## Эквивалентность состояний конечных детерминированных распознавателей

Цепочки, позволяющие связать состояние $s_i$ с каким либо допускающим, назовём цепочками, допускаемыми состоянием $s_i$.

Язык, образованный множеством цепочек, допускаемых состоянием $s_i$, обозначим через $L(s_i)$.

На множестве состояний можно определить отношение R:

$$R=\{(s_i, s_j) \mid L(s_i)=L(s_j)\}.$$

*Два состояния $s_i$ и $s_j$ эквивелентны*, если из них можно составить пару, принадлежащую отношению R (если равны допускаемые ими языки $L(s_i)=L(s_j)$).

Эквивалентными могут быть состояния одного распознавателя или состояния различных распознавателей.

Если *два состояния неэквивалентны*, то любая цепочка, под действием которой одно из них переходит в допускающее состояние, а другое — в отвергающее, называется *цепочкой, различающей эти два состояния*.

Два состояния эквивалентны тогда и только тогда, когда не существует различающей их цепочки.

Алгоритм проверки эквивалентности двух состояний основан на следующем факте:

состояния $s_i$ и $s_j$ эквивалентны тогда и только тогда, когда выполняются следующие два условия:

1) *условие подобия* — состояния $s_i$ и $s_j$ должны быть либо оба допускающими, либо оба отвергающими;

2) *условие преемственности* — для всех входных символов состояния $s_i$ и $s_j$ должны переходить в эквивалентные состояния.

В процессе выполнения алгоритма строится *дерево проверки эквивалентности состояний*, в вершинах которого записываются пары состояний (в корне — пара проверяемых на эквивалентность состояний), а дуги отмечаются входными символами. Вершины строящегося дерева определяются как внутренние, граничные, дублирующие, конечные или различающие.

*Граничная* — это вершина, являющаяся листом и ещё не обработанная алгоритмом. После обработки эта вершина станет либо внутренней, имеющей сыновей, либо дублирующей, либо конечной, либо различающей.

*Дублирующая* — это вершина, являющаяся листом и содержащая такую пару состояний, которая есть хотя бы в одной из внутренних вершин.

*Конечная* — это вершина, которая содержит два одинаковых состояния.

*Различающая* — это вершина, которая содержит одно допускающее и одно отвергающее состояние.

*Алгоритм проверки эквивалентности двух состояний*.

1. Построить вершину — корень дерева, записать в неё пару проверяемых состояний. Если в корне одно состояние допускающее, а другое отвергающее, то проверяемые состояния неэквивалентны (нарушено условие подобия), конец алгоритма. В противном случае корень считать граничной вершиной и выполнить п.2.

2. Пока есть граничные вершины, обрабатывать их в порядке построения следующим образом.

Для каждого входного символа создать вершину, в которую провести дугу, отмеченную этим символом, из обрабатываемой вершины; в созданную вершину записать состояния переходов из состояний, записанных в обрабатываемой вершине, под действием входного символа, отмечающего входящую дугу.

Если в созданной вершине одно состояние допускающее, а другое отвергающее, то сделать эту вершину различающей; проверяемые состояния неэквивалентны, конец алгоритма.

Если в созданной вершине содержится такая пара состояний, которая есть хотя бы в одной из внутренних вершин, то сделать эту вершину дублирующей.

Если в созданной вершине содержится пара одинаковых состояний, то сделать эту вершину конечной.

В остальных случаях созданную вершину считать граничной.

Обработанные граничные вершины считать внутренними.

3. Если в множестве вершин дерева нет различающей, то проверяемые состояния являются эквивалентными, конец алгоритма.

