Mini models

EECS 4315

www.eecs.yorku.ca/course/4315/

Mini model

Question

What do the model and the mini model have in common?

Mini model

Question

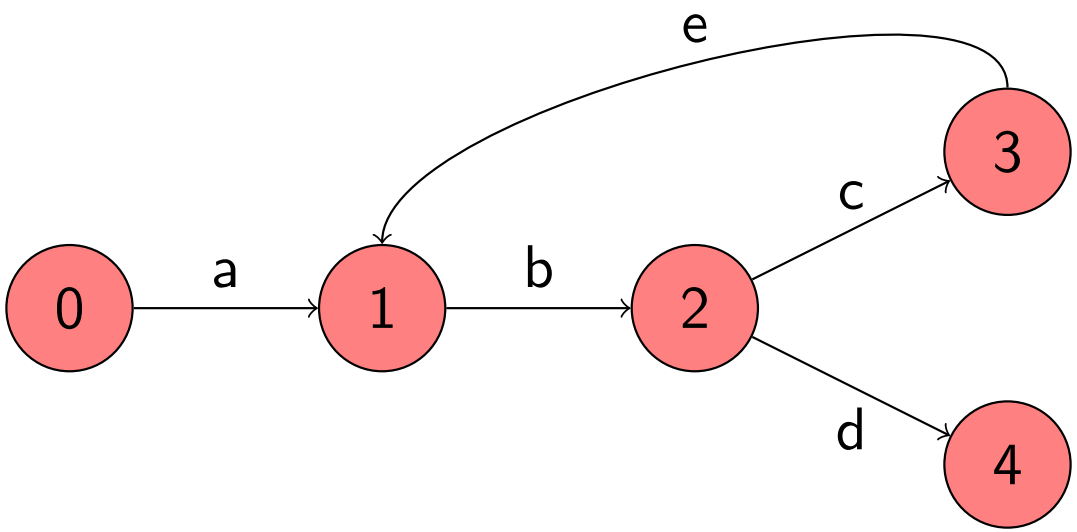
What do the model and the mini model have in common?

Answer

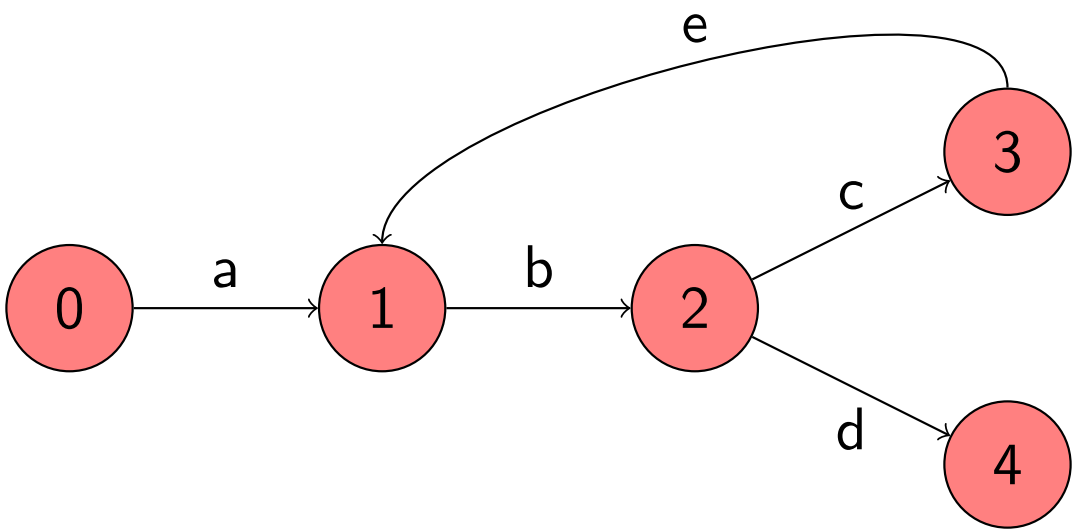
- The initial state.
- The final states.
- The branching structure.
- The language: (finite and infinite) sequences of actions.[a]

[a] Similar to the language accepted by a finite automaton, as discussed in EECS 2001 Introduction to Theory of Computation.

Model



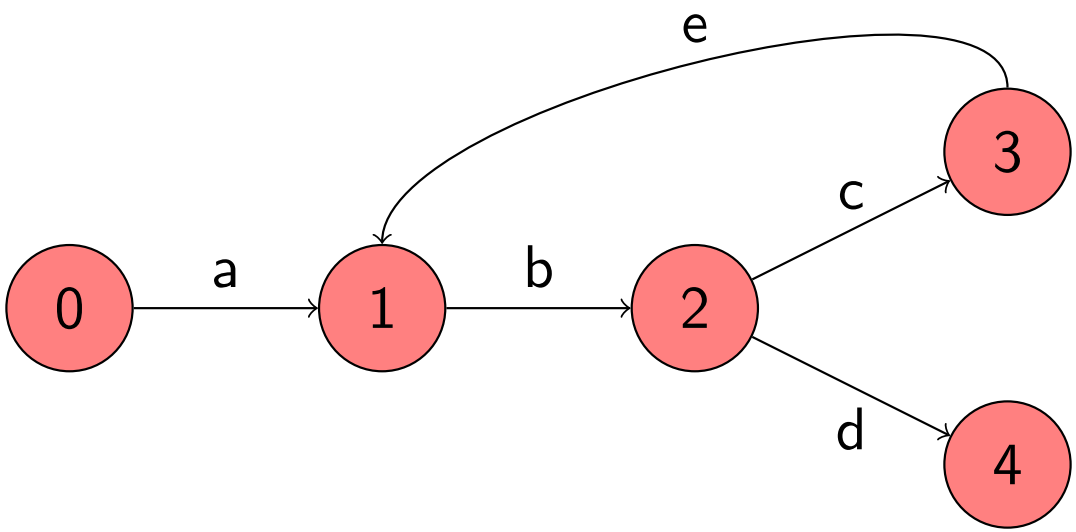
Model



Question

Which is the initial state?

Model



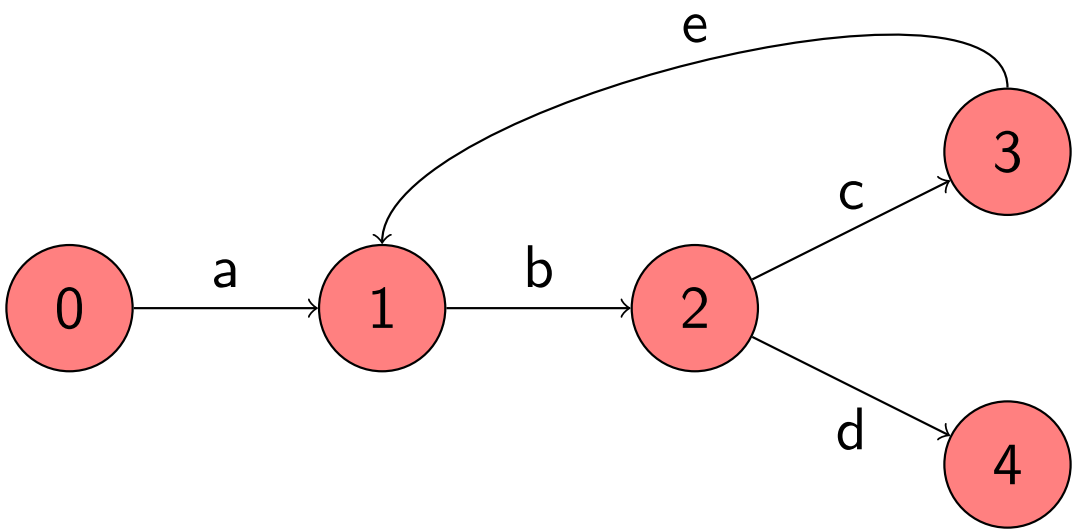
Question

Which is the initial state?

Answer

State 0.

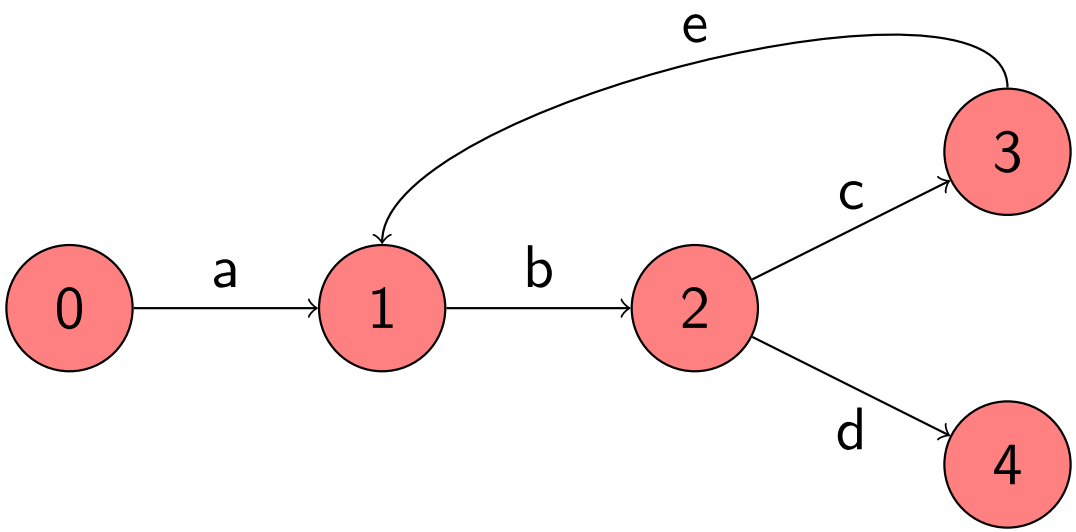
Model



Question

Which are the final states?

