jpf-label

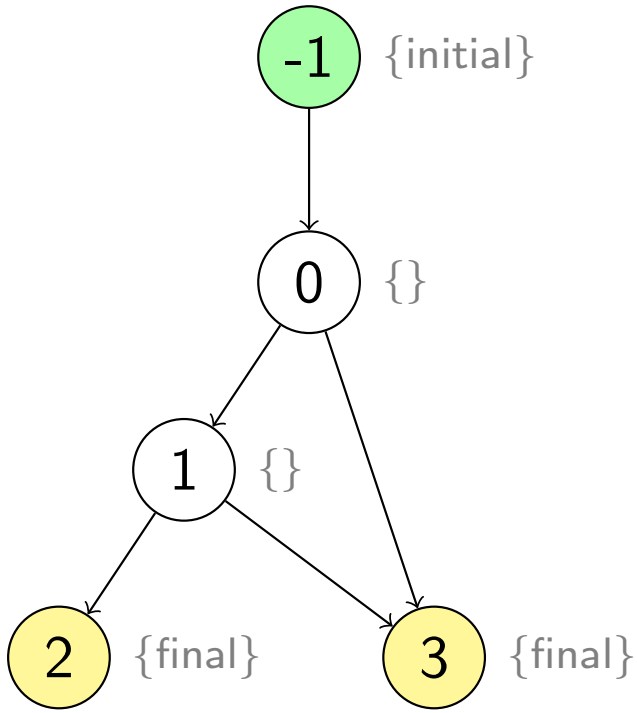
Zainab Fatmi

November 2020

Introduction

Software is often modelled as a transition system. To capture simple known facts about the modelled software, the states are usually labelled with a set of atomic propositions.

These atomic propositions may be used to express properties of the code.

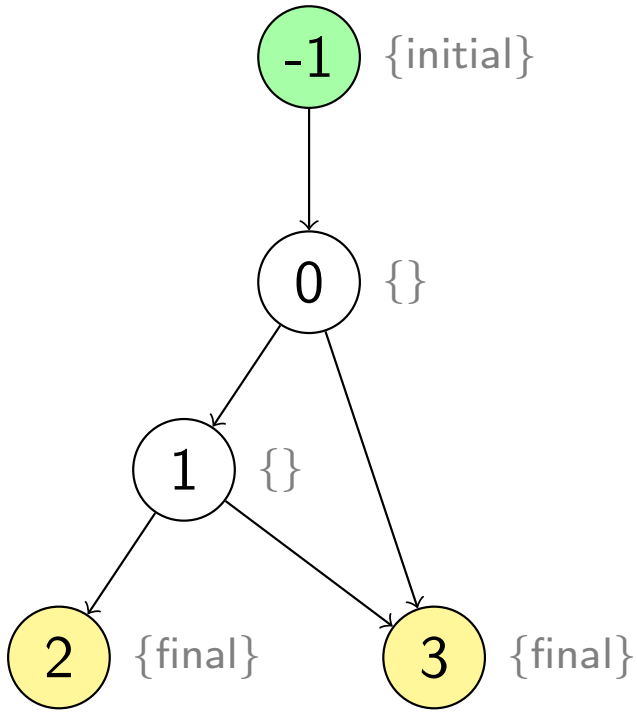


Introduction

Software is often modelled as a transition system. To capture simple known facts about the modelled software, the states are usually labelled with a set of atomic propositions.

These atomic propositions may be used to express properties of the code.

JPF does not support the labelling of states with atomic propositions.



jpf-label

We developed an extension of JPF, named jpf-label, that allows users to easily label states with atomic propositions, by defining custom labelling functions.

jpf-label

We developed an extension of JPF, named jpf-label, that allows users to easily label states with atomic propositions, by defining custom labelling functions.

Our extension supports both

- state labelling and
- transition labelling.

The labelled transition system can be represented

- graphically as a dot file or
- textually as a label file.