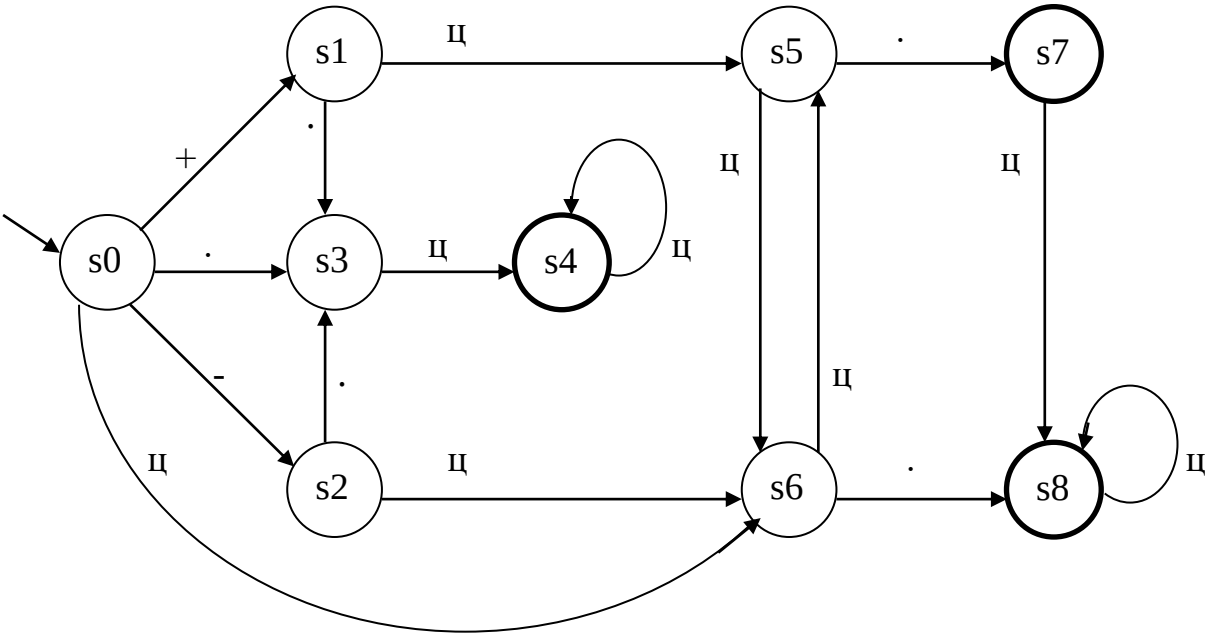
Если в результате выполнения алгоритма построено дерево, в котором нет различающей вершины, то все пары состояний, записанные в вершинах дерева, являются парами эквивалентных состояний.

Если же в результате выполнения алгоритма построено дерево, в котором есть различающая вершина, то неэквивалентными будут все пары состояний, записанные в вершинах на пути от корня к различающей вершине (для них не выполняется условие преемственности). Цепочка, соответствующая пути от корня к различающей вершине, представляет собой цепочку, различающую проверяемые на эквивалентность состояния, причём это будет кратчайшая из различающих состояния цепочек, т.к. дерево проверки строится в ширину.

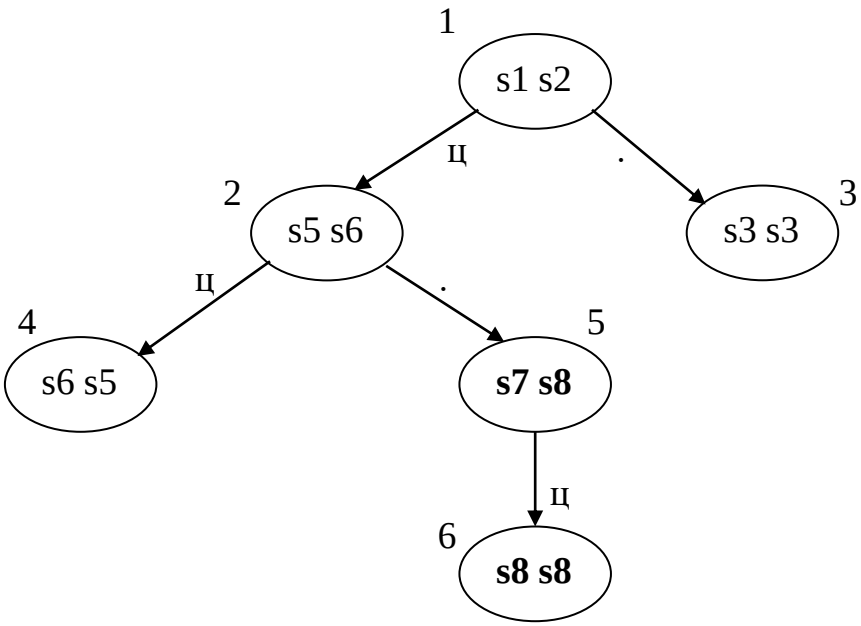
Если в множестве состояний распознавателя есть единственное состояние ошибки $s_{ош} \in S$, которое не допускает ни одной цепочки ($L(s_{ош})=\varnothing$), а любое другое состояние $s_i \in S$, допускающее или отвергающее, допускает хотя бы одну цепочку ($L(s_i)\neq\varnothing$), то состояния $s_{ош}$ и $s_i$ неэквивалентны, и в этом случае вершину, содержащую пару состояний $s_{ош}$ и $s_i$ , будем считать различающей. Если $\alpha$ — цепочка, соответствующая пути от корня к различающей вершине, содержащей пару состояний $s_{ош}$ и $s_i$ , то цепочка $\alpha\beta$ недопускается одним из проверяемых на эквивалентность состояний не при каких $\beta$ и допускается другим состоянием при некоторых $\beta$. При построении дерева проверки эквивалентности состояний вершины, содержащие два состояния ошибки, и дуги, ведущие в эти вершины, не будем включать в дерево.