Model



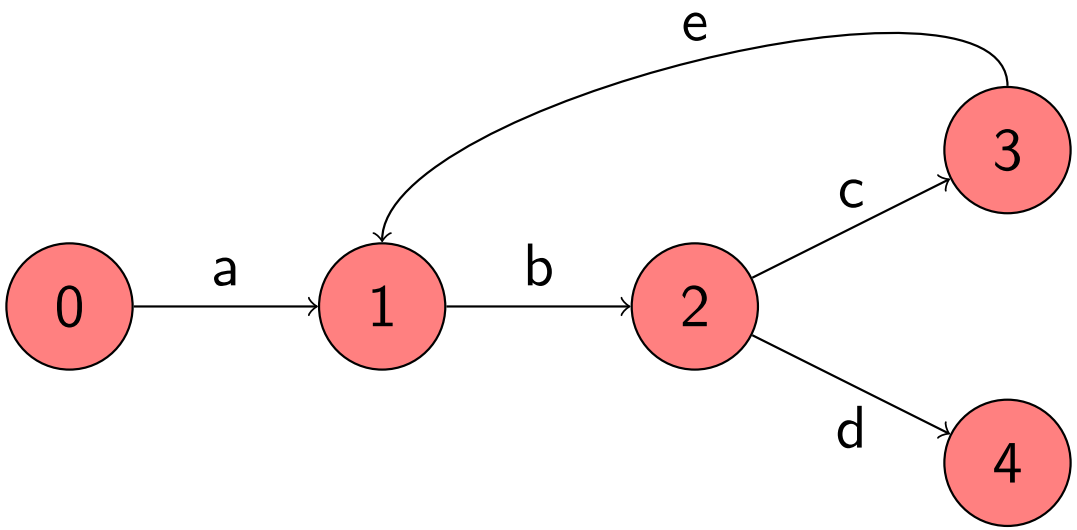
Question

Which are the final states?

Answer

State 4.

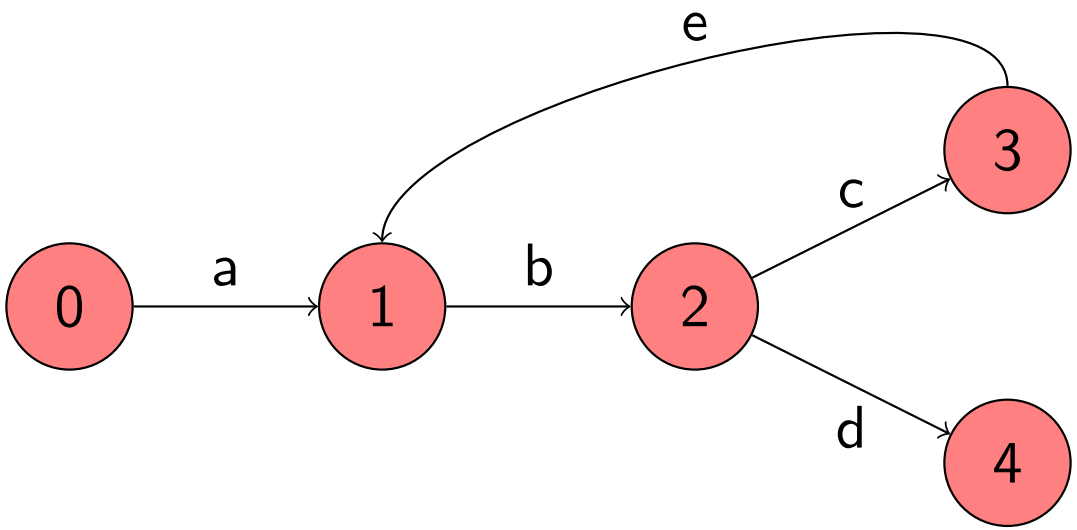
Model



Question

Which are the branching states?

Model



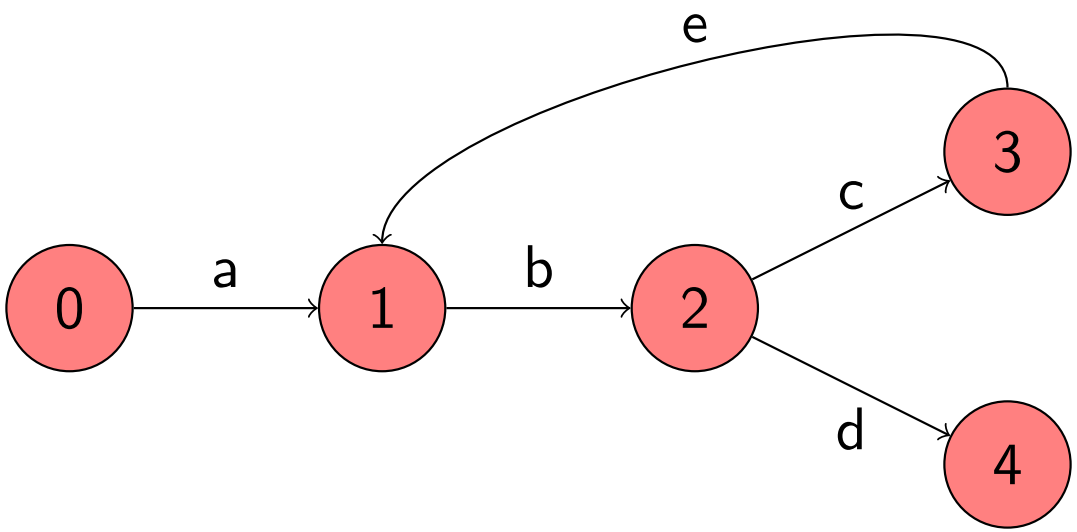
Question

Which are the branching states?

Answer

State 2.

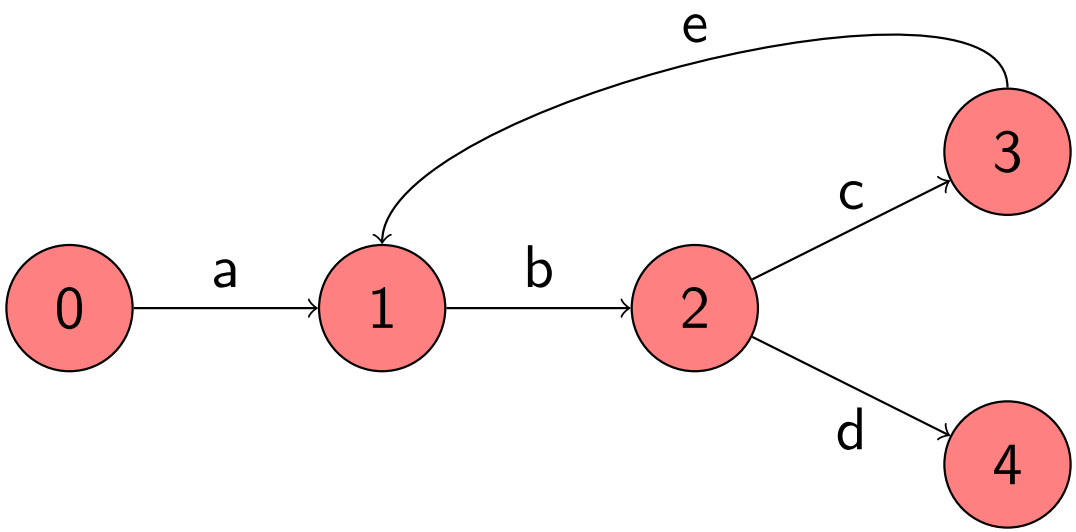
Model



Question

What is the language?

Model



Question

What is the language?

Answer

$\{abd, abcebd, abcebcebd, \ldots, abcebcebce \ldots\}$.

