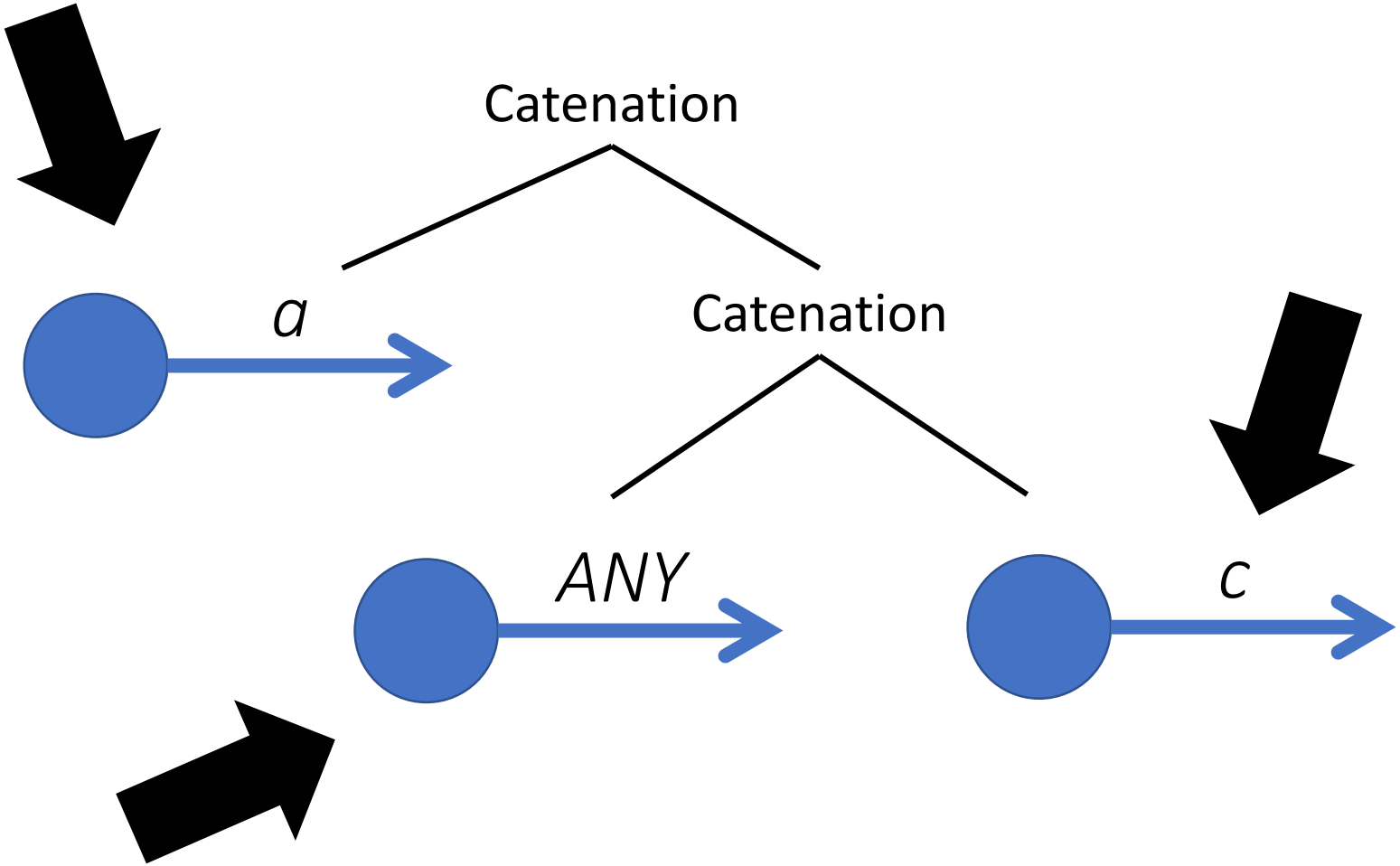
As the NFA is being constructed, fragments are being created and (later) joined together.

At this stage of the construction there are *three* fragments each containing a single state.

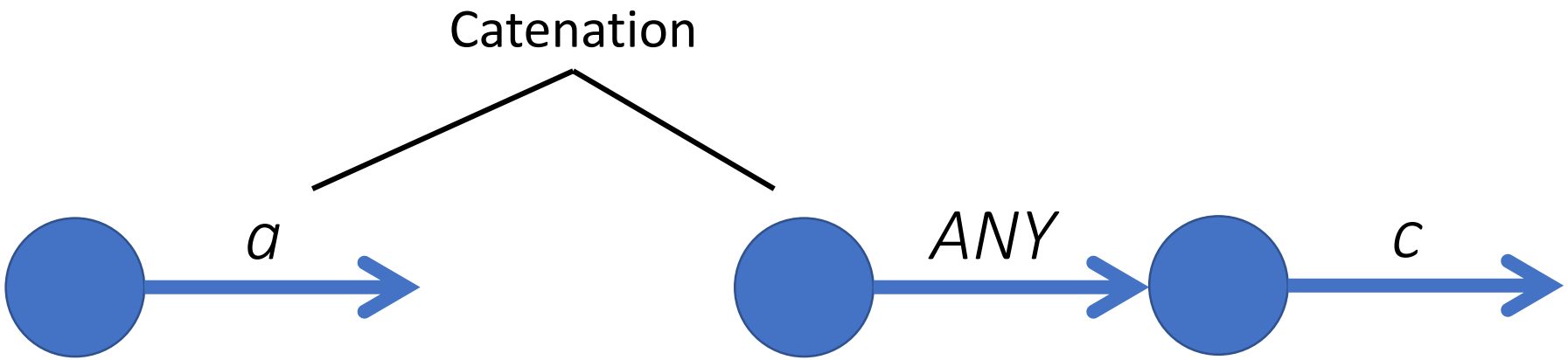
Every fragment is made up of:

1. Its start state
2. Its unconnected end state(s)

At this point, each fragment is just the single state.



Produce the NFA for: a.c

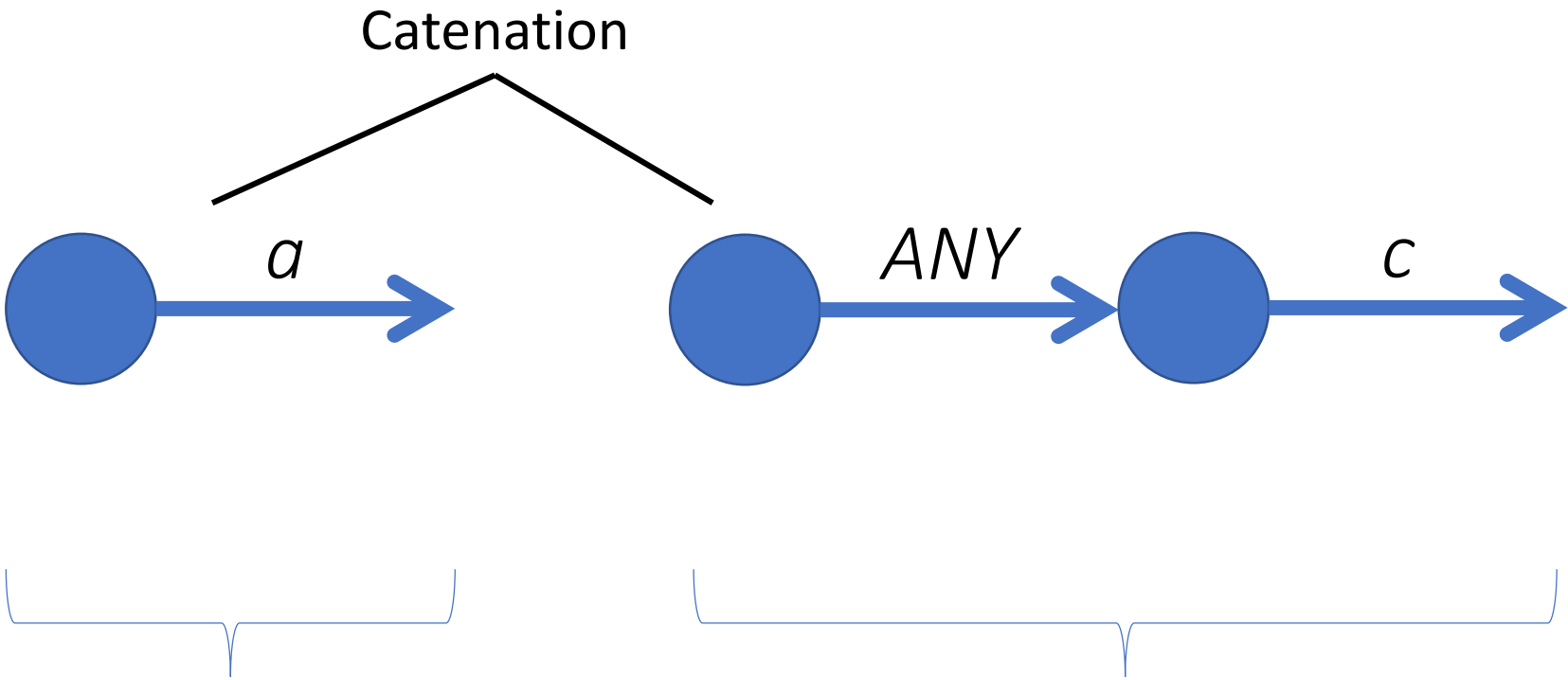


Fragments

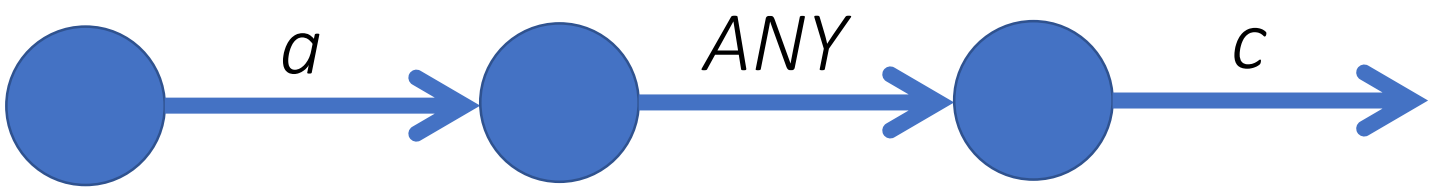
Notice the operation of catenation joins its two fragments into a single fragment.

Catenation joins the end of its left hand side with the start of its right hand side.

At this point there are two fragments. Notice the second fragment's start and end states are no longer the same.



Produce the NFA for: a.c



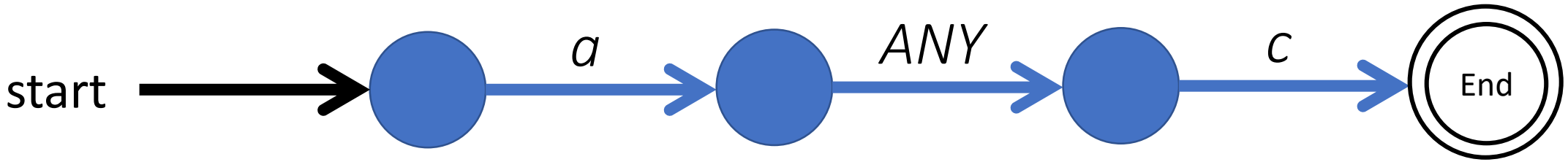
Fragments

Once again, a catenation operator is evaluated by joining the end(s) of the fragment on its left-hand side with the start of the fragment on the right-hand side.

At this point, the AST has been traversed in its entirety and a single fragment results.

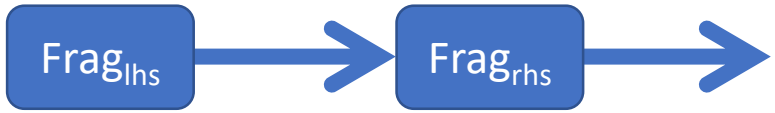
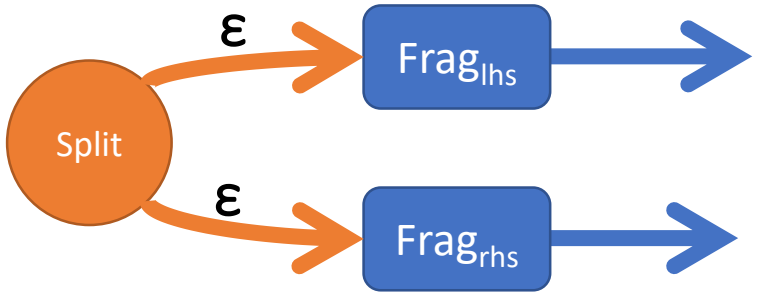
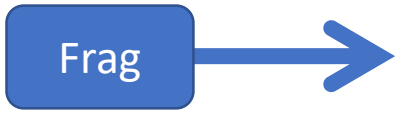
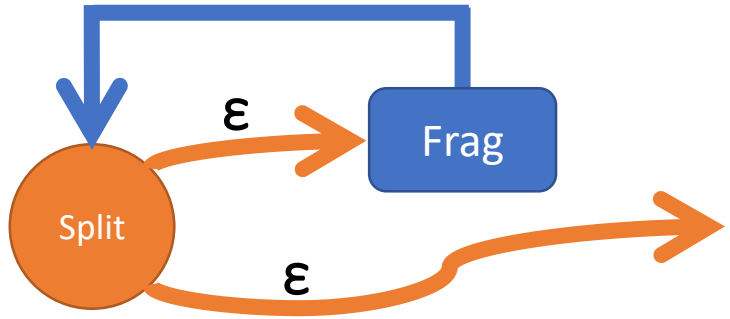
This fragment's start state is labelled 1 and its end(s) state is labelled 3.