Конечные вершины, содержащие два состояния ошибки, в дерево включать не будем.

**Пример**.
Граф конечного детерминированного распознавателя.

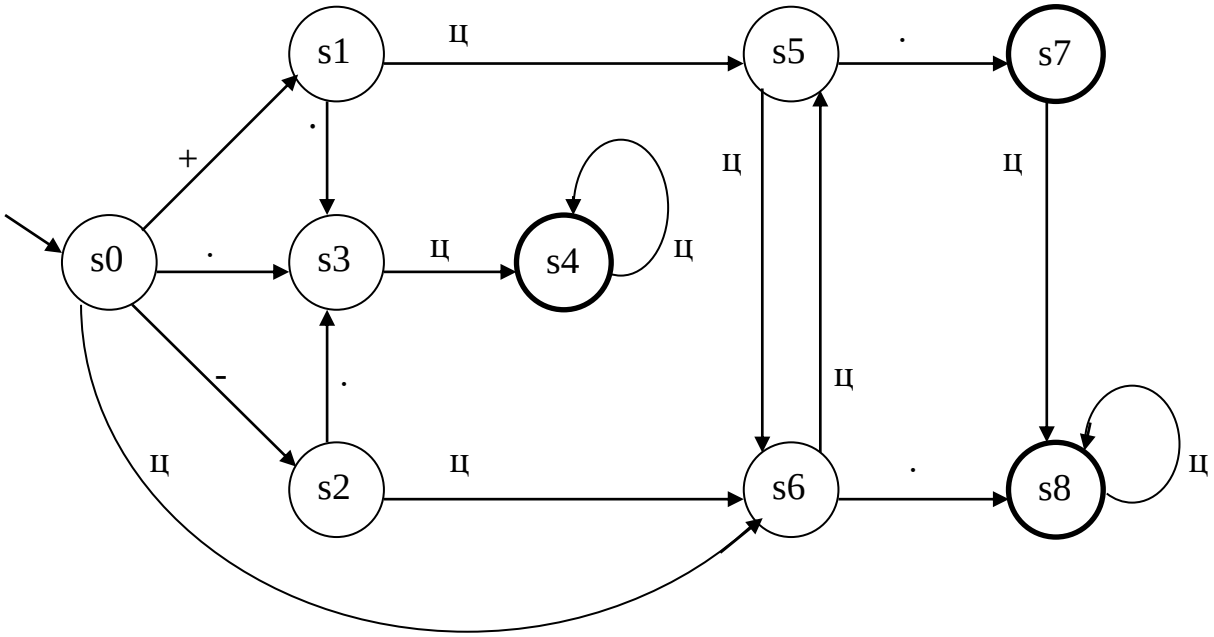



Дерево проверки эквивалентности состояний s1 и s2 .