Mini model

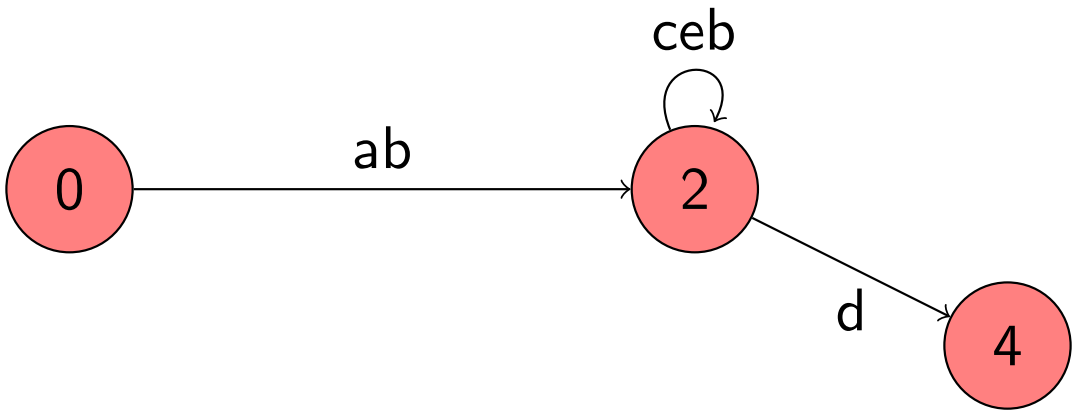
Question

What is the corresponding mini model?

Mini model

Question

What is the corresponding mini model?



Model

Definition

A labelled transition system is a tuple $\langle S, A, \rightarrow, s \rangle$ consisting of

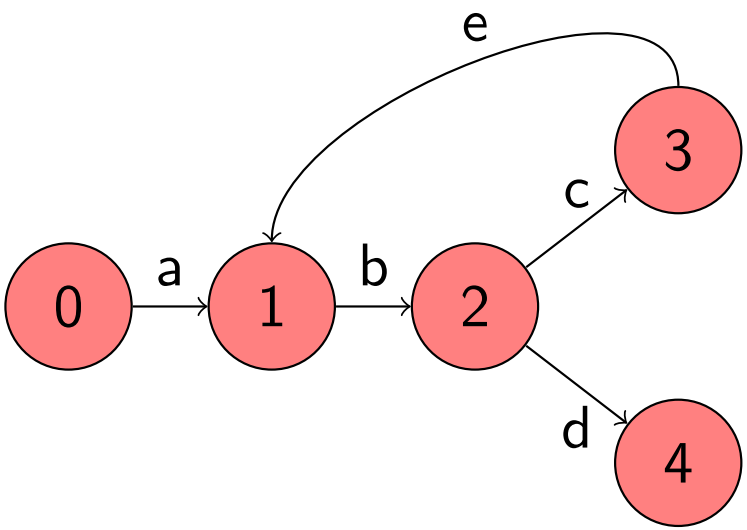
- a set S of states,
- a set A of actions,
- a transition relation $\rightarrow \subseteq S \times A \times S$, and
- a start state $s \in S$.

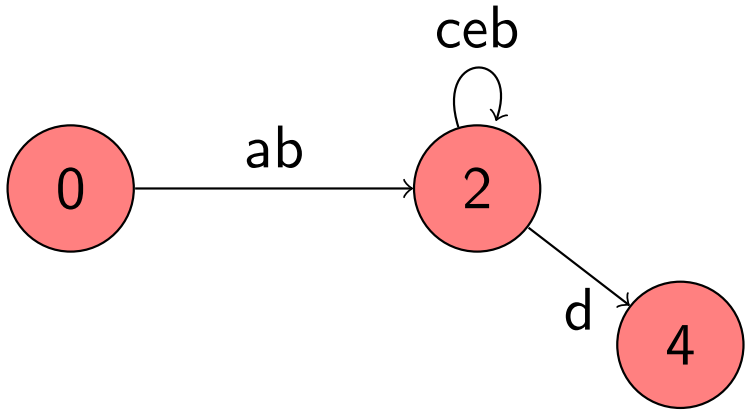
From model to mini model

Problem

Given a model, expressed as a labelled transition system, construct the corresponding mini model, also expressed as a labelled transition system.

States

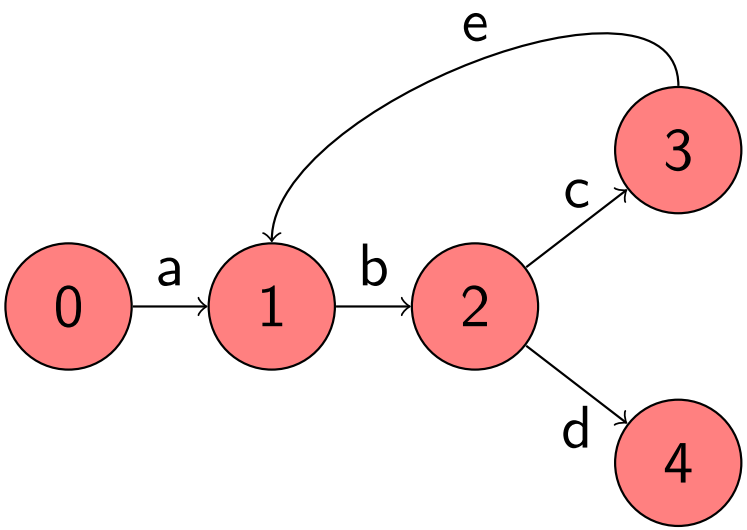


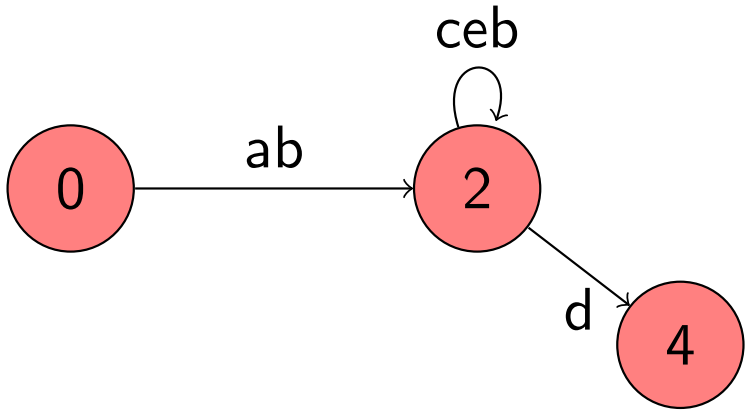


Question

Which states do we keep?

States





Question

Which states do we keep?

Answer

Initial state, final states, and all branching states.

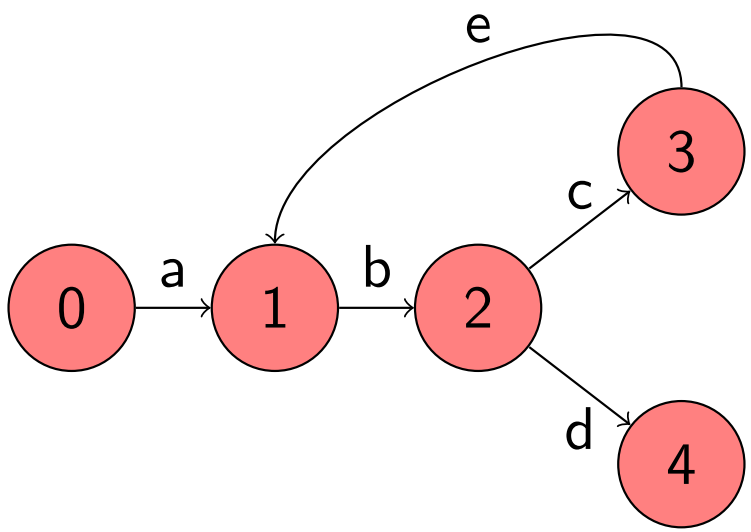
Labelled transition systems: successors

Definition

The set $succ(s)$ of successors of the state s is defined by

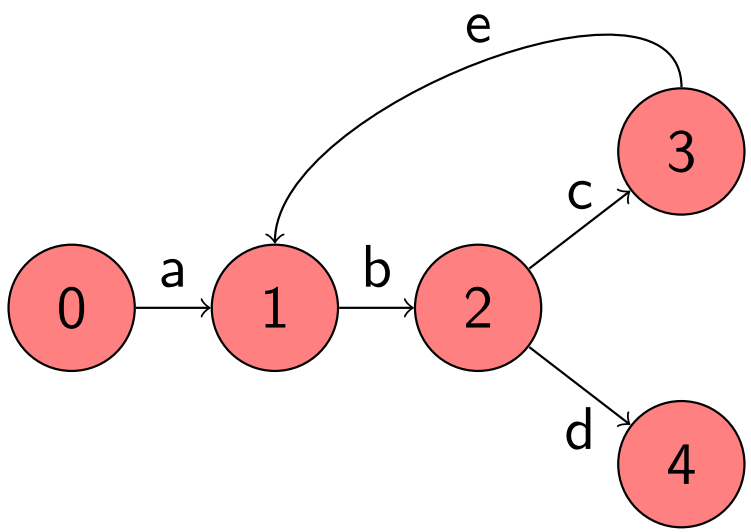
$$succ(s) = \{\, t \in S \mid \exists a \in A : s \xrightarrow{a} t \,\}.$$