Produce the NFA for: a.c



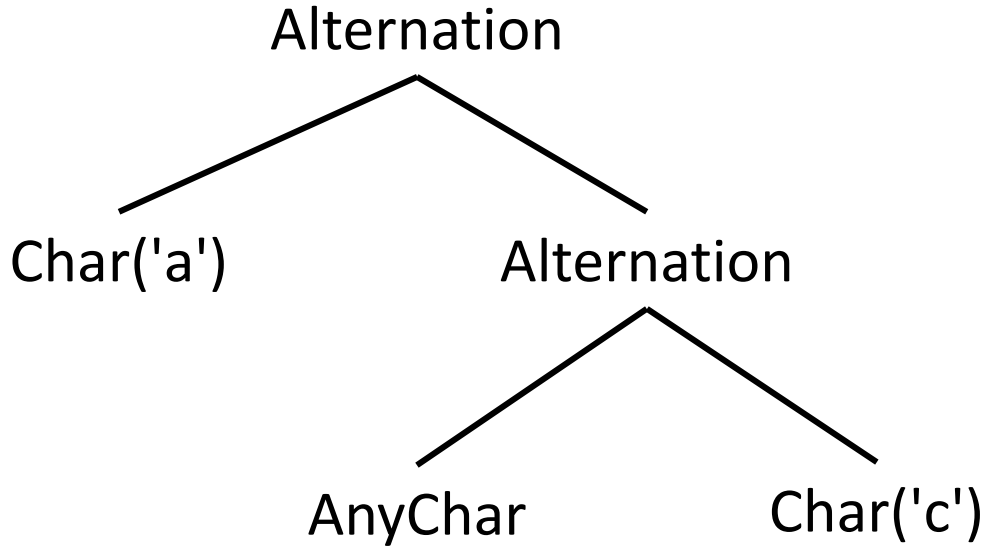
Finally, a **Start State** is added and joined to the start of the resulting Fragment.

The end state is also added and the ends of the fragment are joined to it.

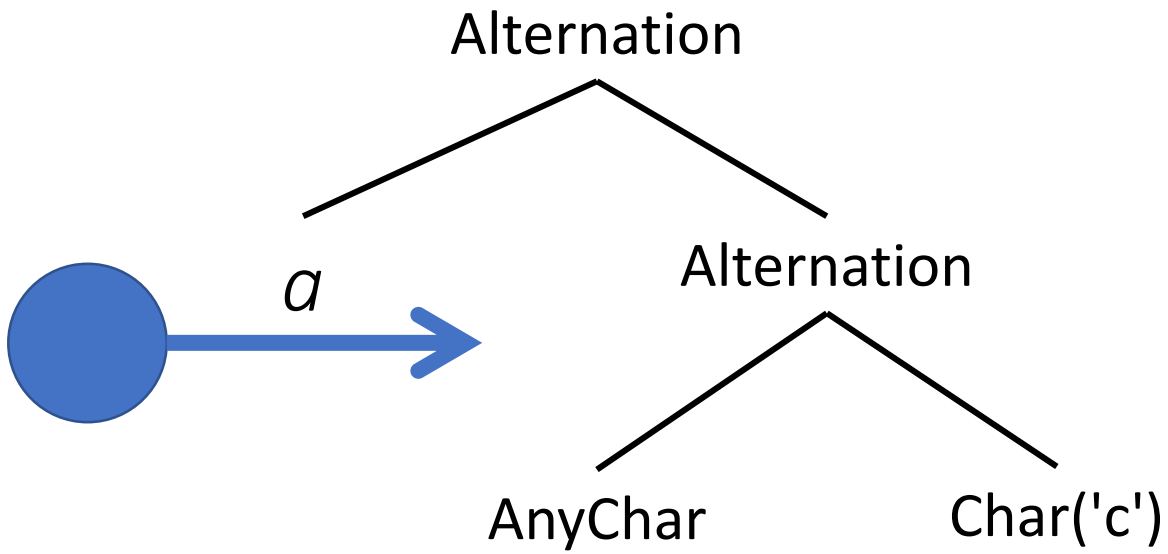
	Before	After
Catenation		
Alternation		
Closure		