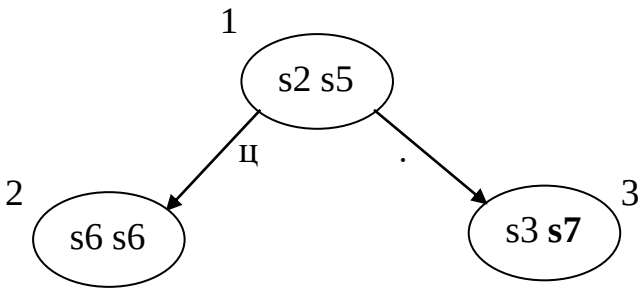



Вершины пронумерованы в порядке их построения. Вершина 4 – дублирующая, т.к. содержит такую же пару состояний, что и вершина 2 (порядок состояний не важен), вершины 3 и 6 – конечные, остальные – внутренние, следовательно состояния s1 и s2 – эквивалентные. Пары состояний, записанные в вершинах 2 и 5 – эквивалентные.

Граф конечного детерминированного распознавателя.




Дерево проверки эквивалентности состояний s2 и s5 .

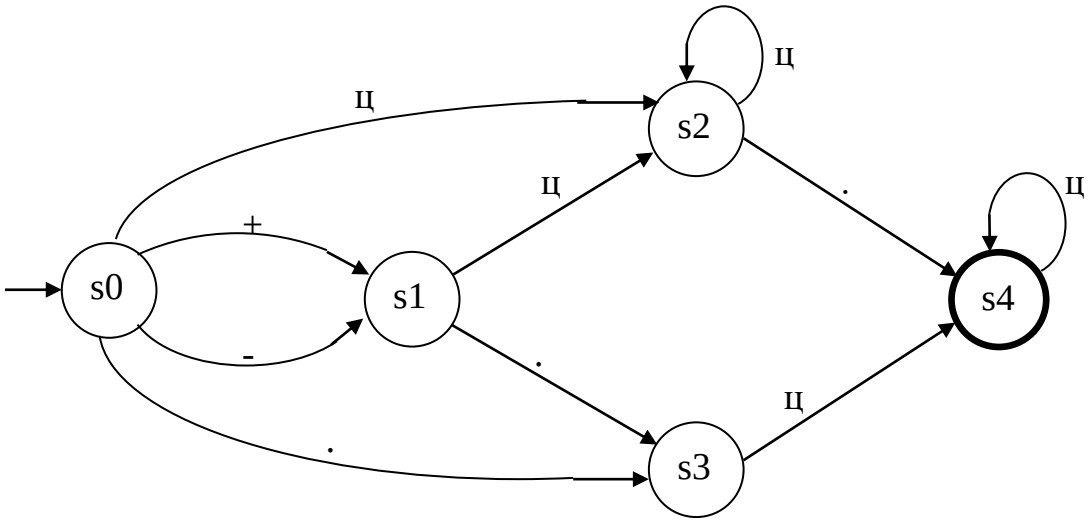



В вершине 3 состояние s3 – отвергающее, а s7 – допускающее, следовательно вершина 3 – различающая, состояния s2 и s5 – неэквивалентны, цепочка, состоящая из одного символа “точка” , является различающей для состояний s2 и s5.