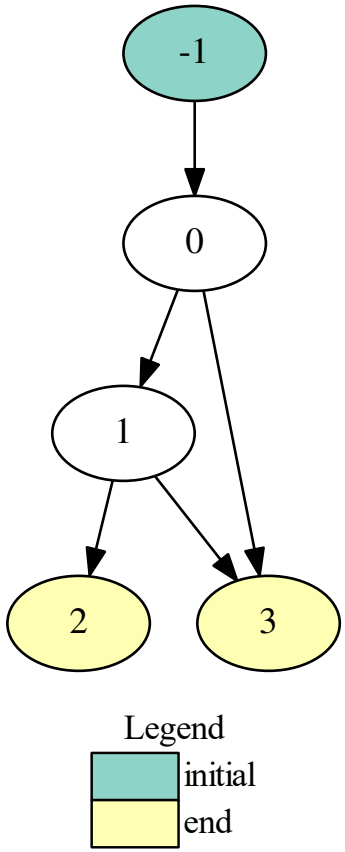
jpf-label

We provide twelve different ways to label states, including:

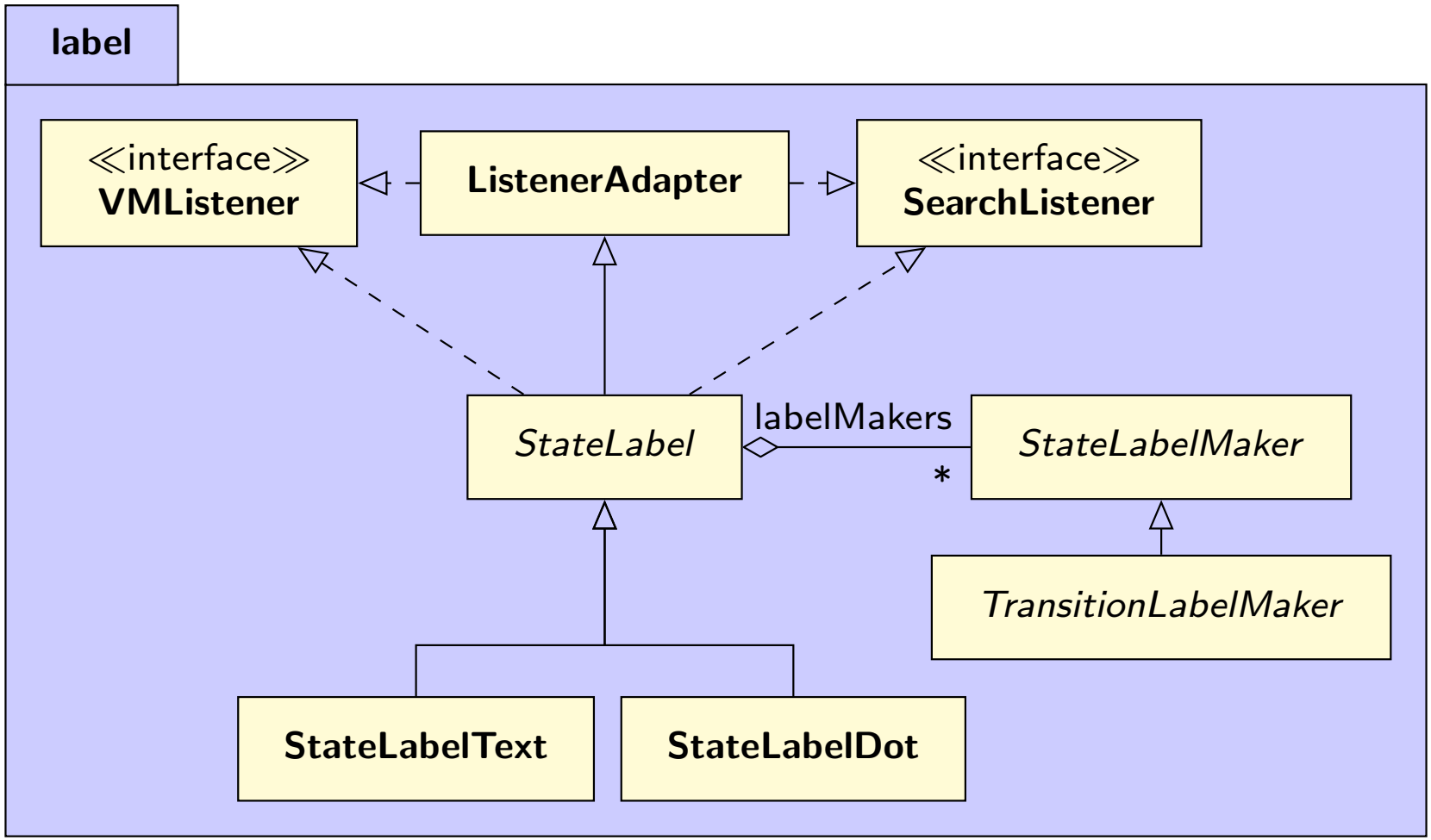
- initial and final states,
- boolean and integer static fields,
- boolean and integer local variables,
- method invocations,
- method returns and the values returned, and
- thrown exceptions and the exception types.