Labelled transition systems: successors



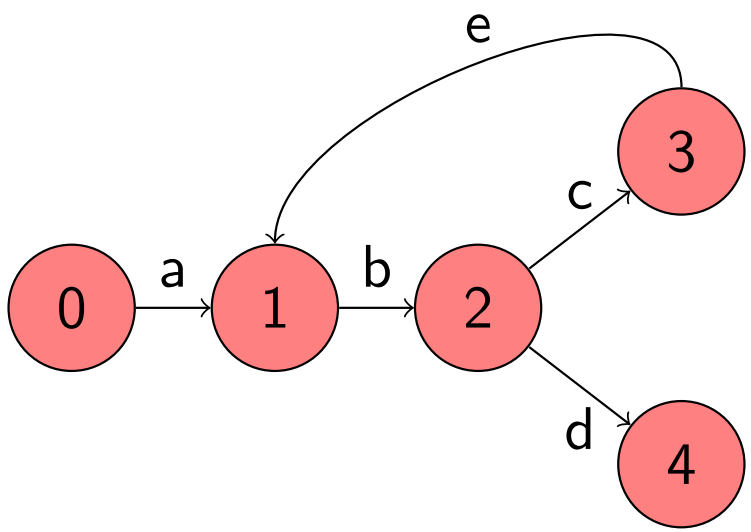
$succ(0) \quad =$

Labelled transition systems: successors



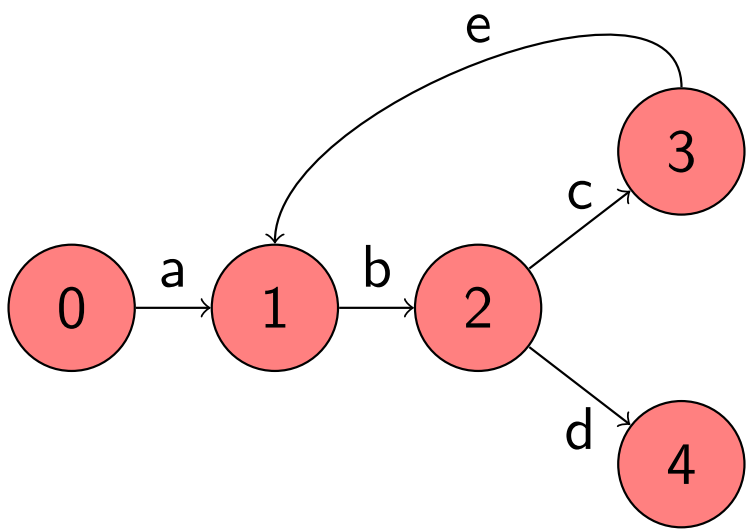
$$succ(0) = \{1\}$$
$$succ(1) =$$

Labelled transition systems: successors



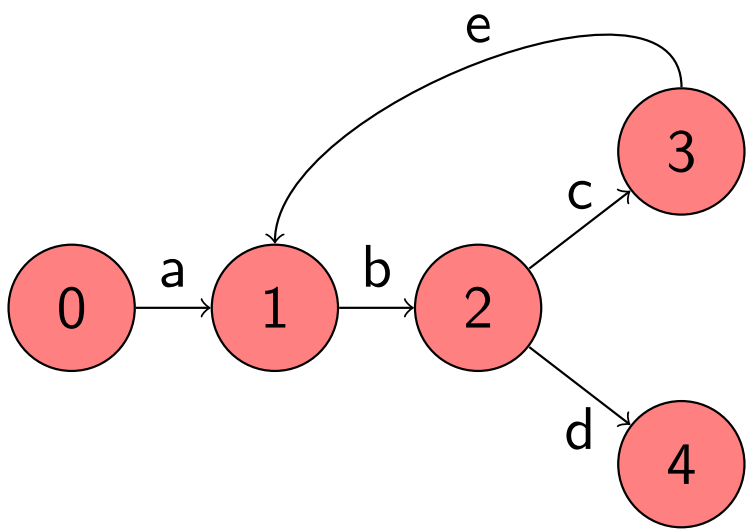
$$
\begin{aligned}
succ(0) &= \{1\} \\
succ(1) &= \{2\} \\
succ(2) &=
\end{aligned}
$$

Labelled transition systems: successors



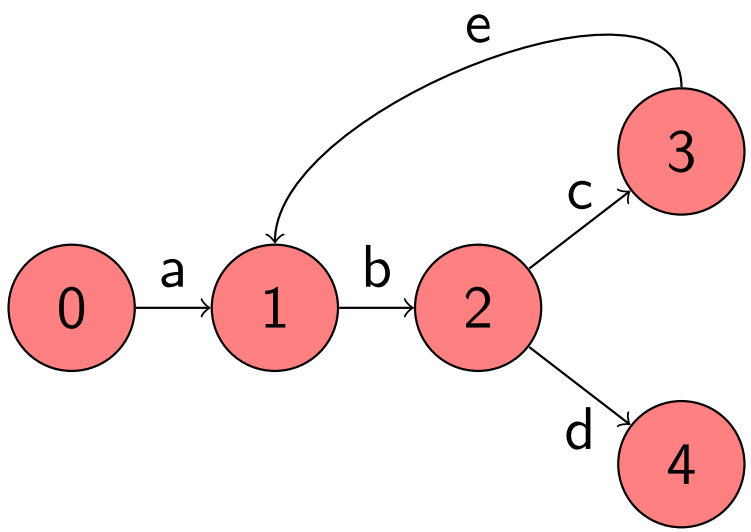
$$
\begin{aligned}
succ(0) &= \{1\} \\
succ(1) &= \{2\} \\
succ(2) &= \{3, 4\} \\
succ(3) &=
\end{aligned}
$$

Labelled transition systems: successors



$$
\begin{aligned}
succ(0) &= \{1\} \\
succ(1) &= \{2\} \\
succ(2) &= \{3, 4\} \\
succ(3) &= \{1\} \\
succ(4) &=
\end{aligned}
$$

Labelled transition systems: successors



$$
\begin{aligned}
succ(0) &= \{1\} \\
succ(1) &= \{2\} \\
succ(2) &= \{3, 4\} \\
succ(3) &= \{1\} \\
succ(4) &= \emptyset
\end{aligned}
$$

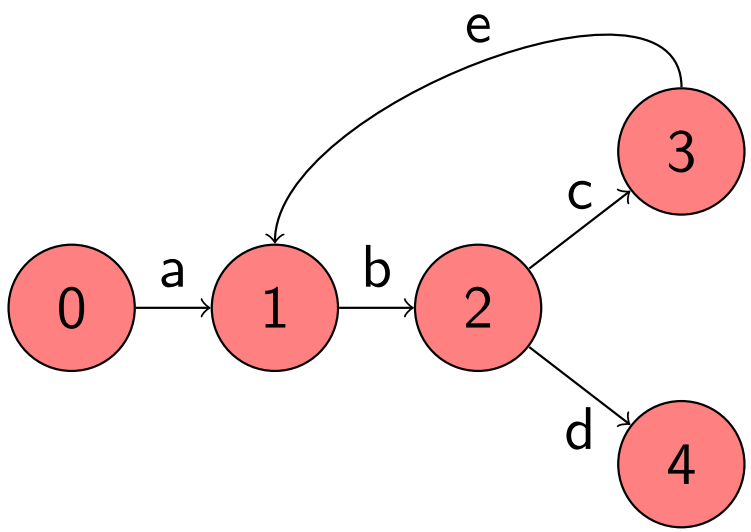
Labelled transition systems: predecessors

Definition

The set $pred(s)$ of predecessors of the state s is defined by

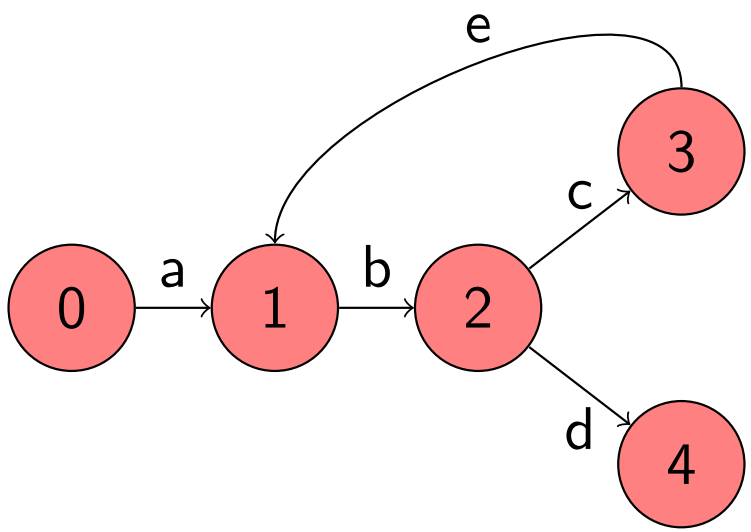
$$pred(s) = \{\, t \in S \mid \exists a \in A : t \xrightarrow{a} s \,\}.$$