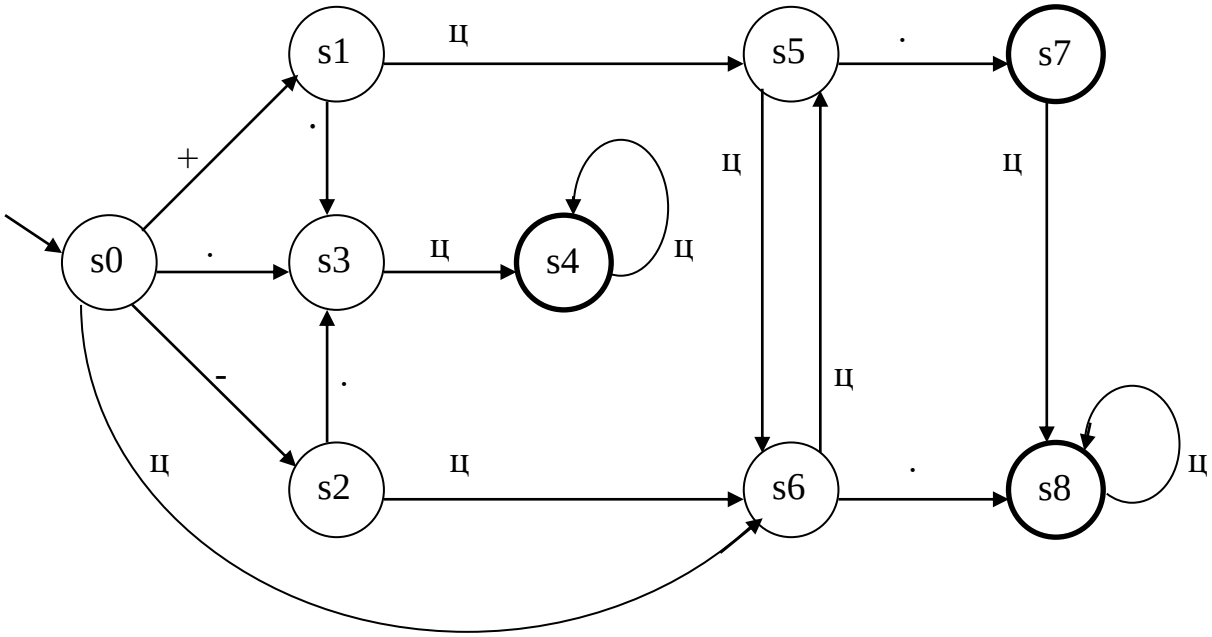
## Эквивалентность конечных детерминированных распознавателей

Конечные распознаватели A и A’ эквивалентны, если равны допускаемые ими языки L(A)=L(A’). Множество цепочек, допускаемых состоянием $s_0$ , образует язык L($s_0$). Естественно, что L(A)=L($s_0$), если $s_0$ — начальное состояние распознавателя A и L(A’)=L($s_0$’), если $s_0$’ — начальное состояние распознавателя A’. Следовательно, для проверки истинности равенства L(A)=L(A’), т.е. проверки эквивалентности распознавателей A и A’, достаточно проверить истинность равенства L($s_0$)=L($s_0$’), т.е. проверить эквивалентность начальных состояний распознавателей A и A’. Таким образом, проверка эквивалентности конечных распознавателей сводится к проверки эквивалентности состояний. При выполнении алгоритма проверки эквивалентности состояний различных распознавателей нужно учитывать то, что в дереве проверки не может быть конечных вершин — вершин, содержащих одинаковые состояния.