Produce the AST for **a|.|c**

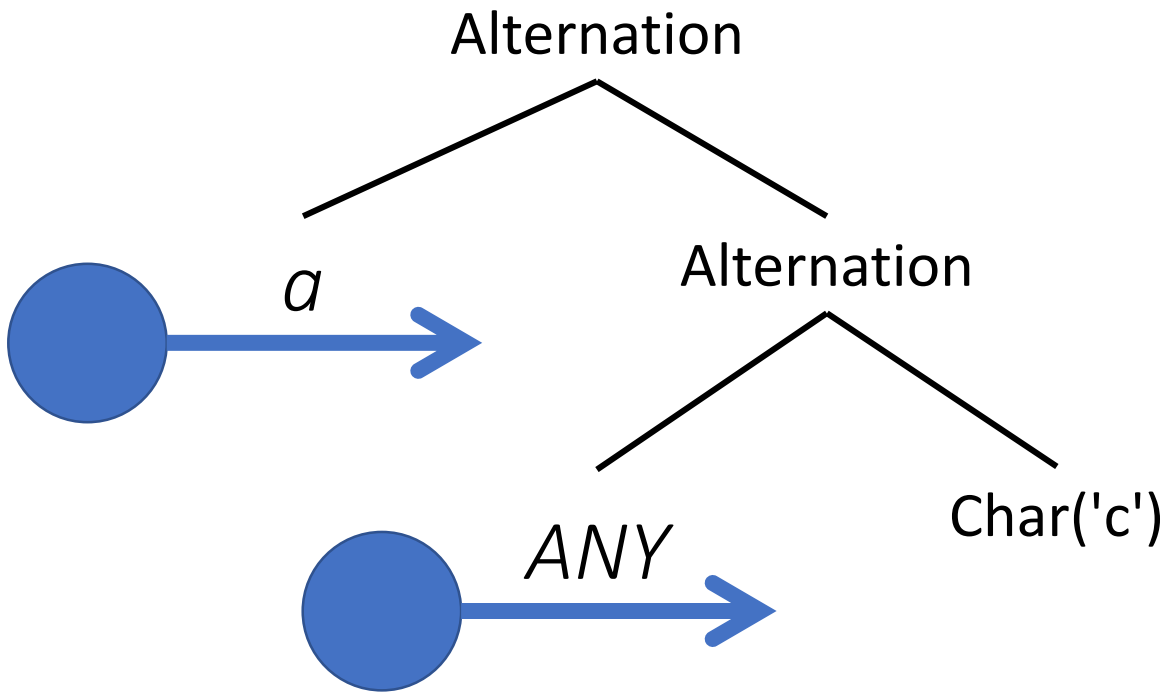
Produce the AST for **a|.|c**



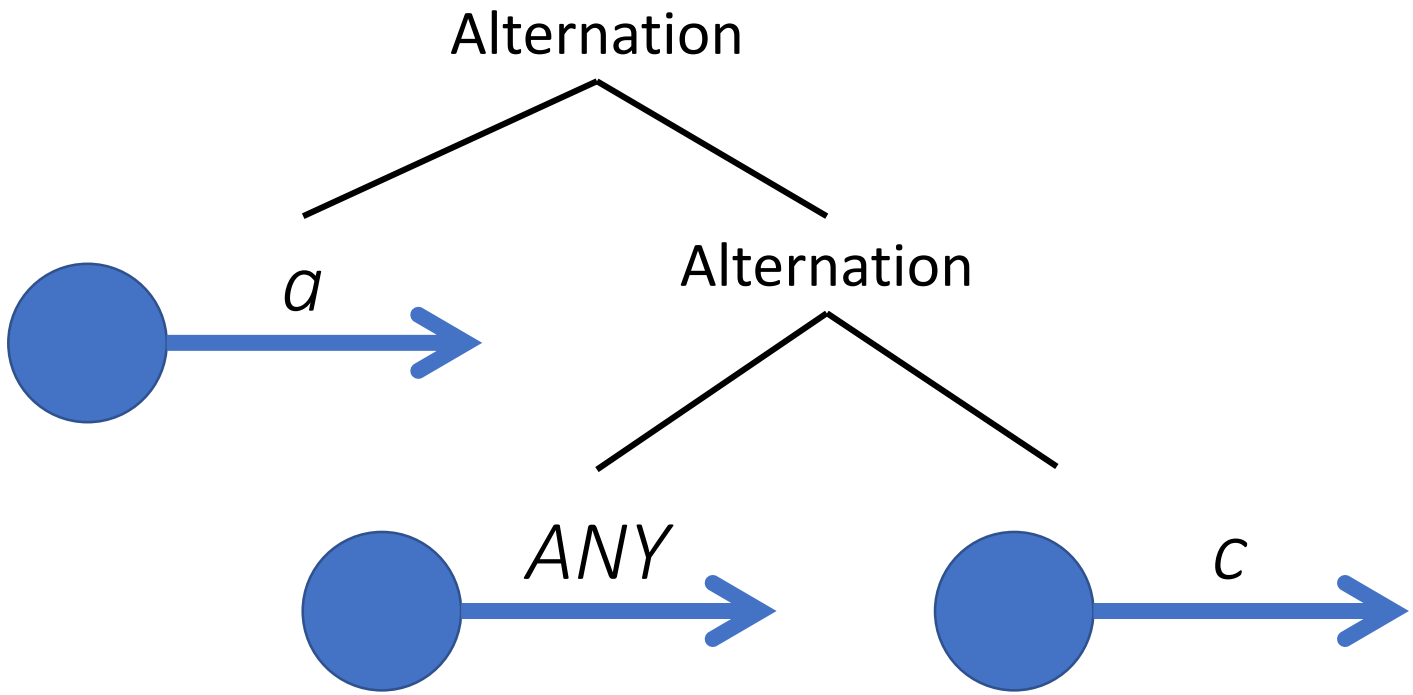
Produce the NFA for **a|.|c**



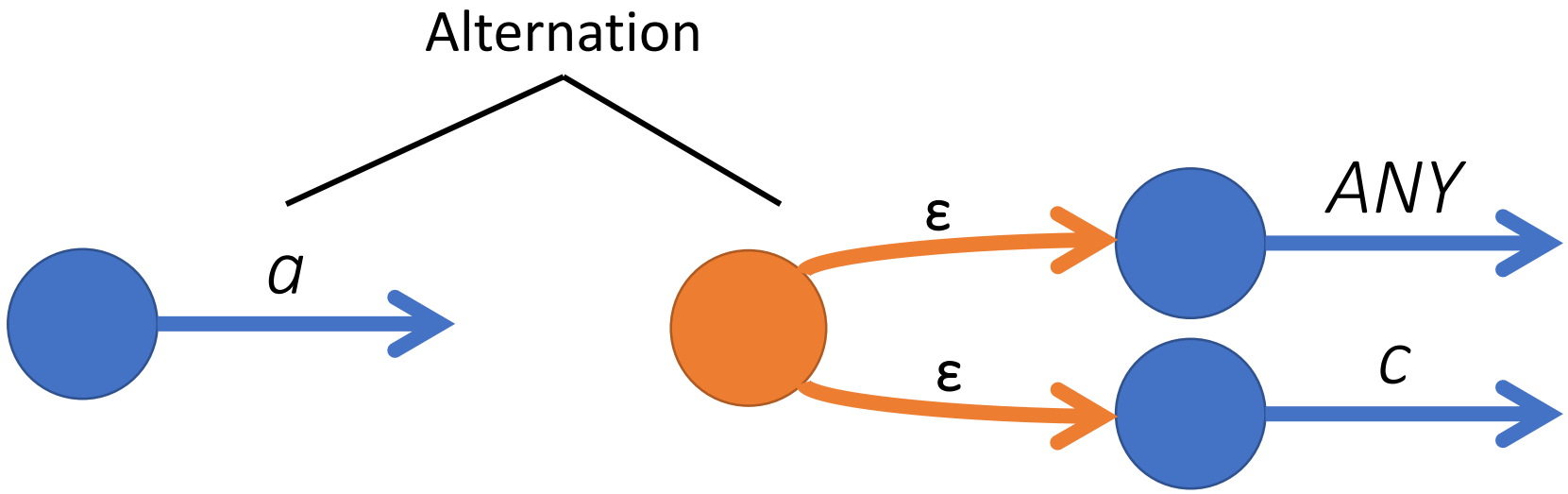
Produce the NFA for **a|.|c**



Produce the NFA for **a|.|c**



Produce the NFA for **a|.|c**

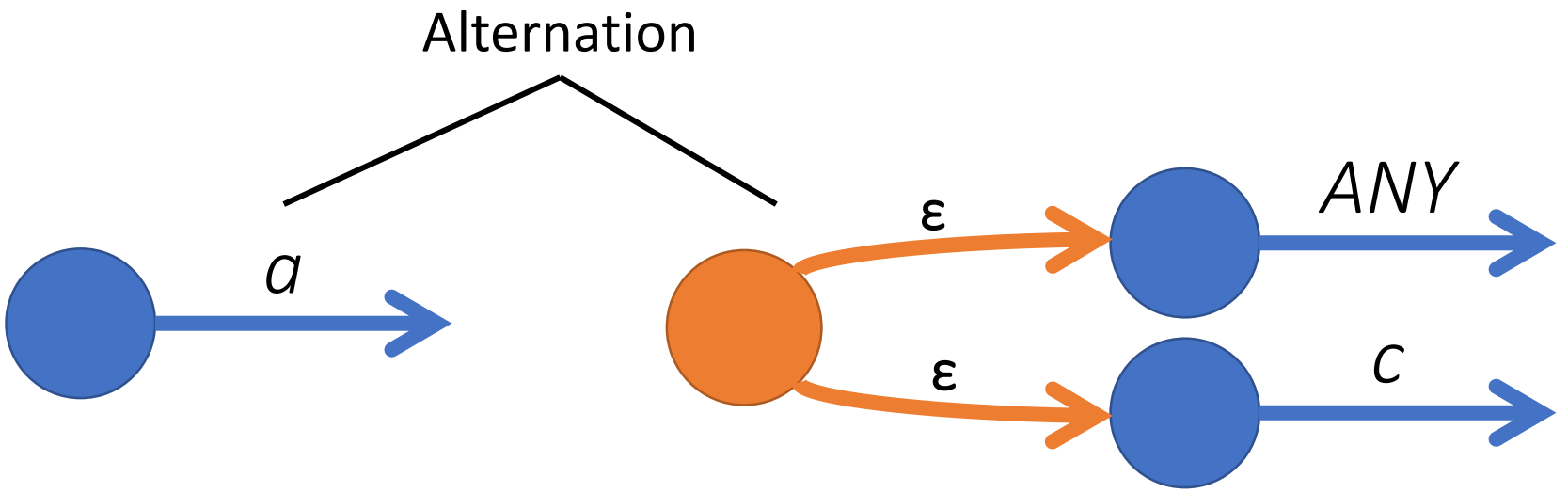


Fragments

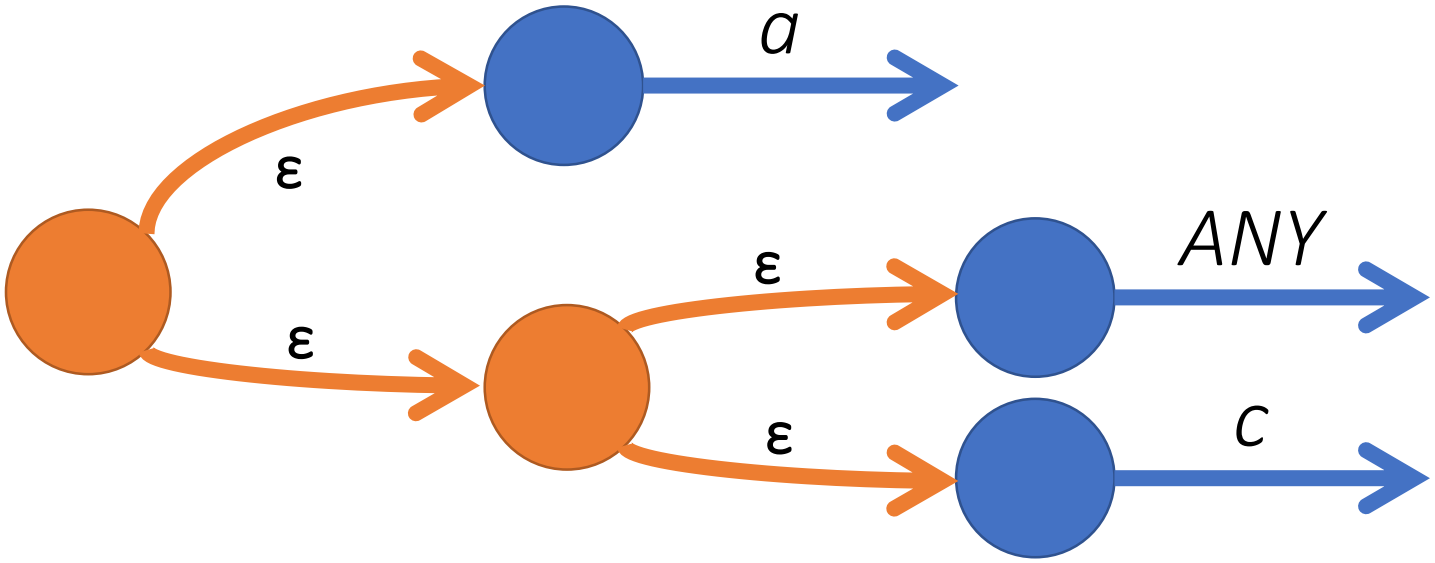
Notice the alternation transformation introduces a *new Split state.*

The Split state has epsilon transitions to the NFAs generated by its left-hand side and right-hand side.

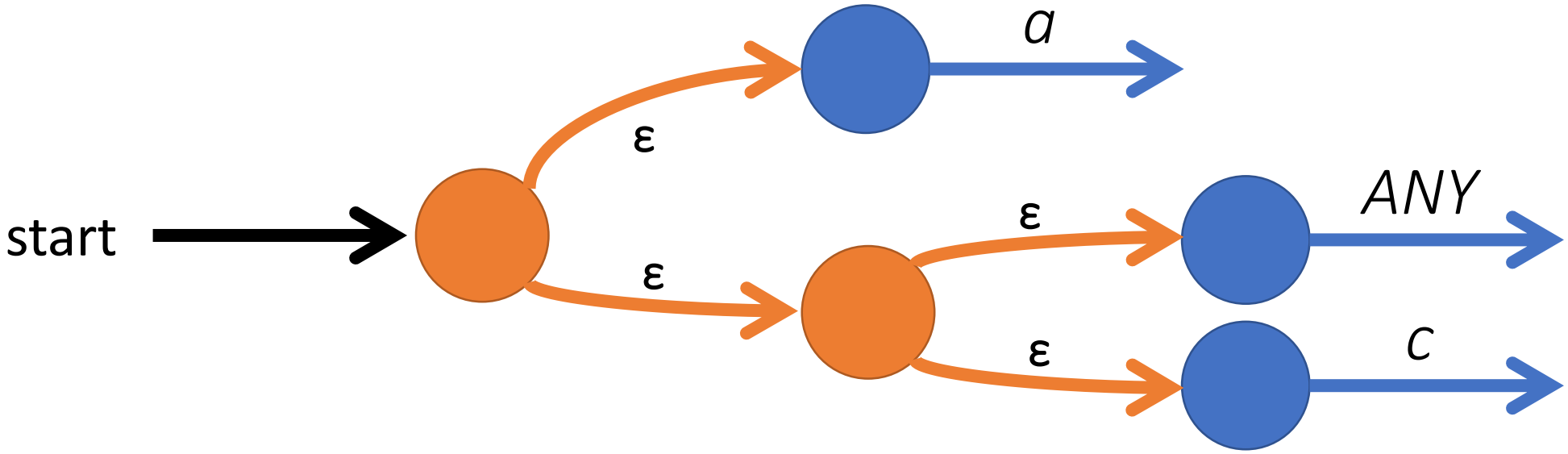
The right fragment illustrates one with *multiple ends.*