Example: Initial and Final States

```
1  import java.util.Random;
2
3  public class Field {
4    /* state -1 */
5    private static boolean value = true;
6
7    public static void main(String[] args) {
8      Random random = new Random();
9      /* state 0 */
10     if (random.nextBoolean()) {
11       Field.value = false;
12       Field.value = true;
13     } else {
14       /* state 1 */
15       Field.value = random.nextBoolean();
16     }
17     /* state 2 if Field.value is false,
18        state 3 otherwise */
19   }
20 }
```

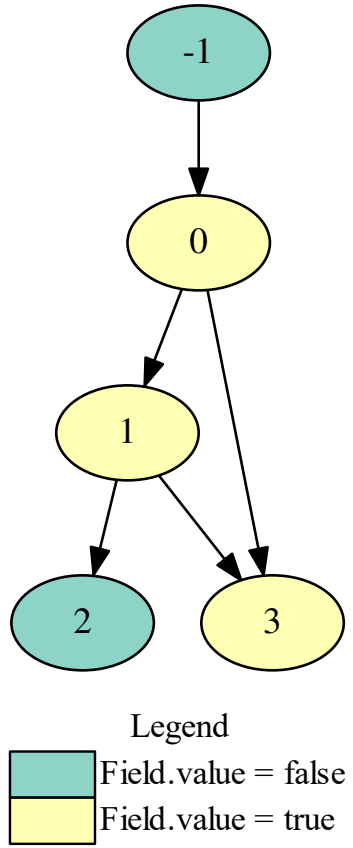


Design Decisions



Example: Boolean Static Field

```
1  import java.util.Random;
2
3  public class Field {
4    /* state -1 */
5    private static boolean value = true;
6
7    public static void main(String[] args) {
8      Random random = new Random();
9      /* state 0 */
10     if (random.nextBoolean()) {
11       Field.value = false;
12       Field.value = true;
13     } else {
14       /* state 1 */
15       Field.value = random.nextBoolean();
16     }
17     /* state 2 if Field.value is false,
18        state 3 otherwise */
19   }
20 }
```



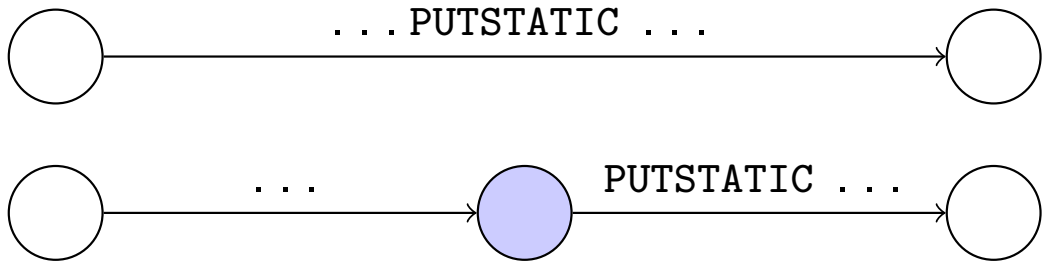
Transition Labels

Transitions between states represent the execution of a sequence of bytecode instructions. In the event that we wish to observe a specific instruction, for example PUTSTATIC, we may break the transition

Transition Labels

Transitions between states represent the execution of a sequence of bytecode instructions. In the event that we wish to observe a specific instruction, for example `PUTSTATIC`, we may break the transition