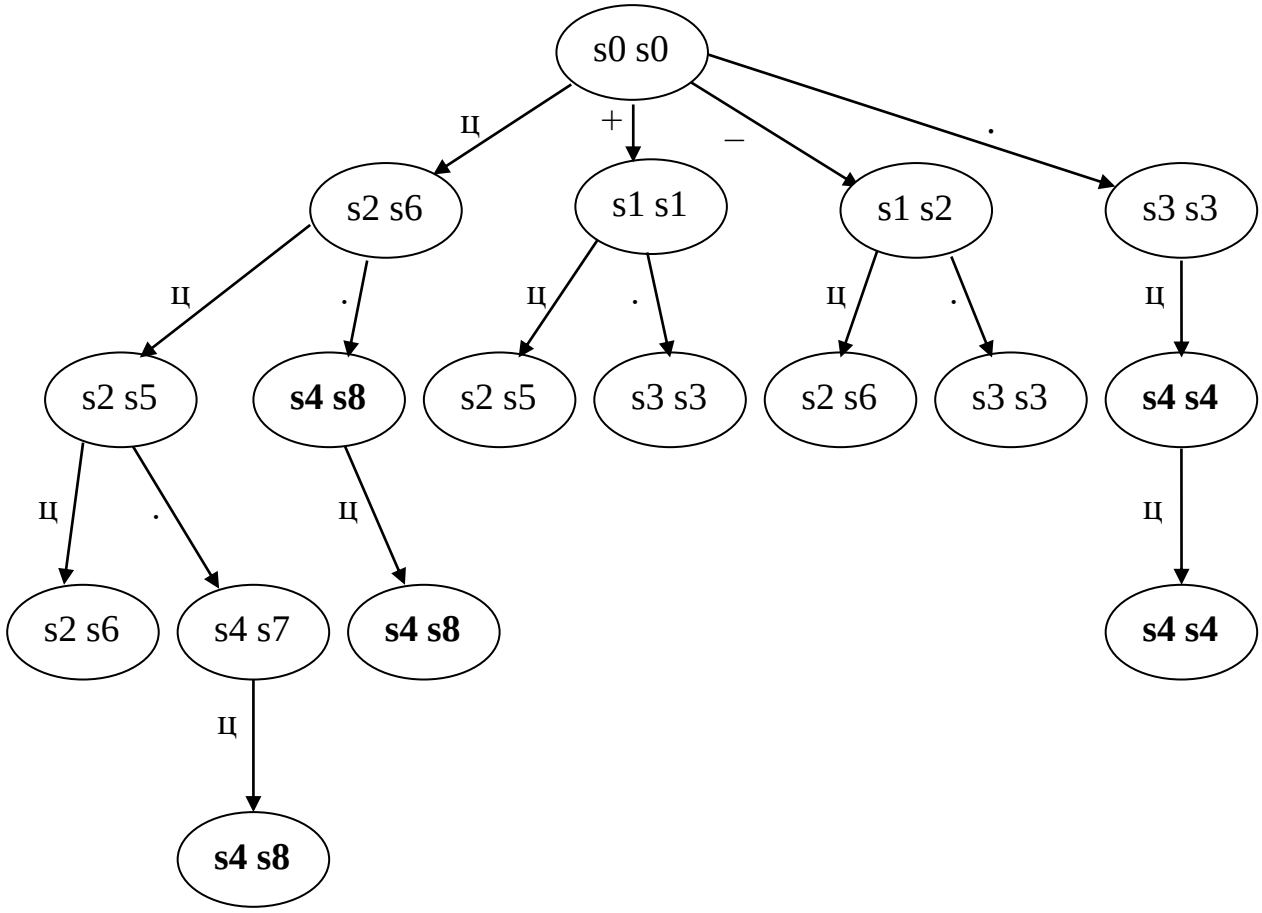
**Пример**.
Граф конечного распознавателя А.




Граф конечного детерминированного распознавателя А’.

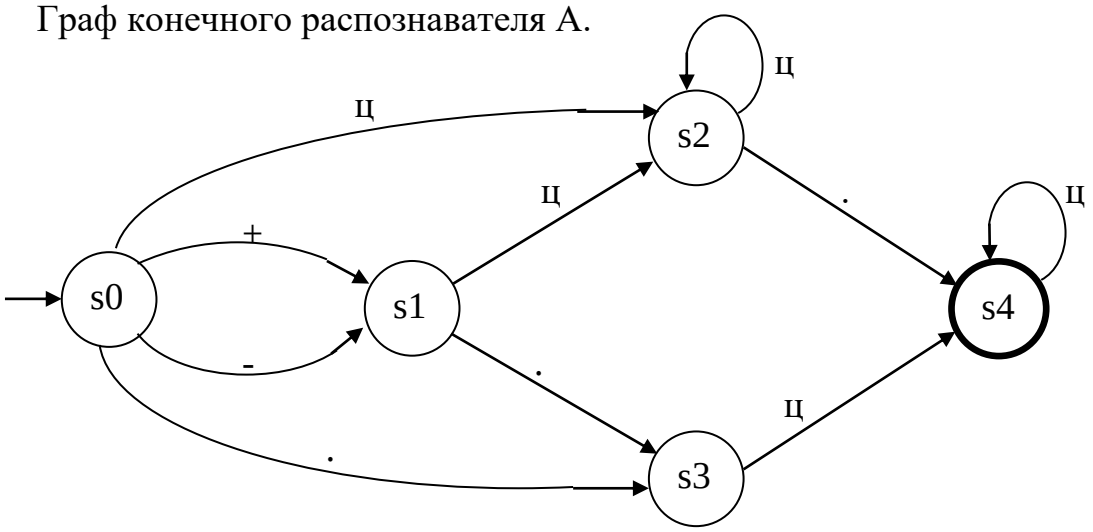


Дерево проверки эквивалентности распознавателей A и A’.