Labelled transition systems: predecessors



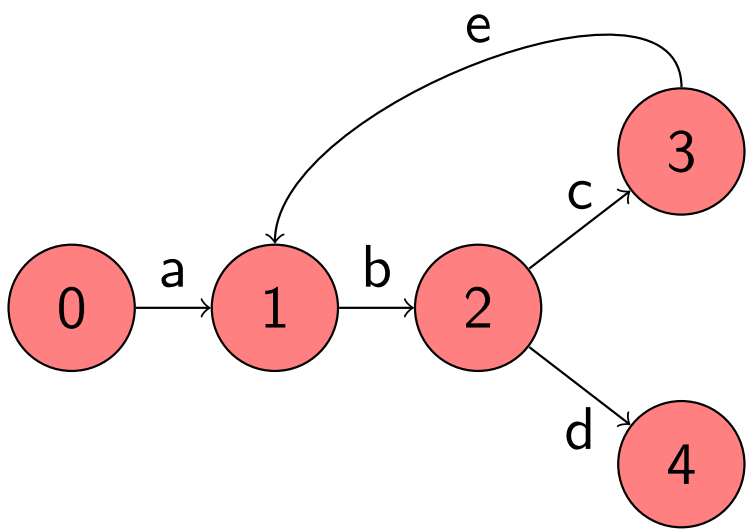
$$pred(0) \quad = \quad$$

Labelled transition systems: predecessors



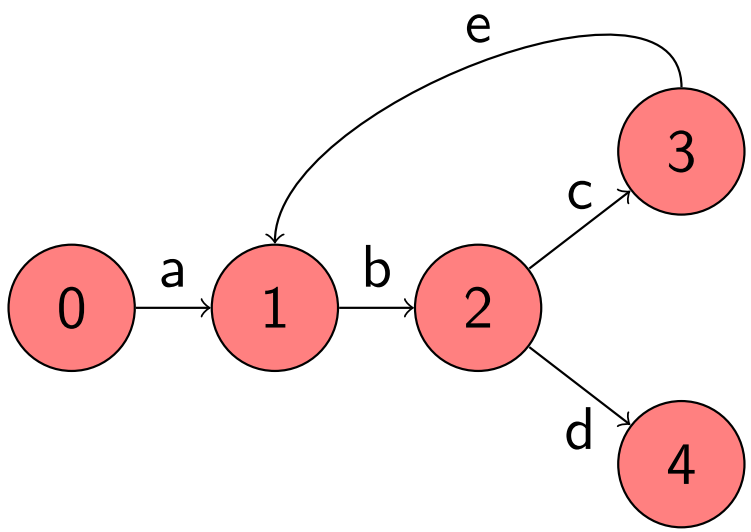
$$\begin{aligned} \mathit{pred}(0) &= \emptyset \\ \mathit{pred}(1) &= \end{aligned}$$

Labelled transition systems: predecessors



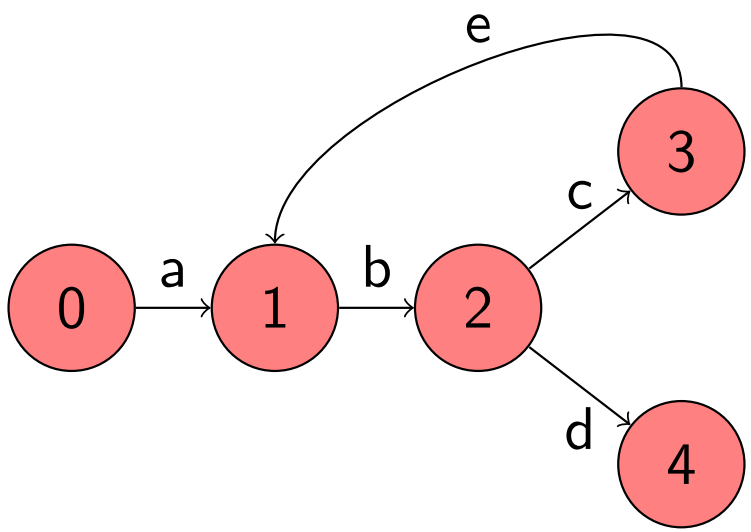
$$
\begin{aligned}
\mathit{pred}(0) &= \emptyset \\
\mathit{pred}(1) &= \{0, 3\} \\
\mathit{pred}(2) &=
\end{aligned}
$$

Labelled transition systems: predecessors



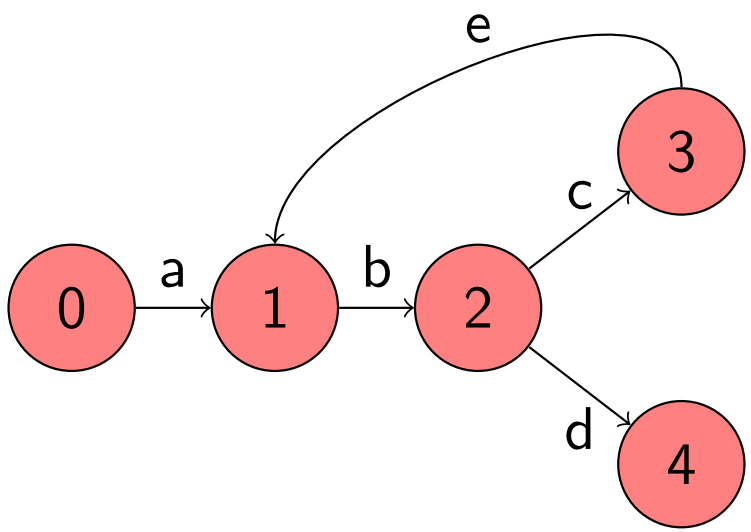
$$
\begin{aligned}
\mathit{pred}(0) &= \emptyset \\
\mathit{pred}(1) &= \{0, 3\} \\
\mathit{pred}(2) &= \{1\} \\
\mathit{pred}(3) &=
\end{aligned}
$$

Labelled transition systems: predecessors



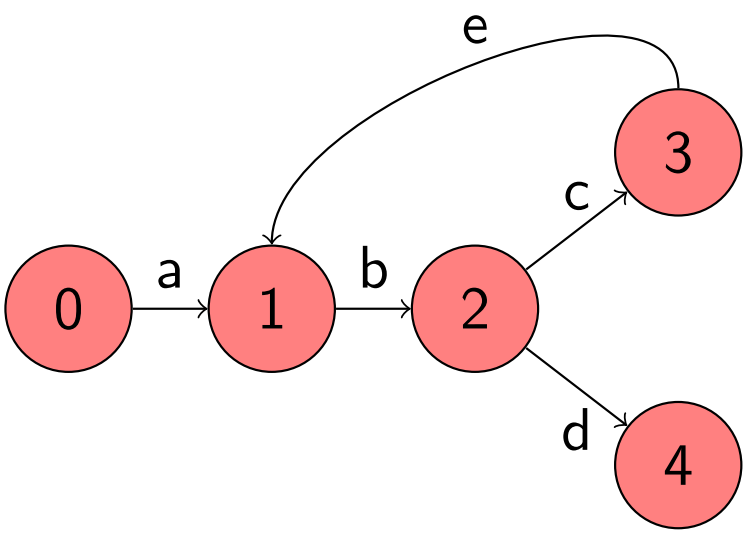
$$
\begin{aligned}
\mathit{pred}(0) &= \emptyset \\
\mathit{pred}(1) &= \{0, 3\} \\
\mathit{pred}(2) &= \{1\} \\
\mathit{pred}(3) &= \{2\} \\
\mathit{pred}(4) &=
\end{aligned}
$$

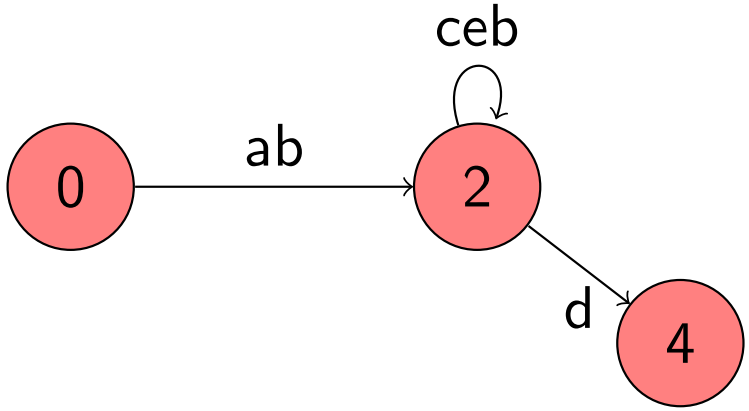
Labelled transition systems: predecessors



$$
\begin{aligned}
\mathit{pred}(0) &= \emptyset \\
\mathit{pred}(1) &= \{0, 3\} \\
\mathit{pred}(2) &= \{1\} \\
\mathit{pred}(3) &= \{2\} \\
\mathit{pred}(4) &= \{2\}
\end{aligned}
$$