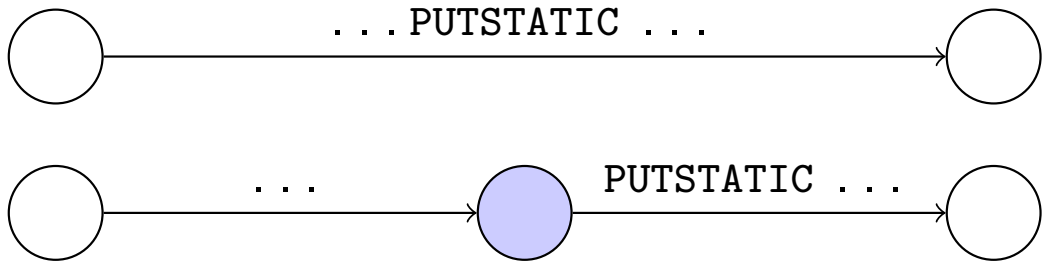
- either before



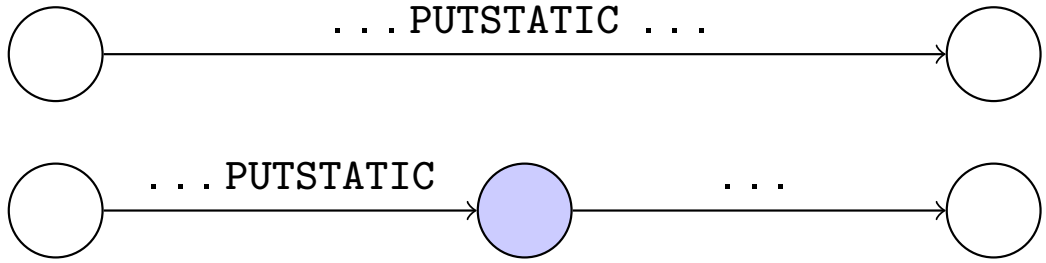
Transition Labels

Transitions between states represent the execution of a sequence of bytecode instructions. In the event that we wish to observe a specific instruction, for example `PUTSTATIC`, we may break the transition

- either before



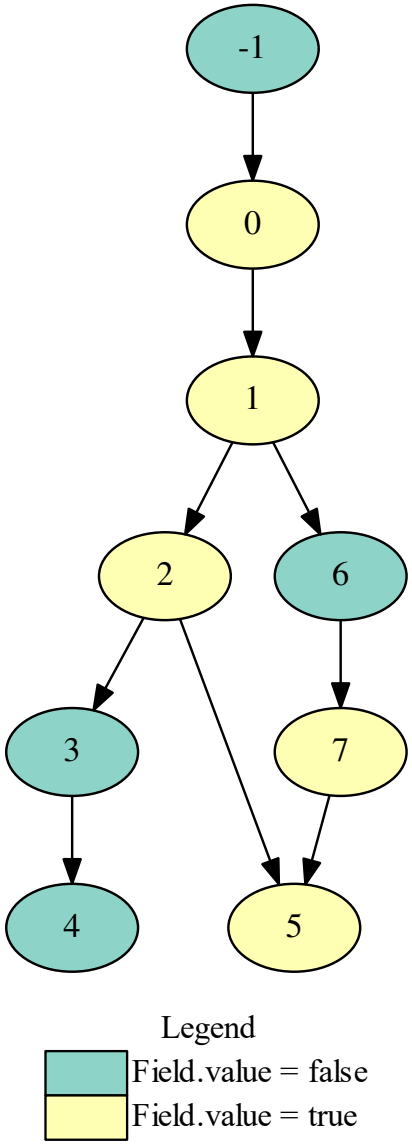
- or after



the point of interest and label the newly generated state.

Example: Boolean Static Field

```
1  import java.util.Random;
2
3  public class Field {
4    /* state -1 */
5    private static boolean value = true;
6
7    public static void main(String[] args) {
8      Random random = new Random();
9      /* state 1 */
10     if (random.nextBoolean()) {
11       Field.value = false;
12       Field.value = true;
13     } else {
14       /* state 2 */
15       Field.value = random.nextBoolean();
16     }
17     /* state 4 if Field.value is false,
18        state 5 otherwise */
19   }
20 }
```



Thank You

jpf-label can be found at:
github.com/javapathfinder/jpf-label