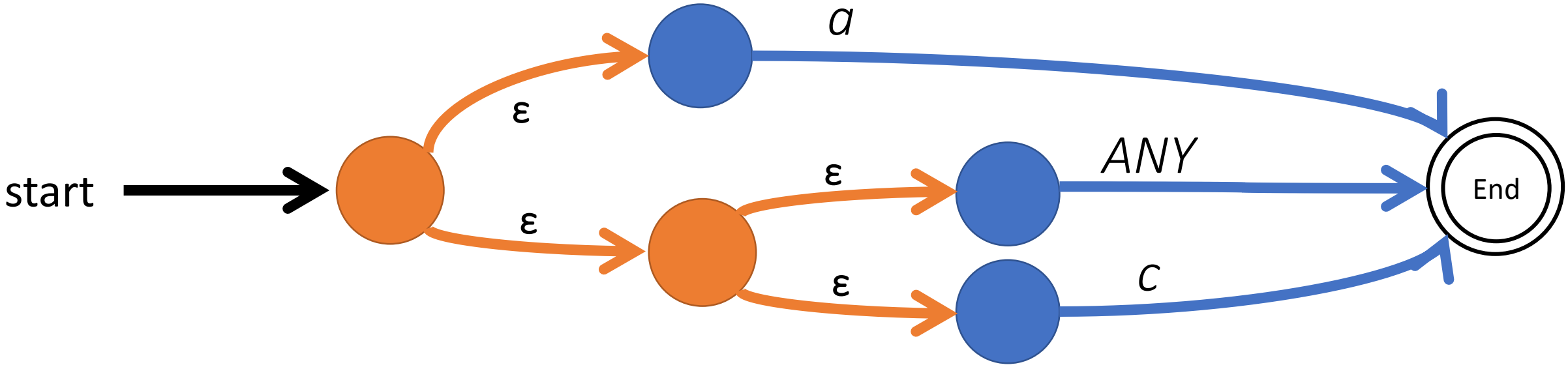
Produce the NFA for **a|.|c**



Produce the NFA for **a|.|c**



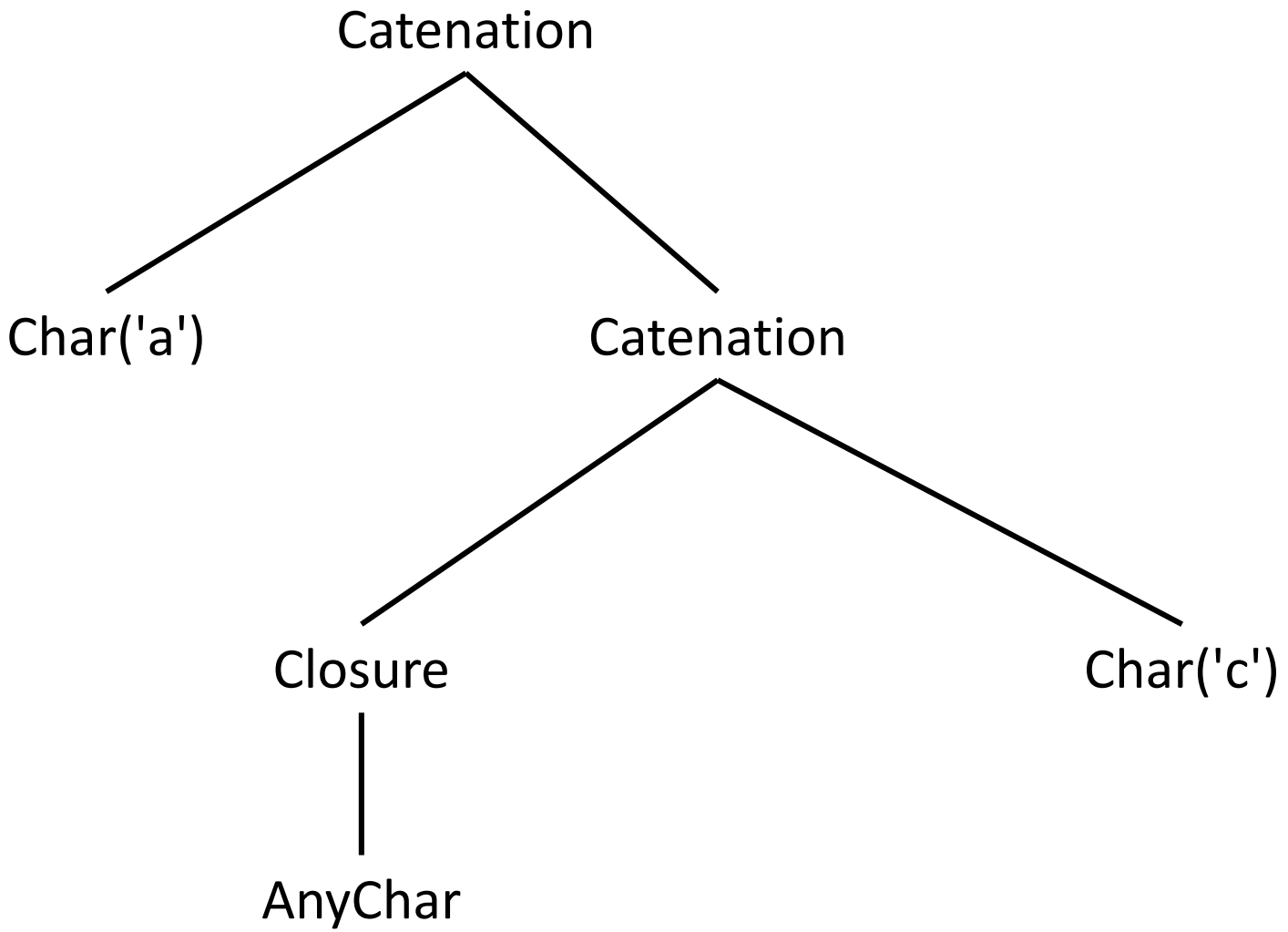
Produce the NFA for **a|.|c**



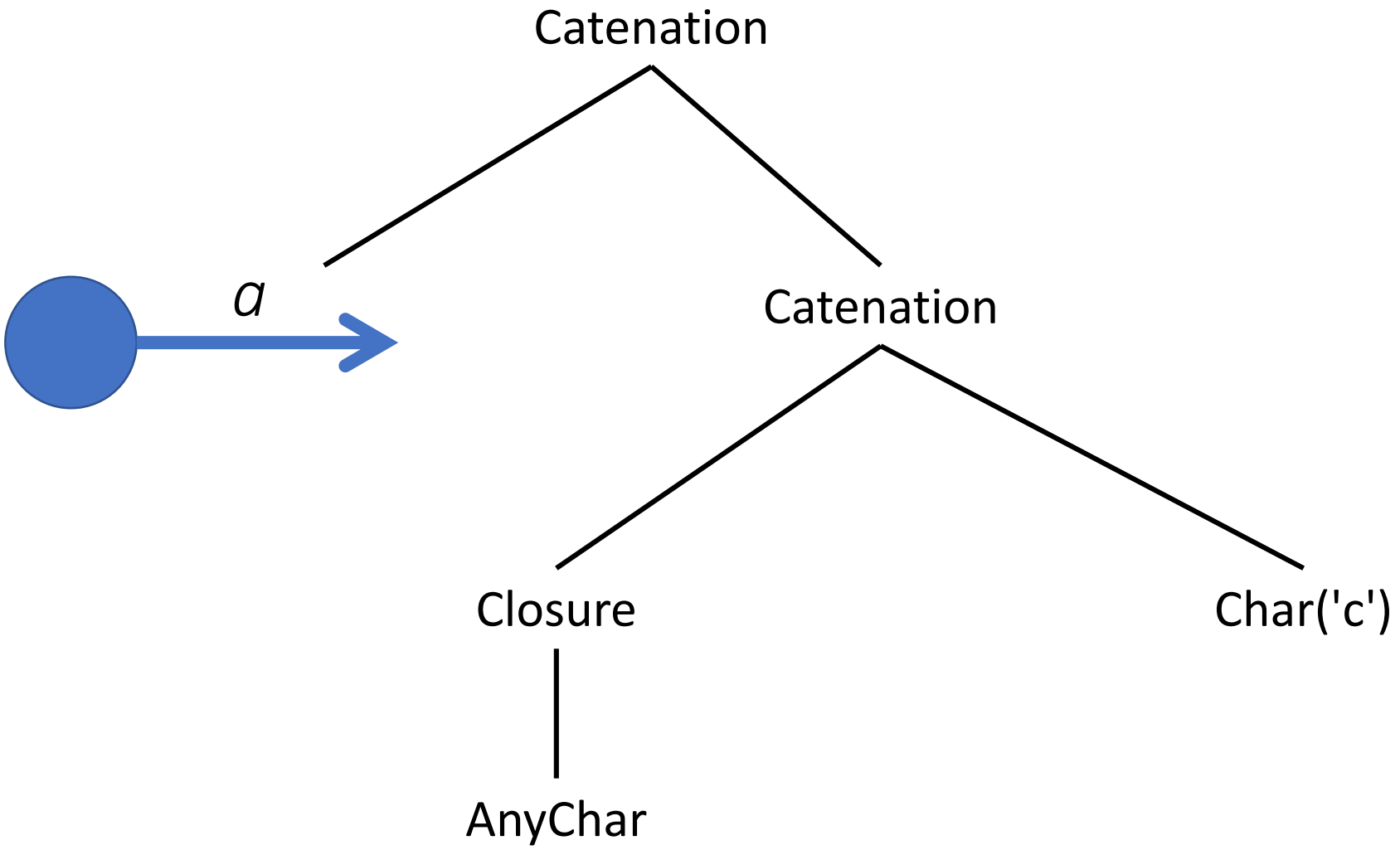
Notice to join the *body* fragment to the *end* state, all 3 of the fragment's ends needed to be joined to the *end* state.

