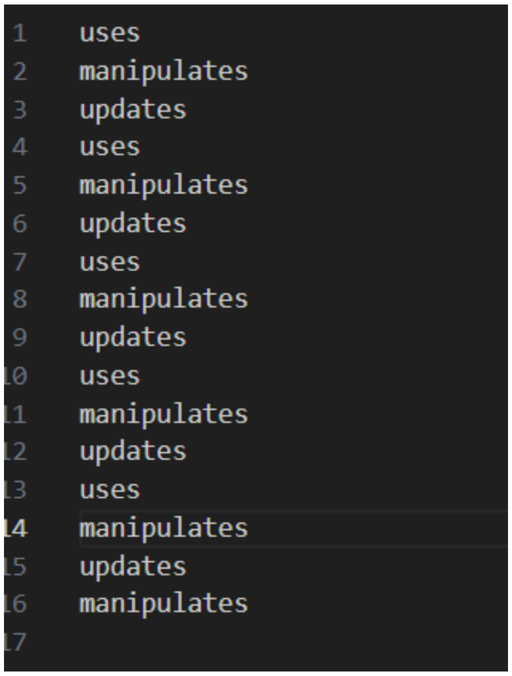
In-class Exercise 4

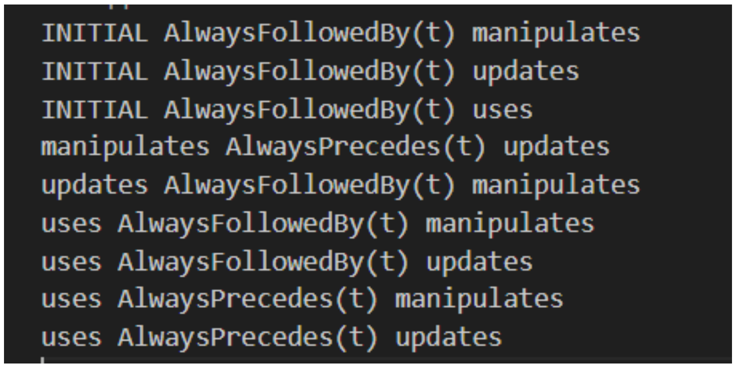
Group Members: Anshumaan Chauhan, Hsin-Yu Wen