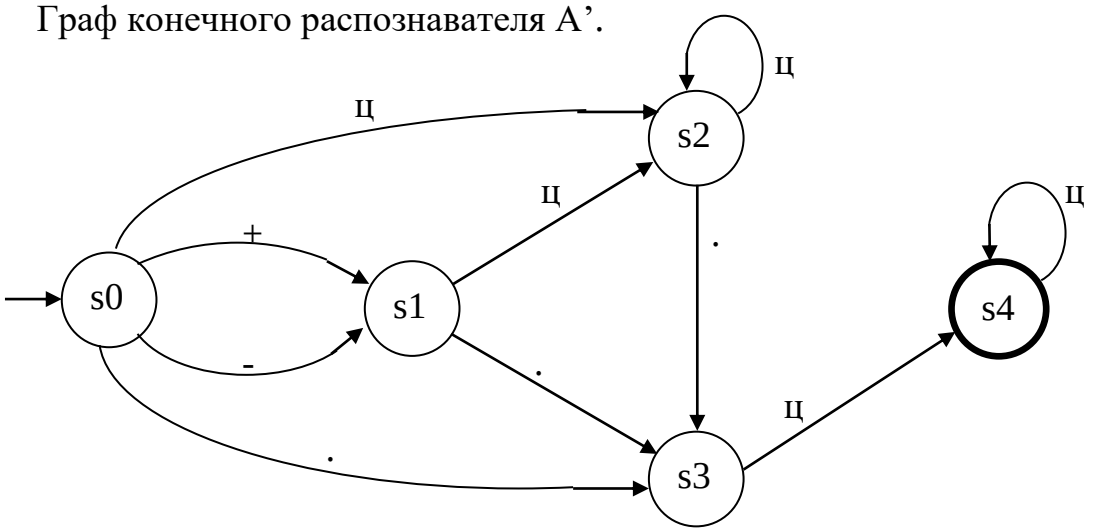



В дереве проверки эквивалентности нет различающих вершин, следовательно эквивалентны начальные состояния проверяемых на эквивалентность распознавателей A и A’ и эквивалентны распознаватели A и A’.

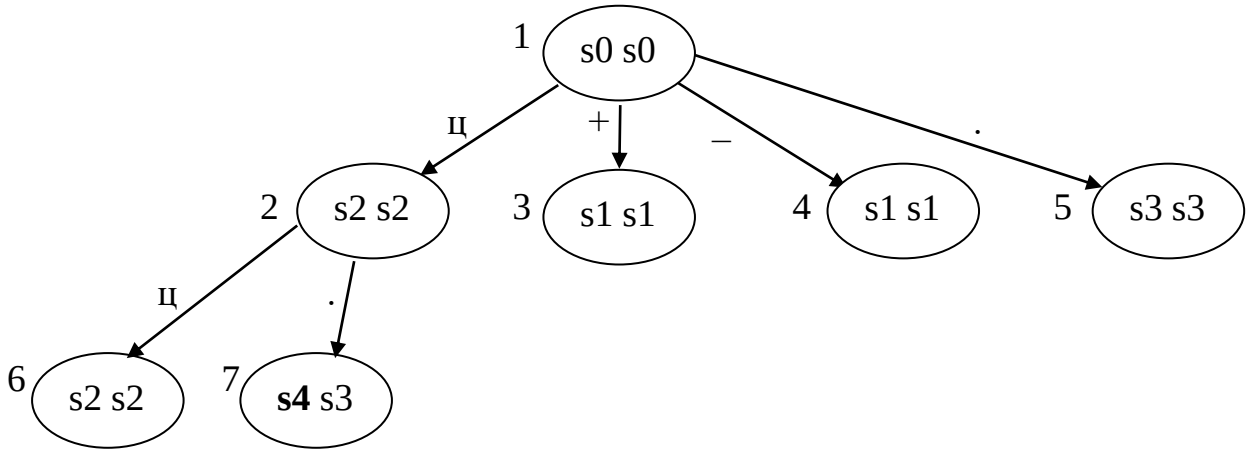
Граф конечного распознавателя А.




Граф конечного распознавателя А’.




Дерево проверки эквивалентности распознавателей А и А’.




В вершине 7 состояние s4 распознавателя А –допускающее, а состояние s3 распознавателя А’ – отвергающее, следовательно распознаватели А и А’ – неэквивалентны, цепочка ц. является различающей, она допускается распознавателем А и отвергается распознавателем А’.