Produce the AST for **a.*c**

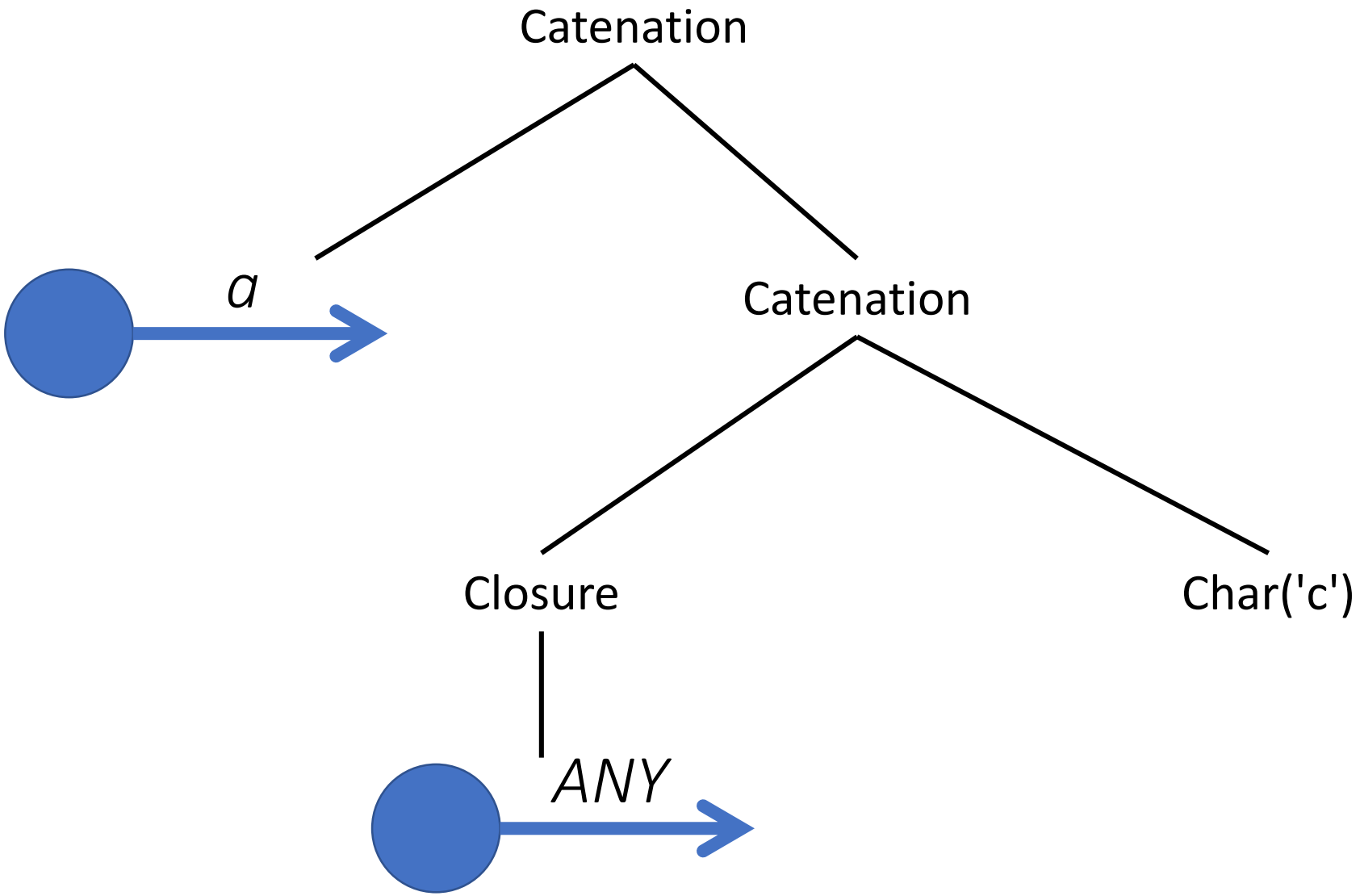
Produce the NFA for **a.*c**



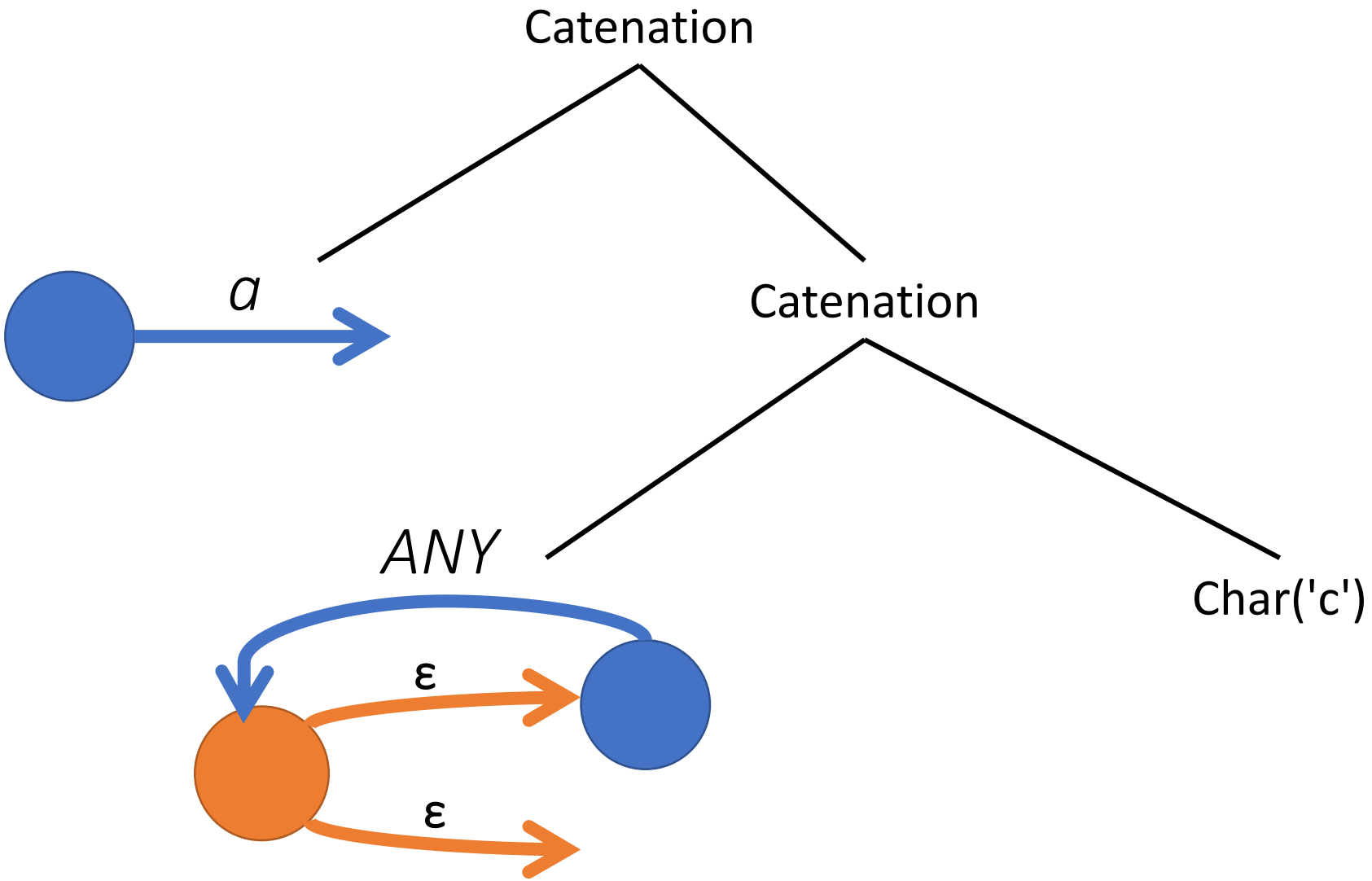
Produce the NFA for **a.*c**



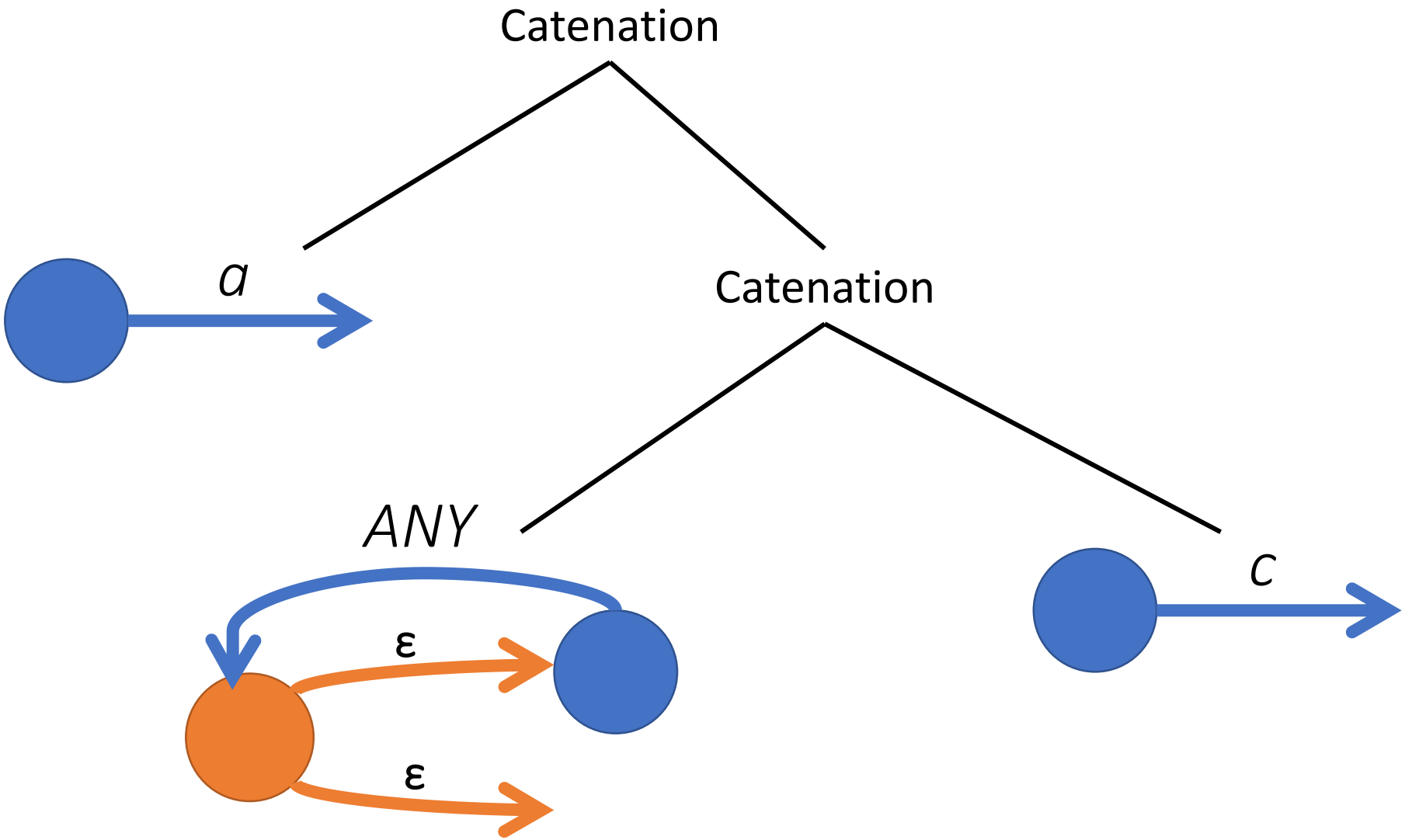
Produce the NFA for **a.*c**



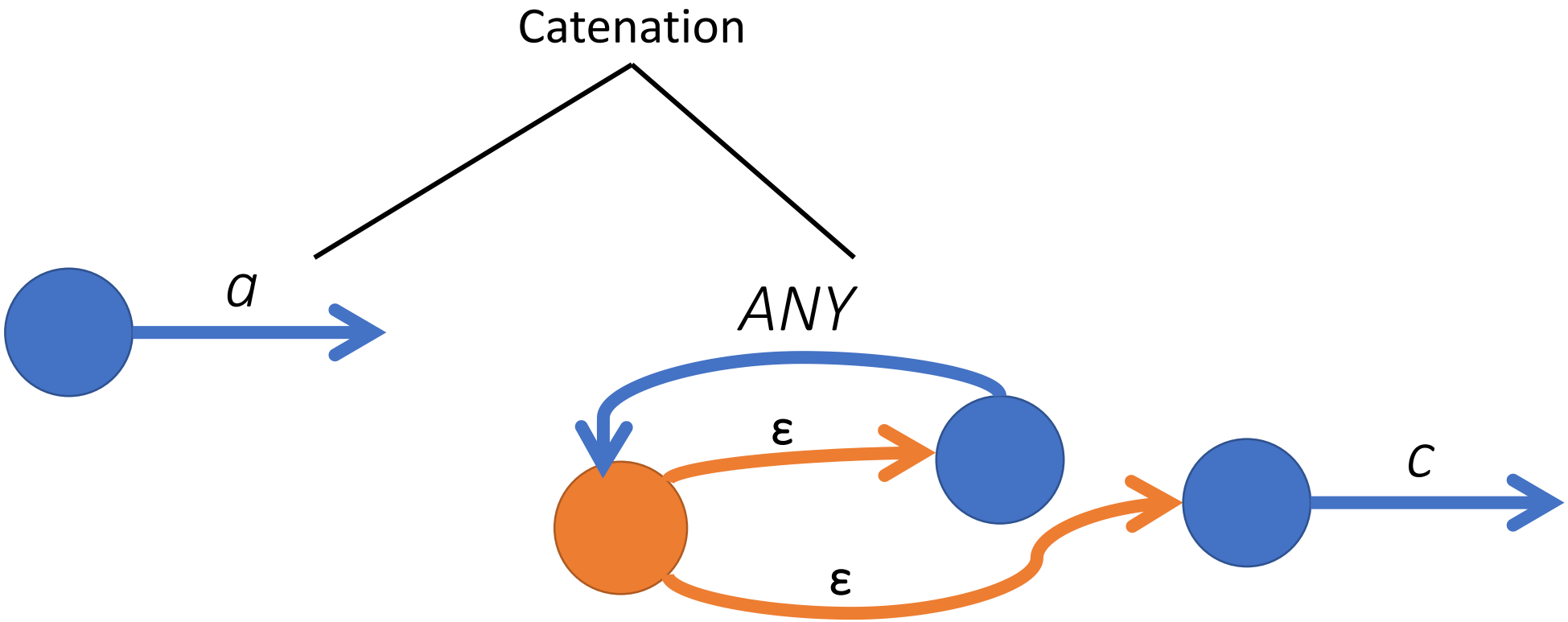
Produce the NFA for **a.*c**



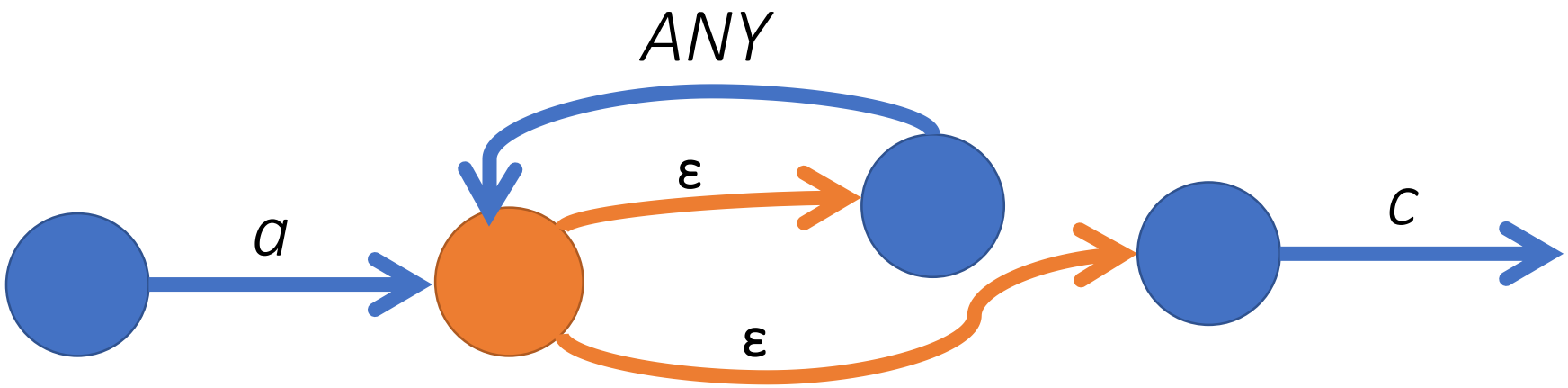