States

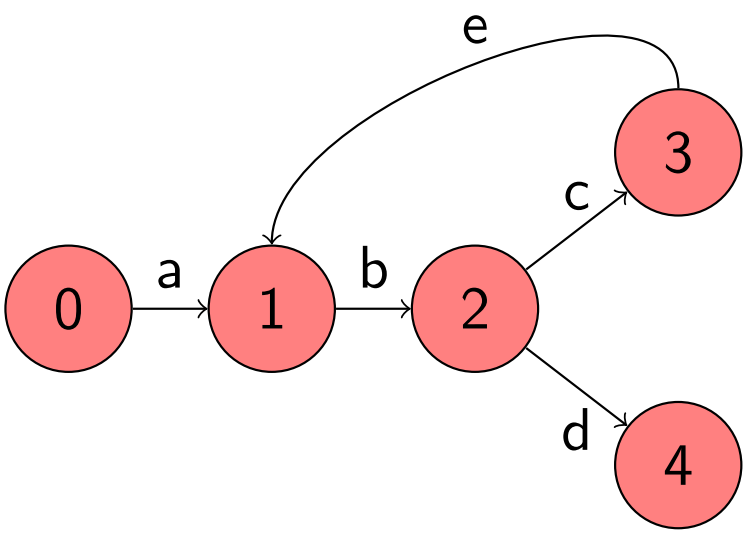


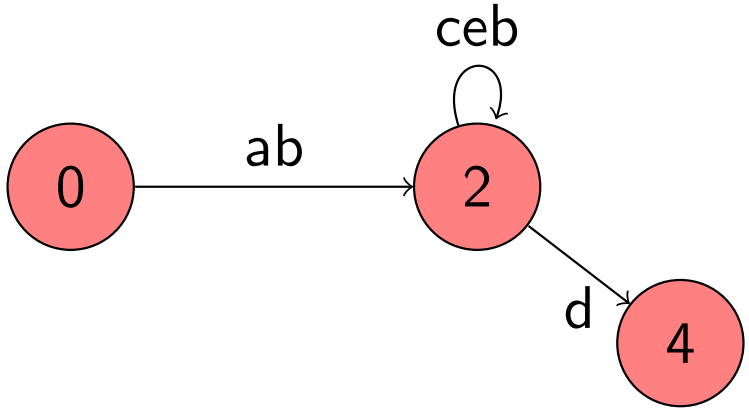


Question

Given a labelled transition system $\langle S, A, \rightarrow, s_0 \rangle$, which states do we keep?

States





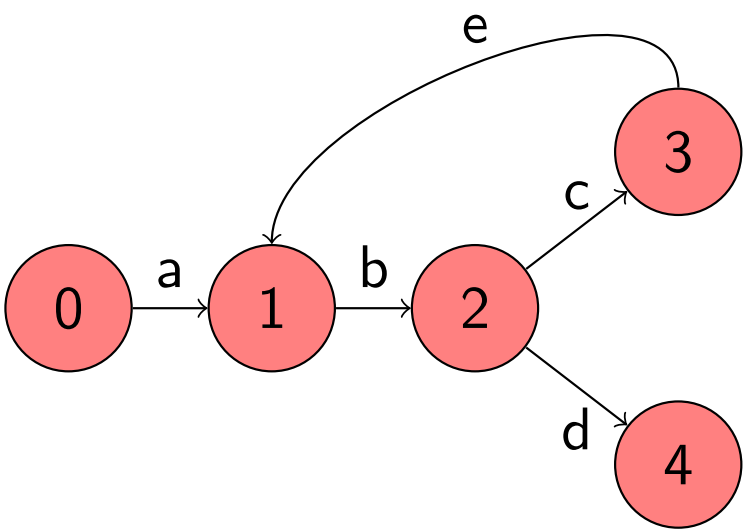
Question

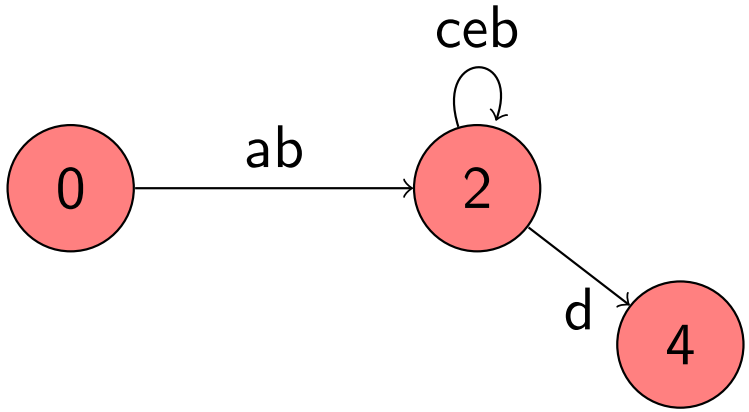
Given a labelled transition system $\langle S, A, \rightarrow, s_0 \rangle$, which states do we keep?

Answer

$S^+ = \{s_0\} \cup \{s \in S \mid |succ(s)| \neq 1\}$.

Actions

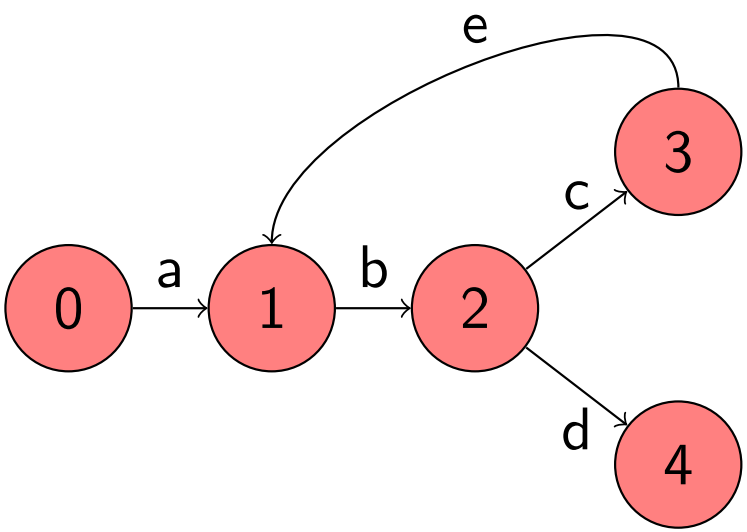


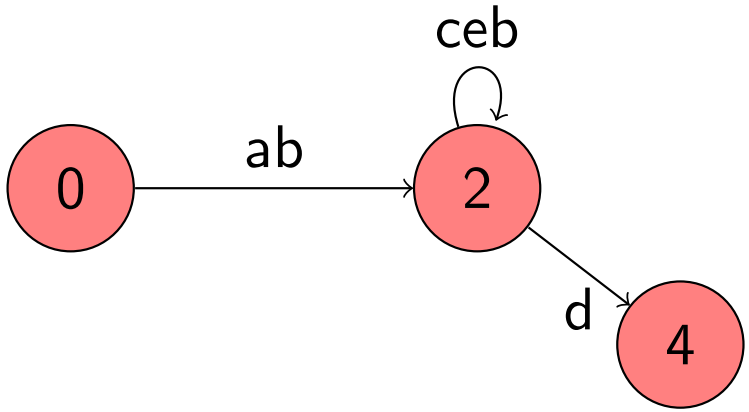


Question

Given a labelled transition system $\langle S, A, \rightarrow, s_0 \rangle$, what are the actions of the labelled transition system of the corresponding mini model?

Actions





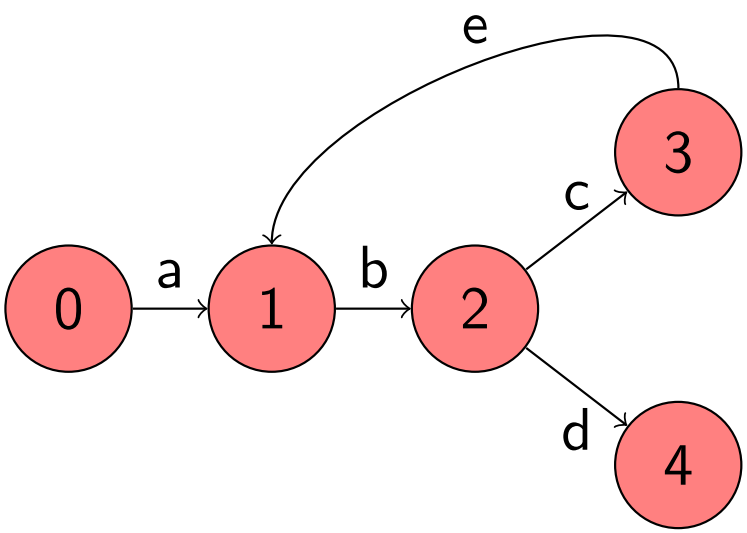
Question

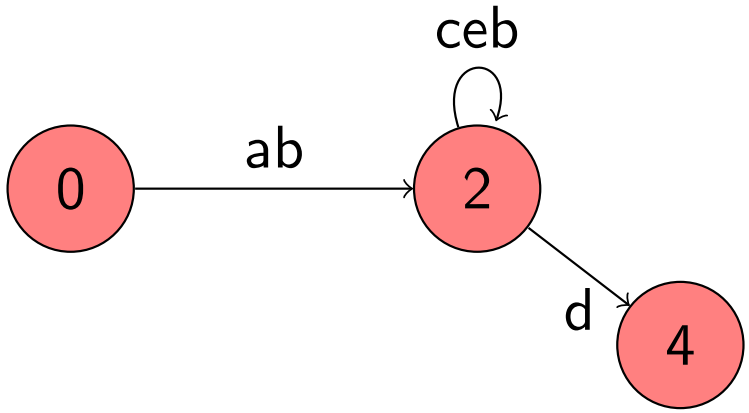
Given a labelled transition system $\langle S, A, \rightarrow, s_0 \rangle$, what are the actions of the labelled transition system of the corresponding mini model?