Produce the NFA for **a.*c**



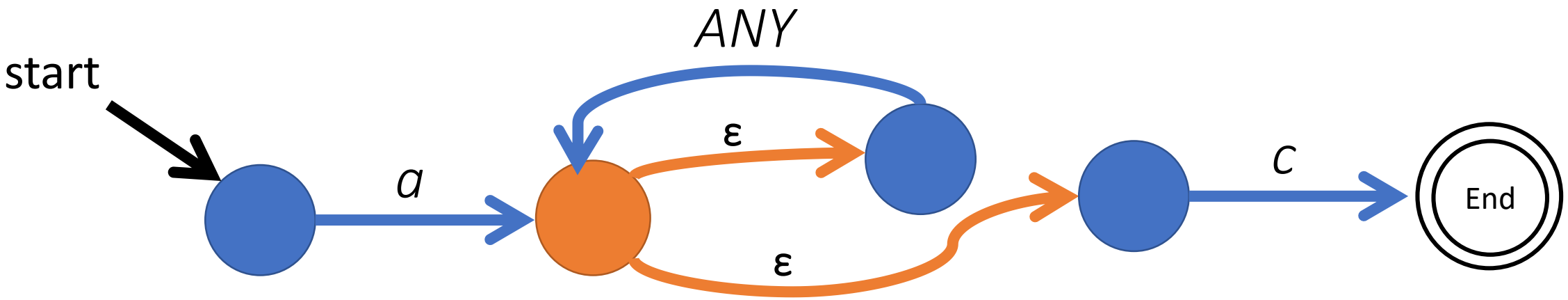
Produce the NFA for **a.*c**



Produce the NFA for **a.*c**

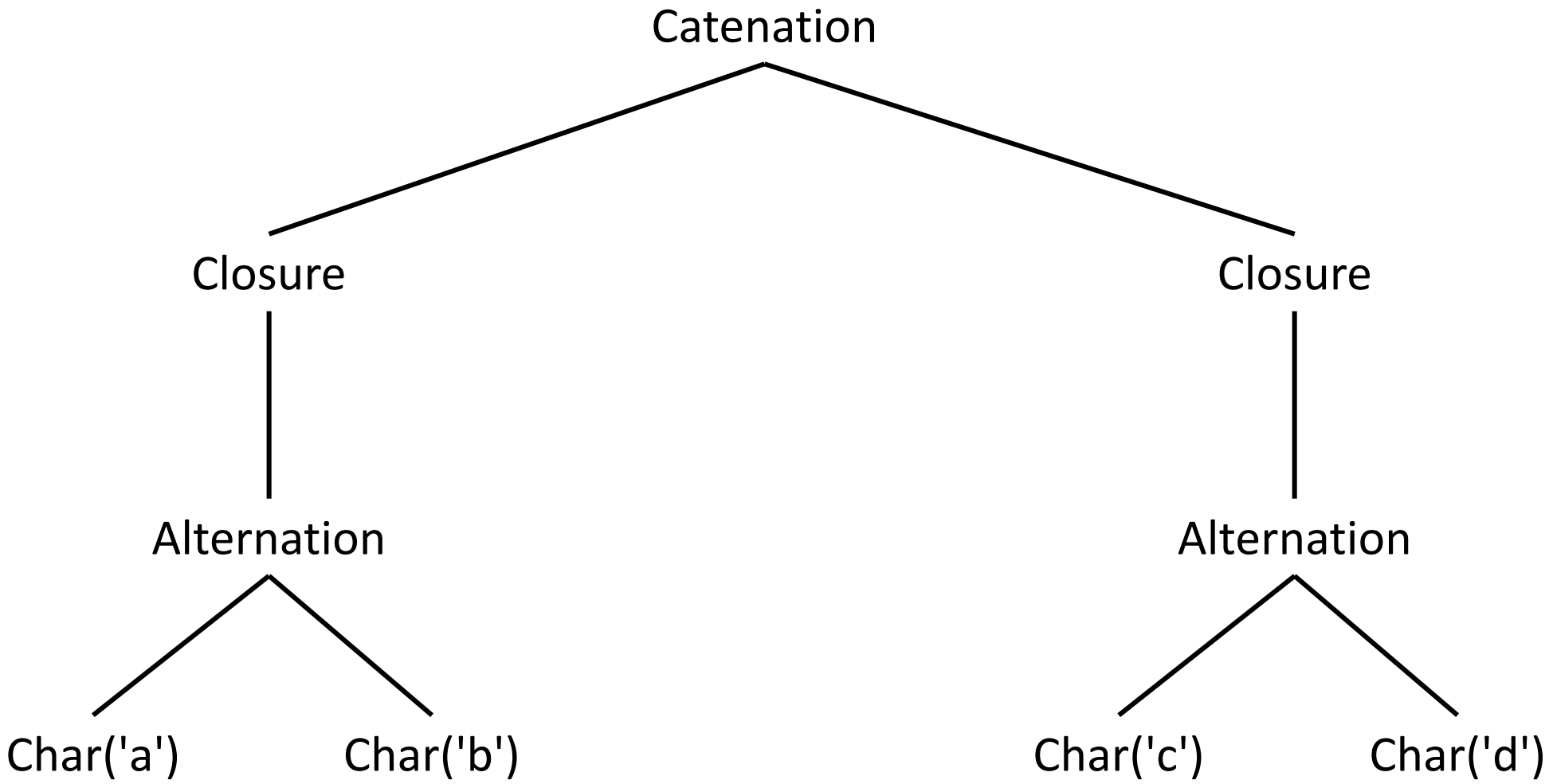


Produce the NFA for **a.*c**

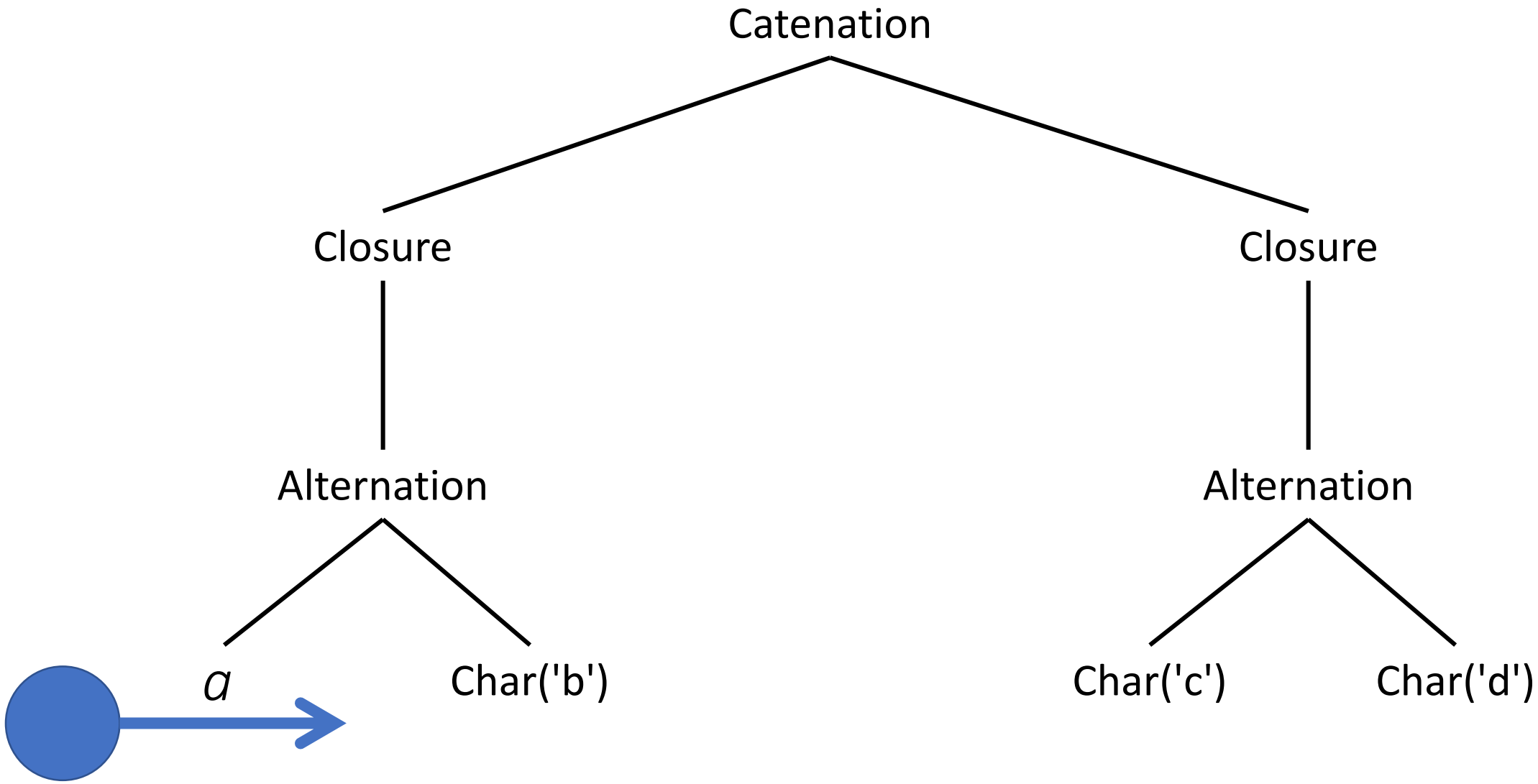


Produce the AST for **(a|b)*(c|d)***

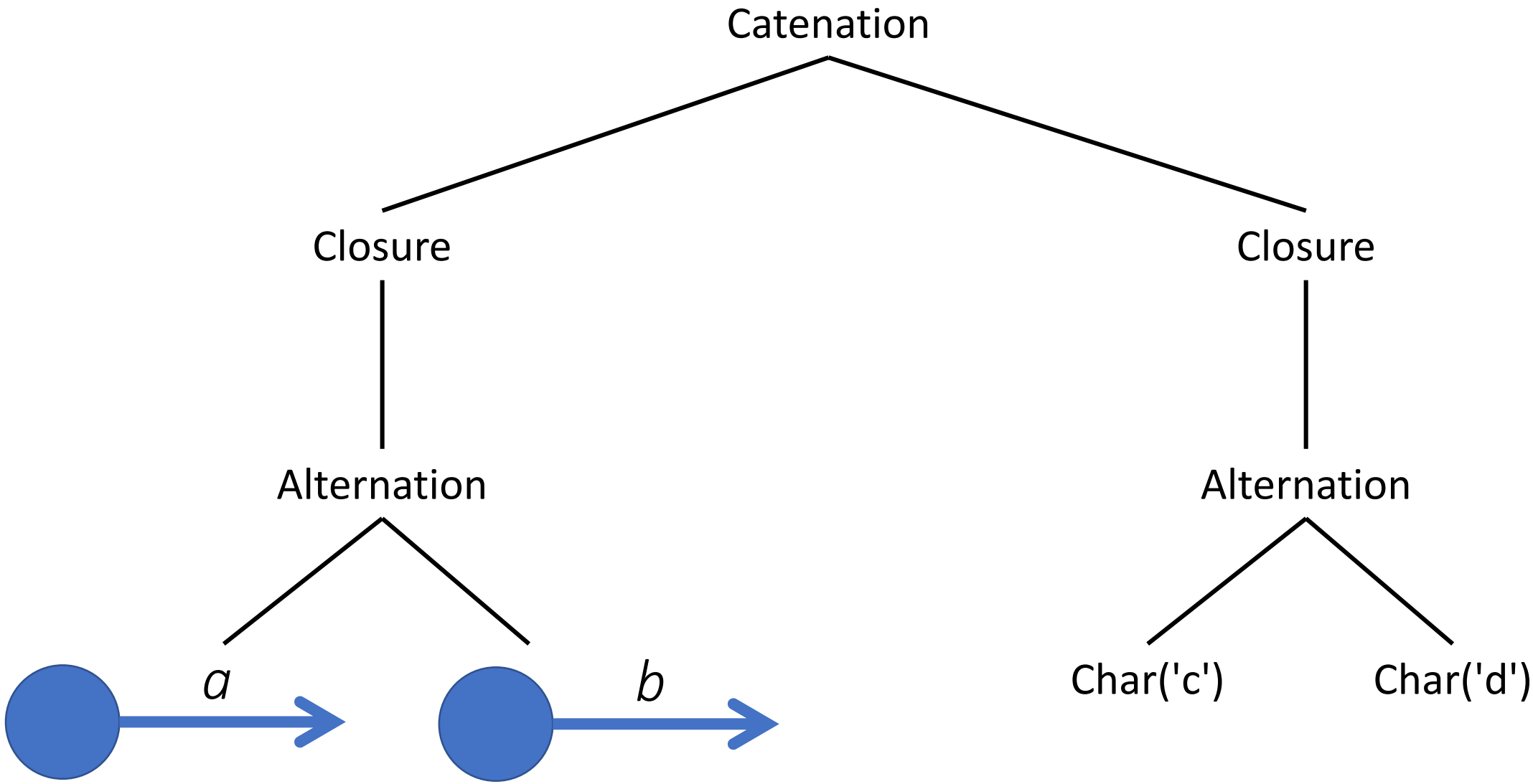
Produce the AST for **(a|b)*(c|d)***



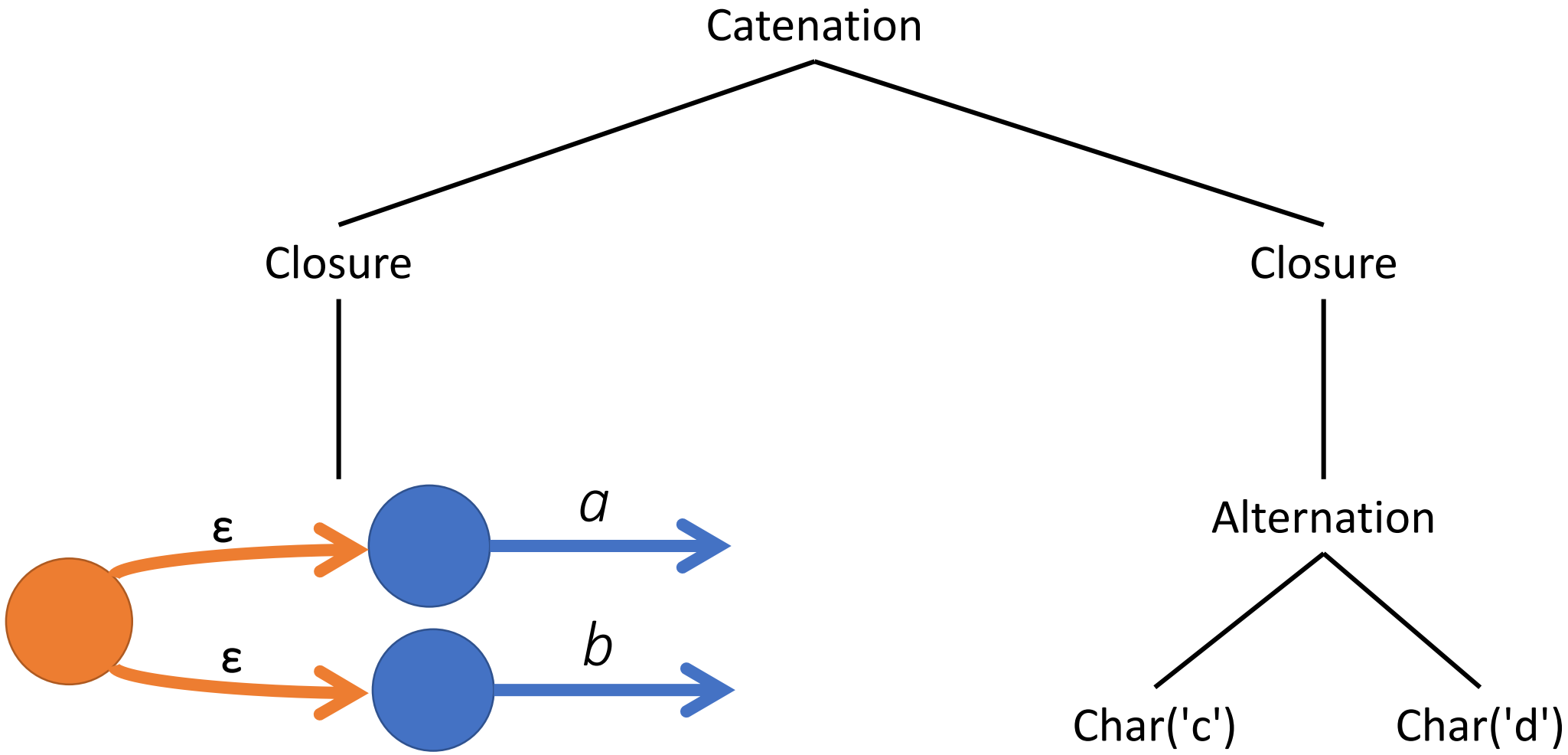
Produce the AST for **(a|b)*(c|d)***



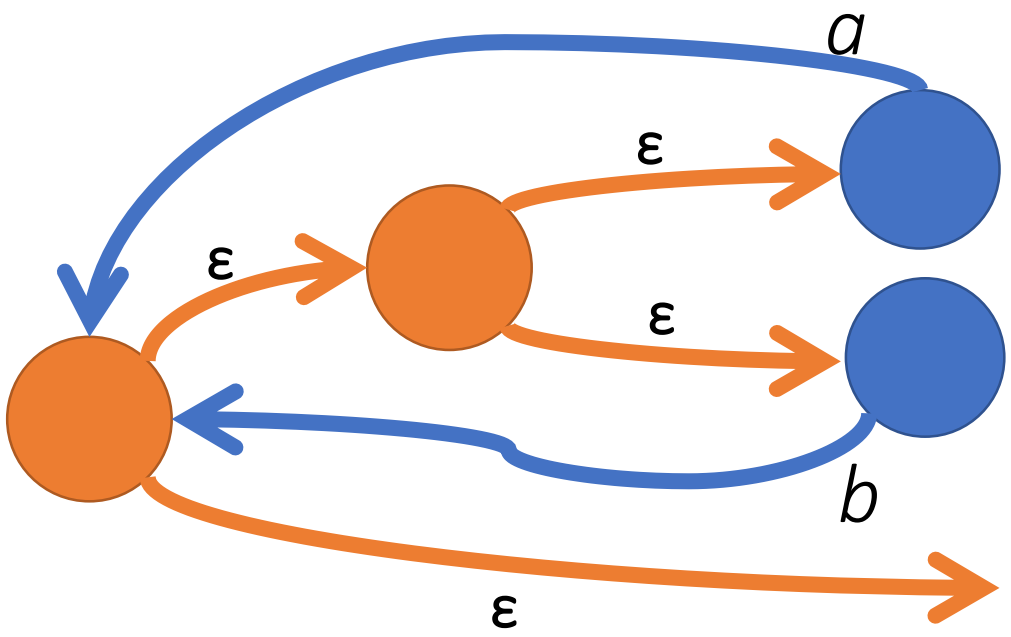
Produce the AST for **(a|b)*(c|d)***

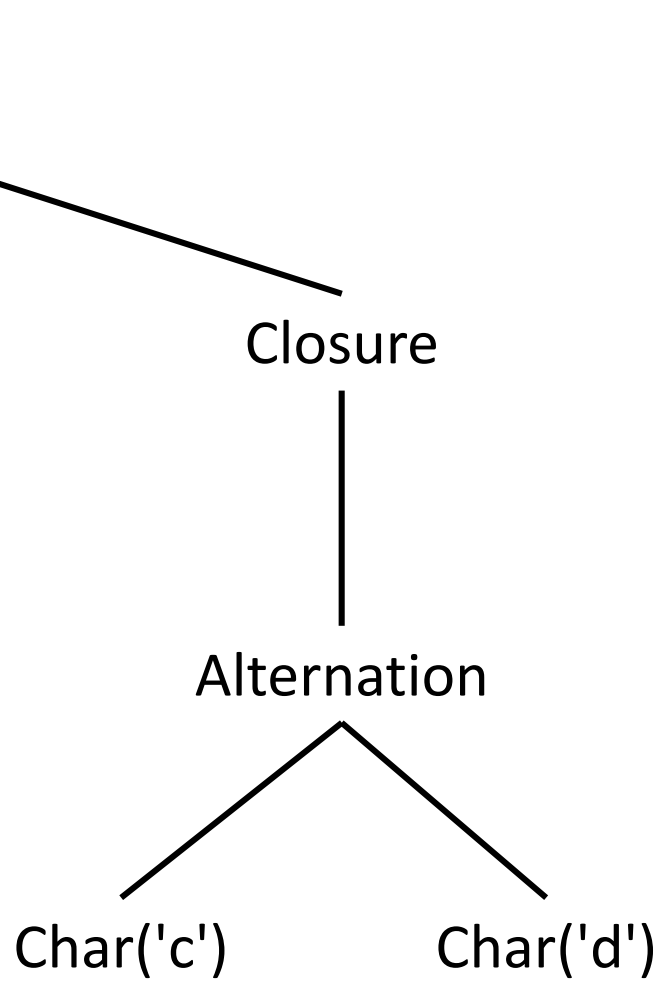


Produce the AST for **(a|b)*(c|d)***

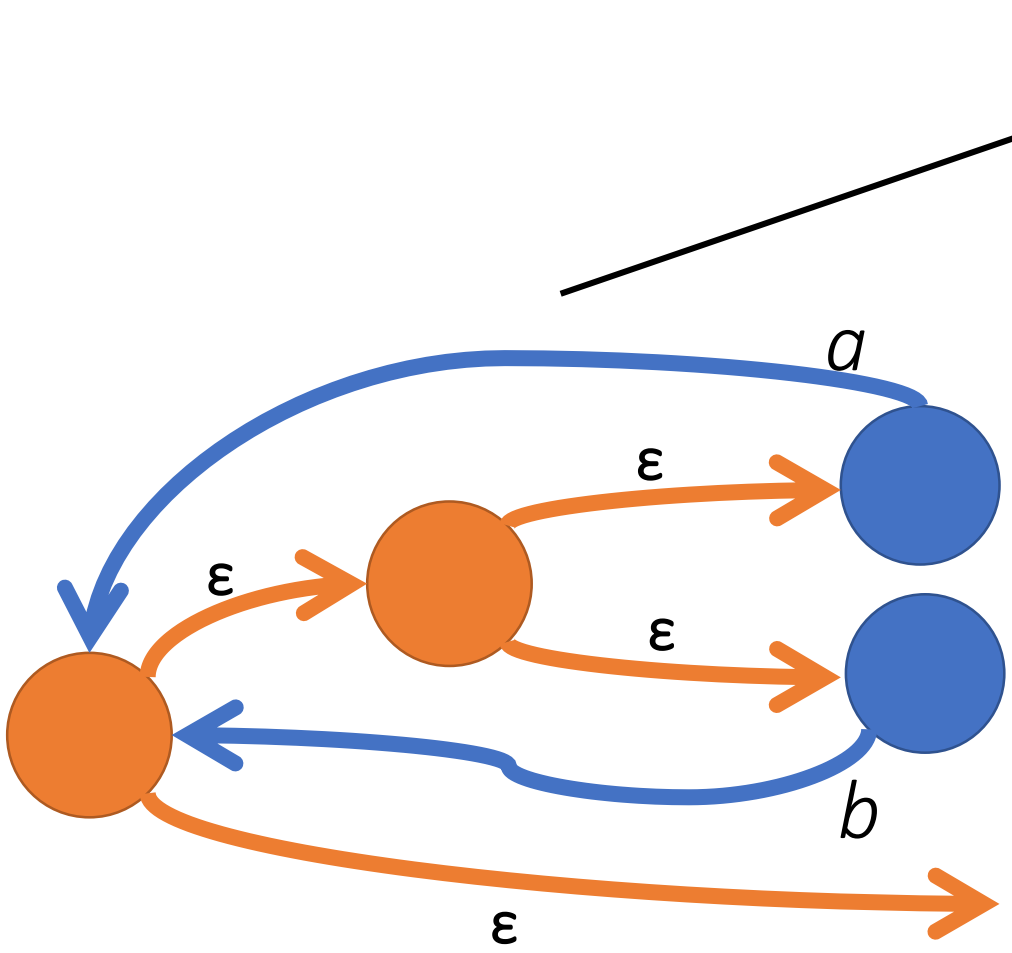


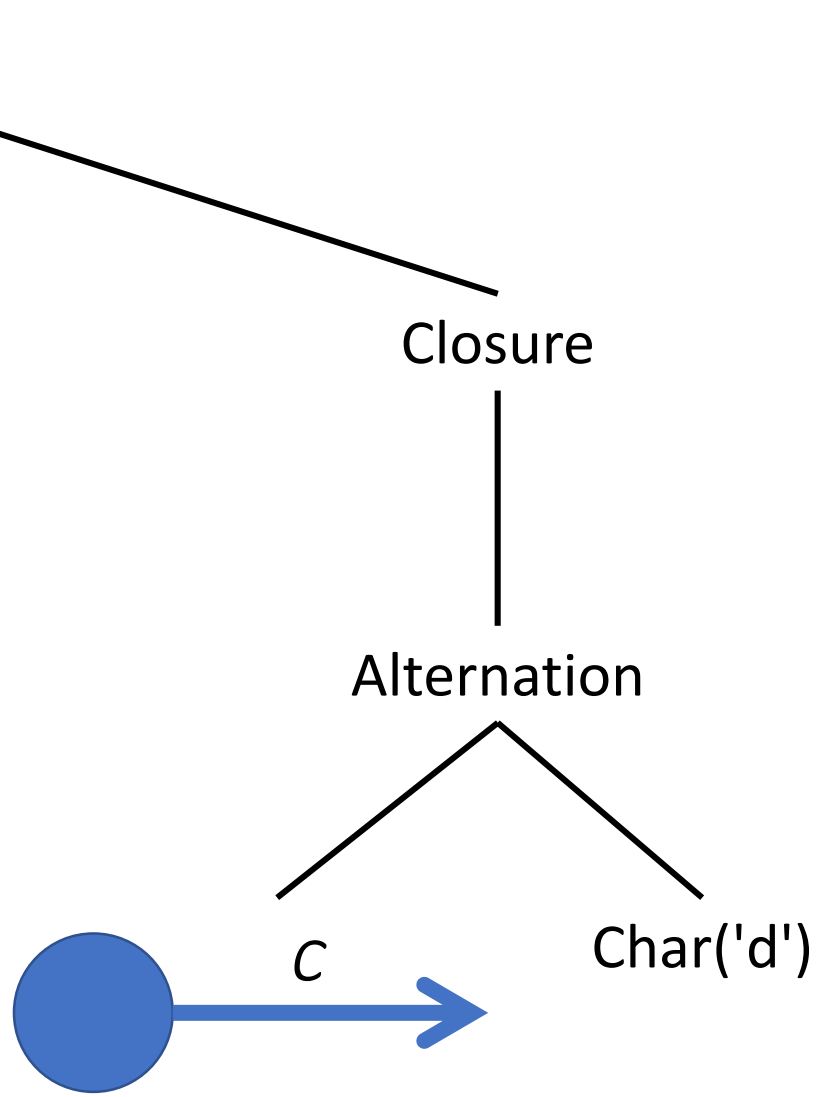
Produce the AST for **(a|b)*(c|d)***



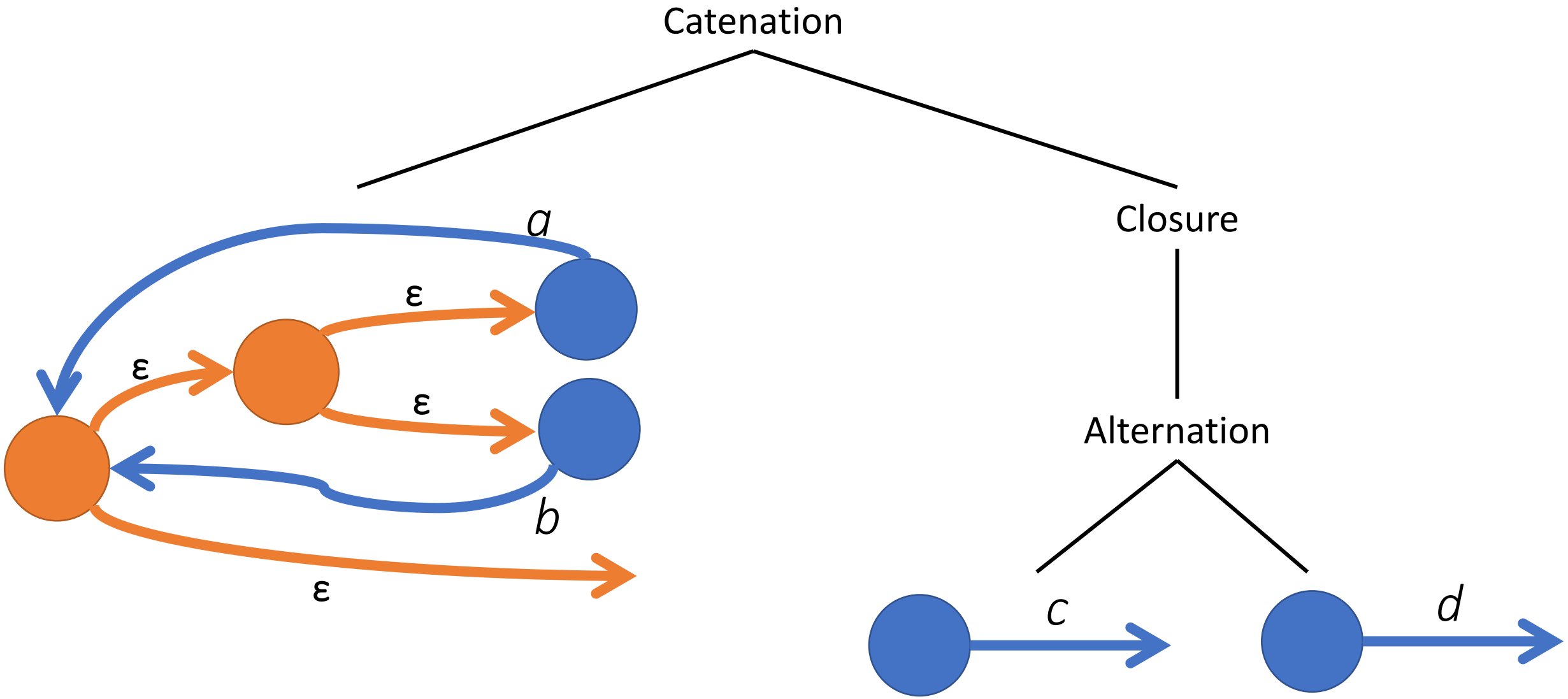