Answer

A^+: nonempty and finite sequences of actions.

Transitions

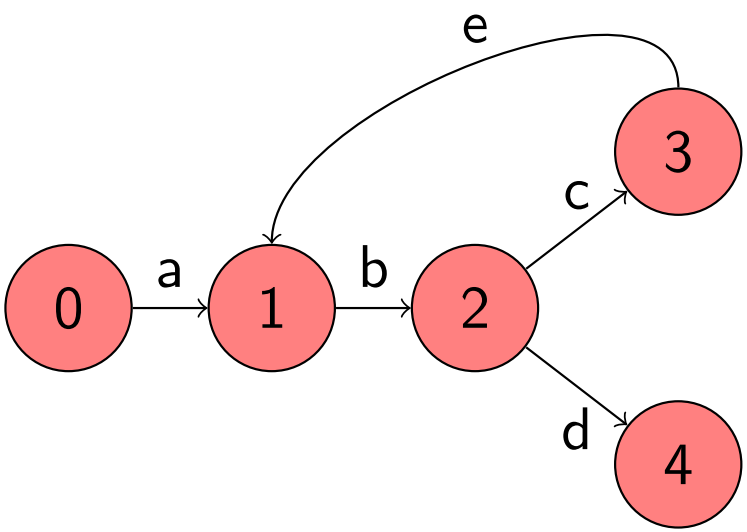


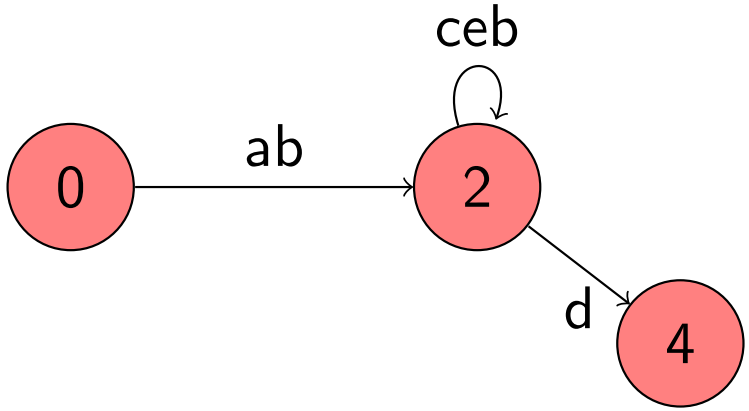


Question

Given a labelled transition system $\langle S, A, \rightarrow, s_0 \rangle$, what are the transitions of the labelled transition system of the corresponding mini model?

Transitions





Answer

$s_1 \xrightarrow{a_1 \ldots a_n}{}^{+} s_{n+1}$ if

$$s_1 \in S^+ \wedge s_{n+1} \in S^+ \wedge$$
$$\exists s_2, \ldots, s_n \in S \setminus S^+ : \forall 1 \leq i < n : s_i \xrightarrow{a_i} s_{i+1} \wedge$$
$$\forall 1 \leq i, j \leq n : s_i = s_j \Rightarrow i = j$$

From model to mini model

Problem

Given a model, expressed as a labelled transition system $\langle S, A, \rightarrow, s_0 \rangle$, construct the corresponding mini model, also expressed as a labelled transition system.

From model to mini model

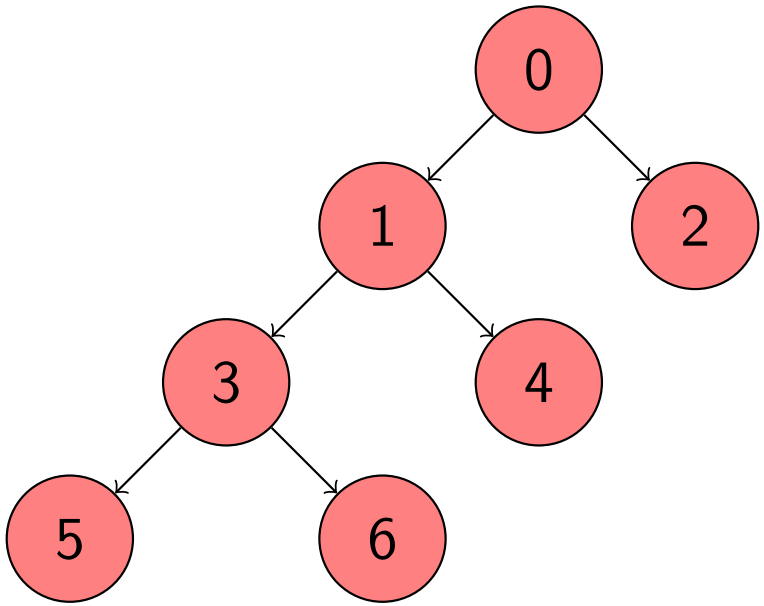
Problem

Given a model, expressed as a labelled transition system $\langle S, A, \rightarrow, s_0 \rangle$, construct the corresponding mini model, also expressed as a labelled transition system.

Solution

$\langle S^+, A^+, \rightarrow^+, s_0 \rangle$.

Search



Task 1

Develop a Java app that prints some output. When checking the Java app by JPF with depth-first search (DFS) the output should be different from the output for breadth-first search (BFS).

Search

```
Random random = new Random();
System.out.println("0");
if (random.nextBoolean()) {
  System.out.println("2");
} else {
  System.out.println("1");
  if (random.nextBoolean()) {
    System.out.println("4");
  } else {
    System.out.println("3");
    if (random.nextBoolean()) {
      System.out.println("6");
    } else {
      System.out.println("5");
    }
  }
}
```

Task 2

Verify your program using JPF with BFS and DFS. To do that, you need to create an application properties file (.jpf file) for your Java app developed in Task 1. Configure the search property to be `gov.nasa.jpf.search.heuristic.BFSHeuristic` or `gov.nasa.jpf.search.heuristic.DFSHeuristic`.

Search

```
target=Traversal
classpath=.
cg.enumerate_random=true
search=gov.nasa.jpf.search.heuristic.BFSHeuristic
```

Search

```
========================================================== syst
Traversal.main()
========================================================== sea
0
1
3
5
6
4
2
==========================================================
```

Search

That is not breadth first search!

Question

Have we set the search property correctly? How can we check that?

Search

That is not breadth first search!

Question

Have we set the search property correctly? How can we check that?

Answer

Use the following command line arguments

- `-log`: lists the order in which properties files got loaded
- `-show`: prints all configuration entries after the initialization is complete

Log option

```
loading property file: ...\.jpf\site.properties
loading property file: ...\jpf\jpf-core\jpf.properties
collected native_classpath=...\jpf\jpf-core/build/jpf.jar,
collected native_libraries=null
```

Show option

```
...
search = gov.nasa.jpf.search.heuristic.BFSHeuristic
search.class = gov.nasa.jpf.search.DFSearch
...
```

Search

```
target=Traversal
classpath=.
cg.enumerate_random=true
search.class=gov.nasa.jpf.search.heuristic.BFSHeuristic
```

Search

```
========================================================== syst
Traversal.main()
========================================================== sea
0
1
2
3
4
5
6
==========================================================
```

Search

```
target=Traversal
classpath=.
cg.enumerate_random=true
search.class=gov.nasa.jpf.search.heuristic.DFSHeuristic
```

Search

```
========================================================== syst
Traversal.main()
========================================================== sea
0
1
2
3
4
5
6
==========================================================
```

Search

That is not depth first search!

Search

That is not depth first search!

Lets try instead `gov.nasa.jpf.search.DFSearch`.