Produce the AST for **(a|b)*(c|d)***

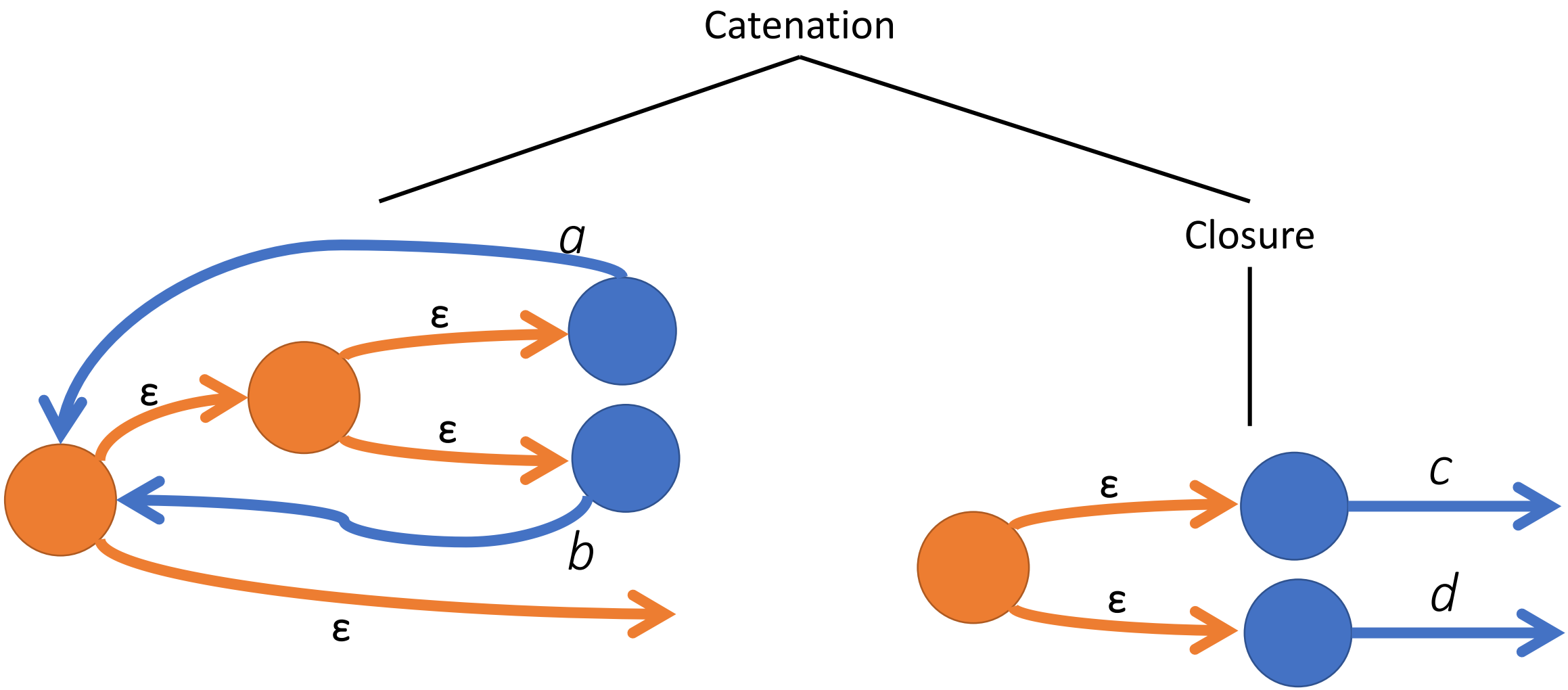




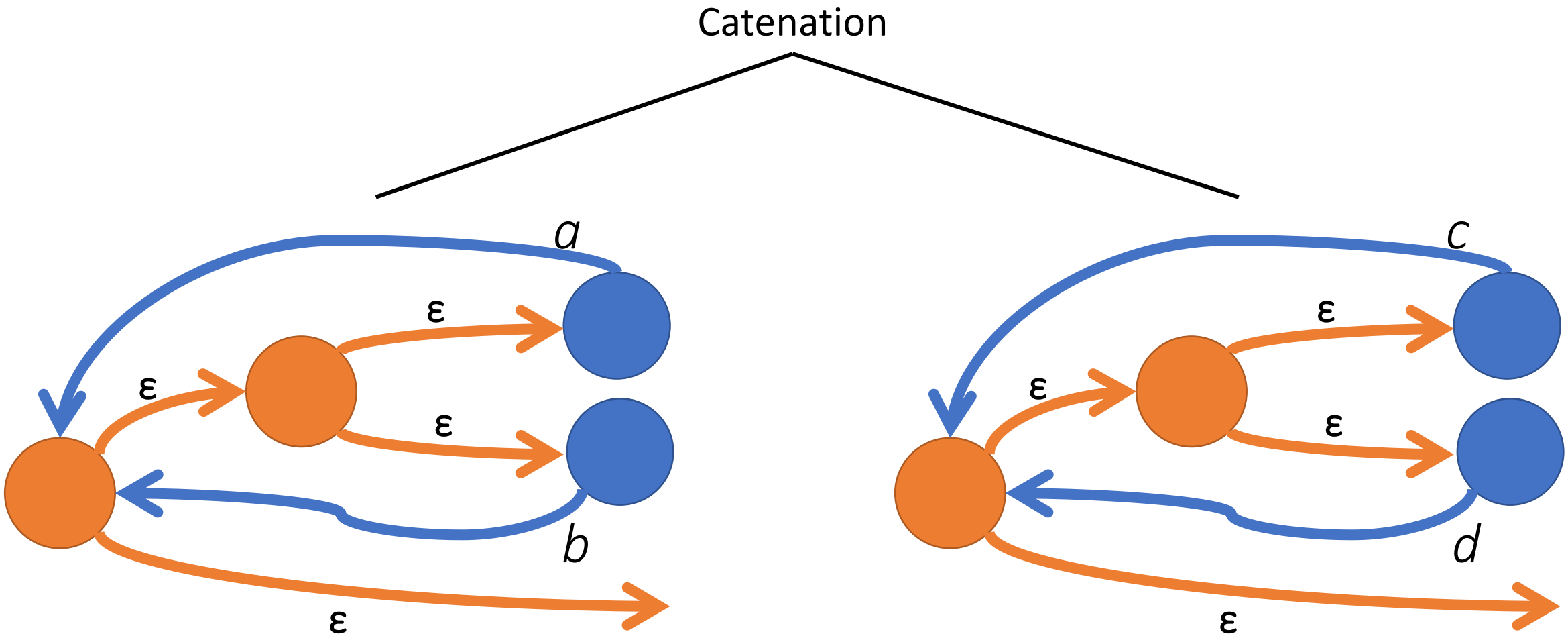
Produce the AST for **(a|b)*(c|d)***



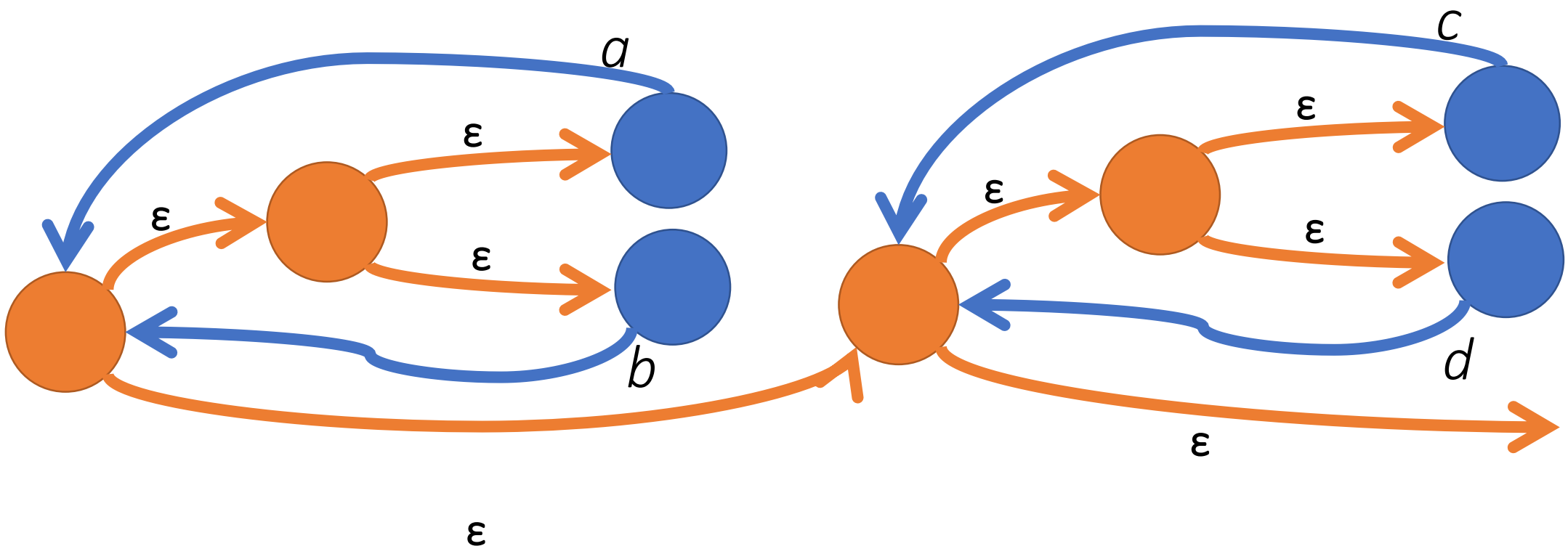
Produce the AST for **(a|b)*(c|d)***



Produce the AST for **(a|b)*(c|d)***



Produce the AST for **(a|b)*(c|d)***



Produce the AST for **(a|b)*(c|d)***