Search

```
target=Traversal
classpath=.
cg.enumerate_random=true
search.class=gov.nasa.jpf.search.DFSearch
```

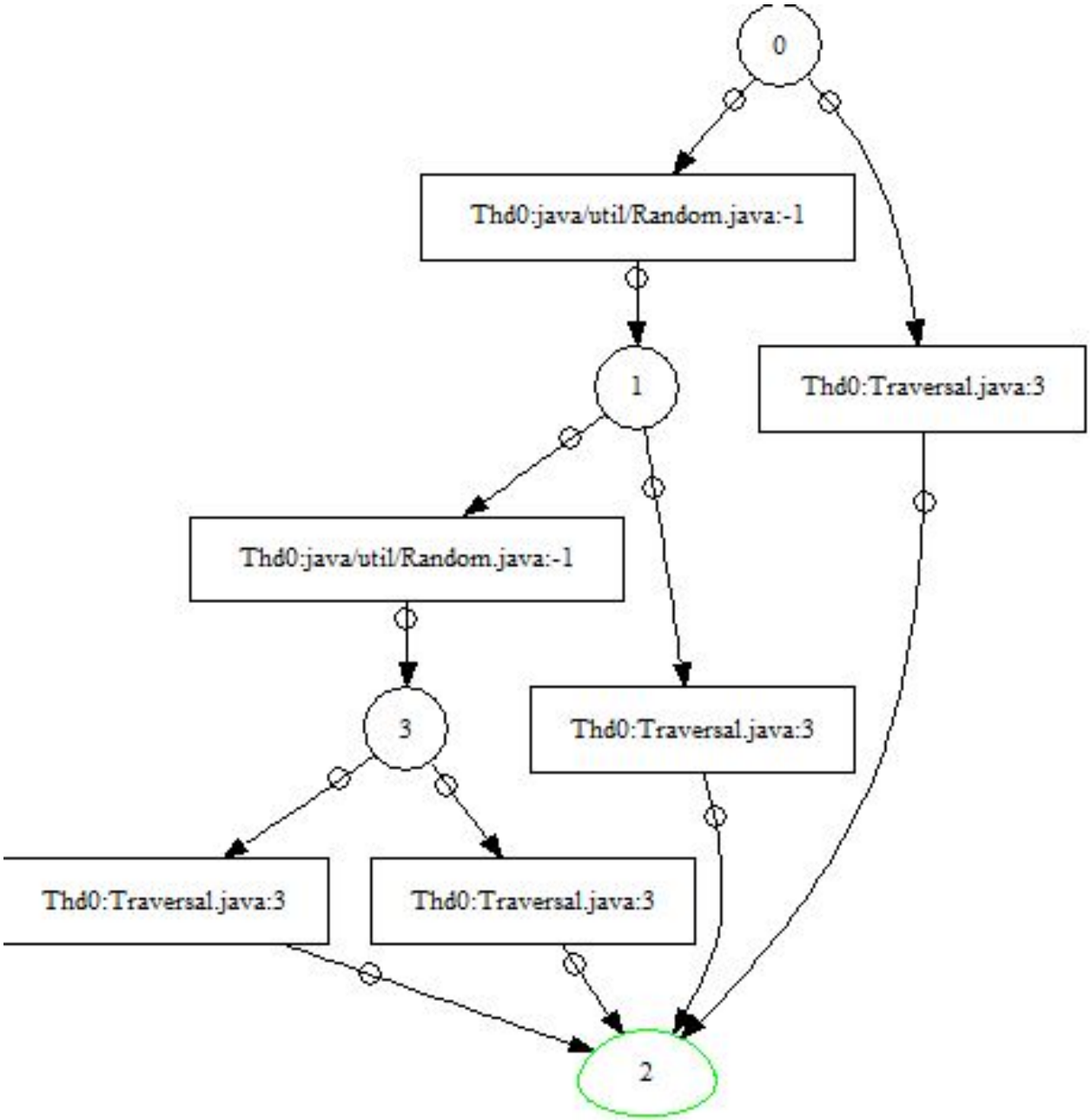
Search

```
========================================================== syst
Traversal.main()
========================================================== sea
0
1
3
5
6
4
2
==========================================================
```

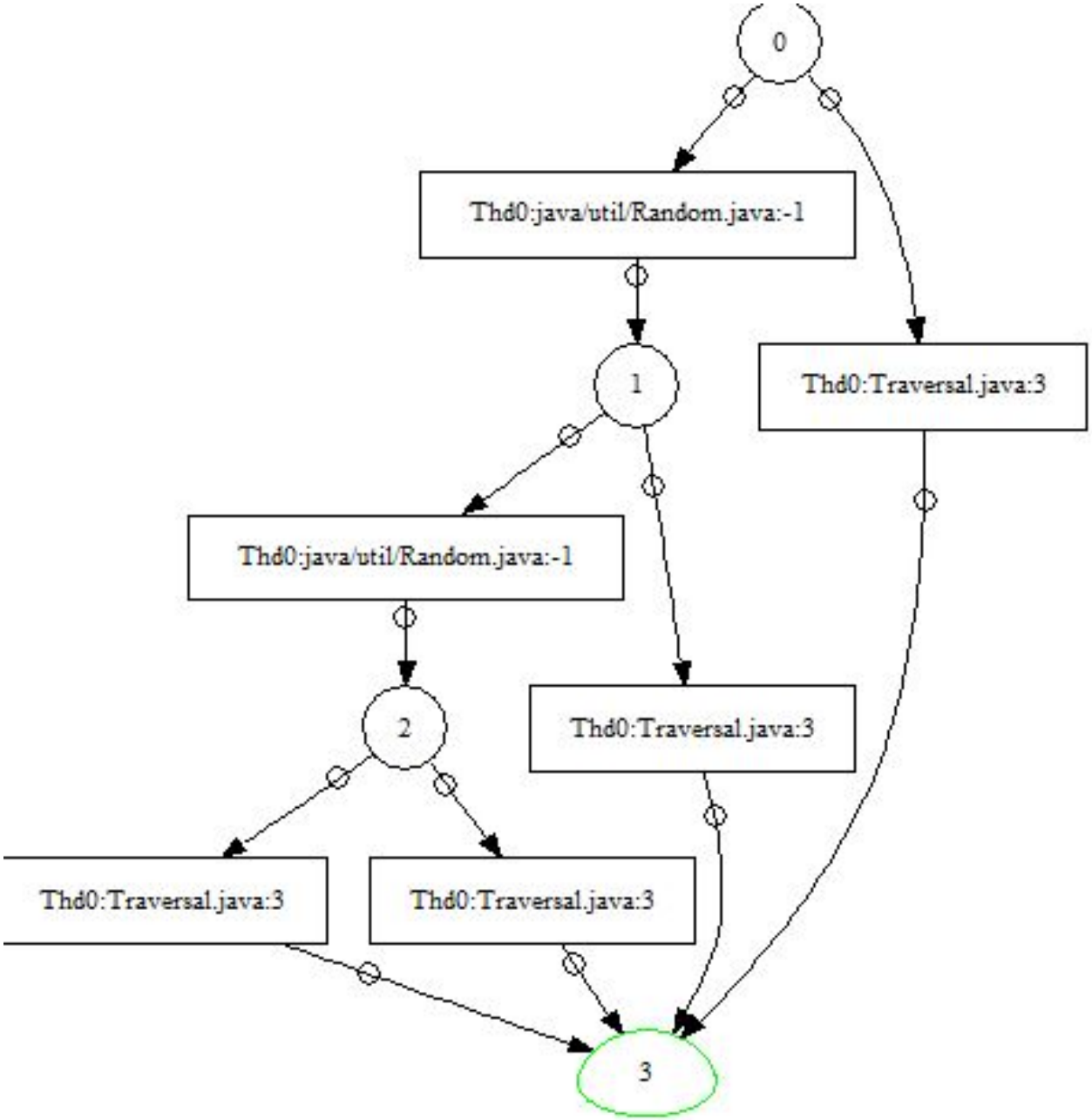
Task 3

Generate the state space diagram for BFS and DFS. To do this you need to set the listener to `StateSpaceDot`.

BFS


0
Thd0:java/util/Random.java:-1
1
Thd0:Traversal.java:3
Thd0:java/util/Random.java:-1
3
Thd0:Traversal.java:3
Thd0:Traversal.java:3
Thd0:Traversal.java:3
2

DFS


0
Thd0:java/util/Random.java:-1
1
Thd0:Traversal.java:3
Thd0:java/util/Random.java:-1
2
Thd0:Traversal.java:3
Thd0:Traversal.java:3
Thd0:Traversal.java:3
3

Task 4

Verify your program using RS. RS can explore several random executions and in JPF you have the freedom to set the maximum number of executions you would like RS to explore. Firstly, set your search strategy to `gov.nasa.jpf.search.RandomSearch`. Secondly, set the `search.RandomSearch.path_limit` property to be any integer larger than 0. Compare the resulting state space diagrams.

Search

```
JavaPathfinder core system v8.0 (rev 2+) - (C) 2005-2014 Ur

========================================================== syst
Traversal.main()

========================================================== sear
0
1
3
5
```

Search

No space diagram has been produced.

Search

No space diagram has been produced.

Bugs are everywhere, even in JPF!