Version 1

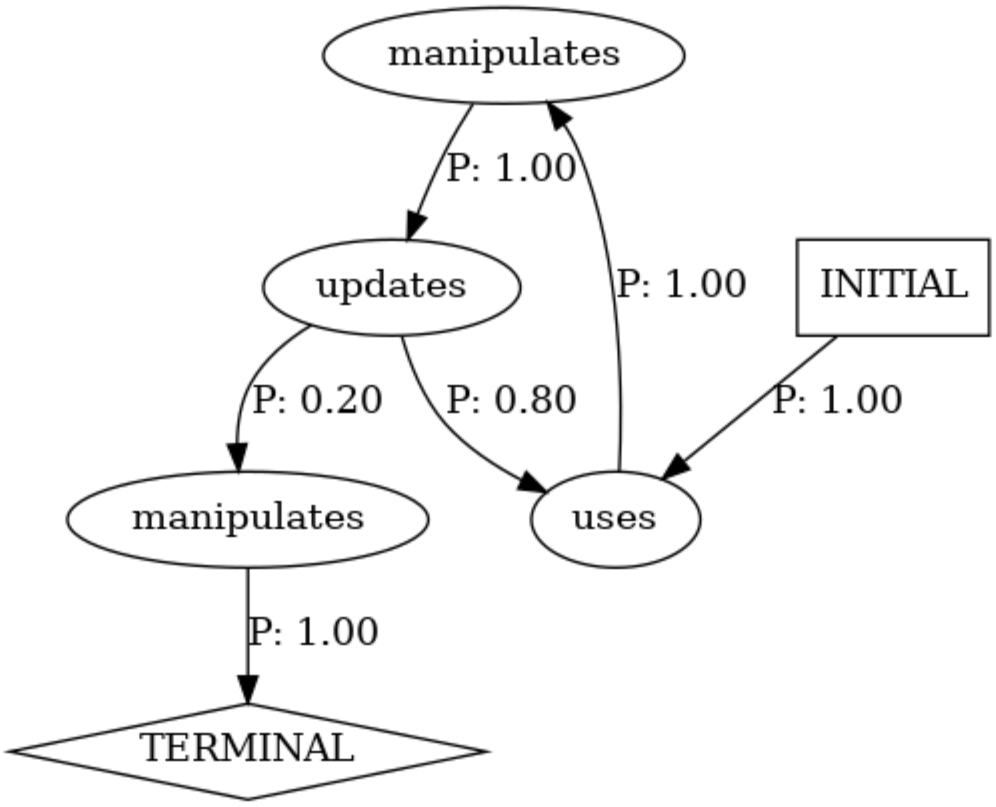
Player 1 Wins

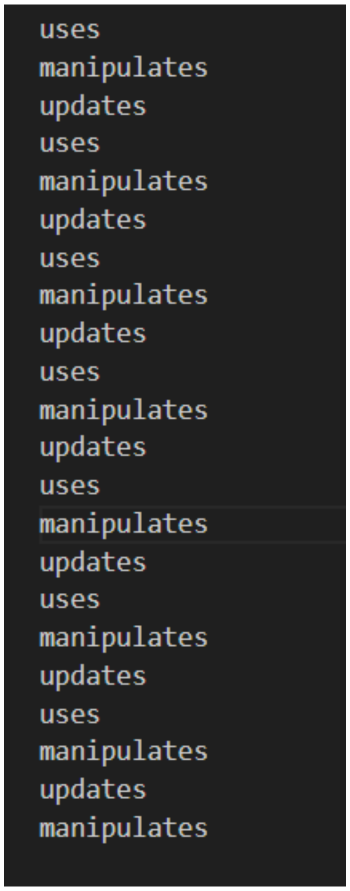


```
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
manipulates
```

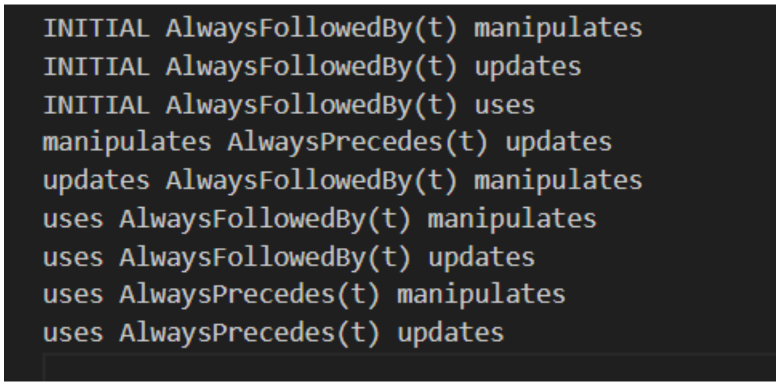


```
INITIAL AlwaysFollowedBy(t) manipulates
INITIAL AlwaysFollowedBy(t) updates
INITIAL AlwaysFollowedBy(t) uses
manipulates AlwaysPrecedes(t) updates
updates AlwaysFollowedBy(t) manipulates
uses AlwaysFollowedBy(t) manipulates
uses AlwaysFollowedBy(t) updates
uses AlwaysPrecedes(t) manipulates
uses AlwaysPrecedes(t) updates
```

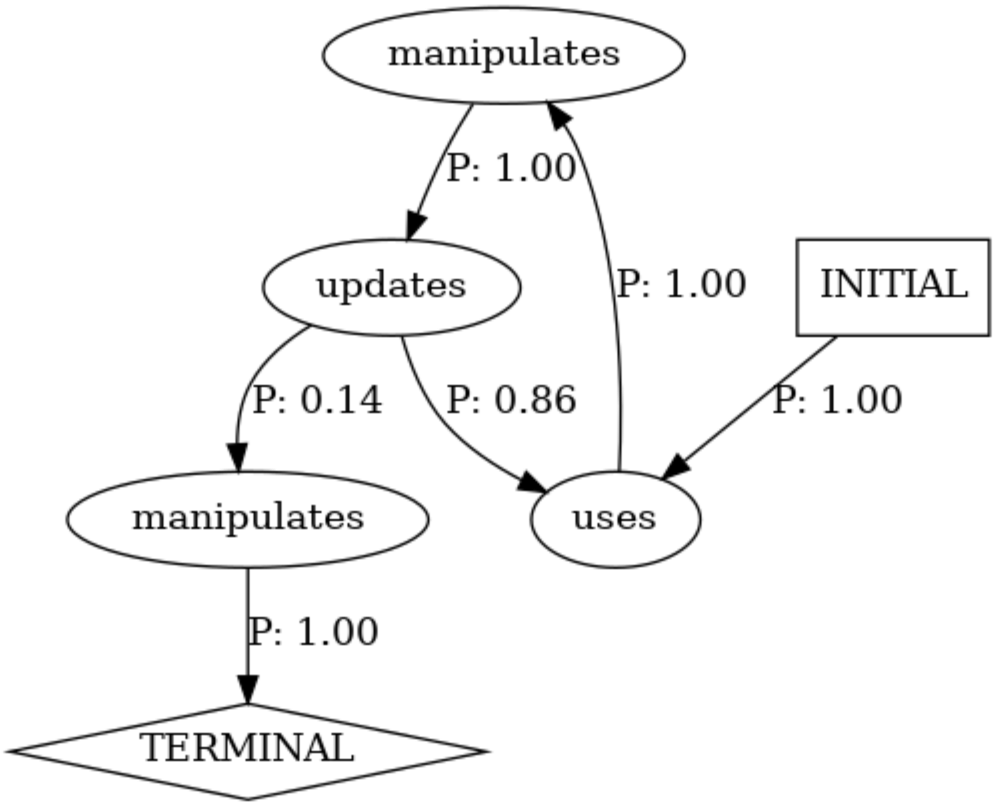

manipulates
P: 1.00
updates
P: 1.00
INITIAL
P: 0.20
P: 0.80
P: 1.00
manipulates
uses
P: 1.00
TERMINAL

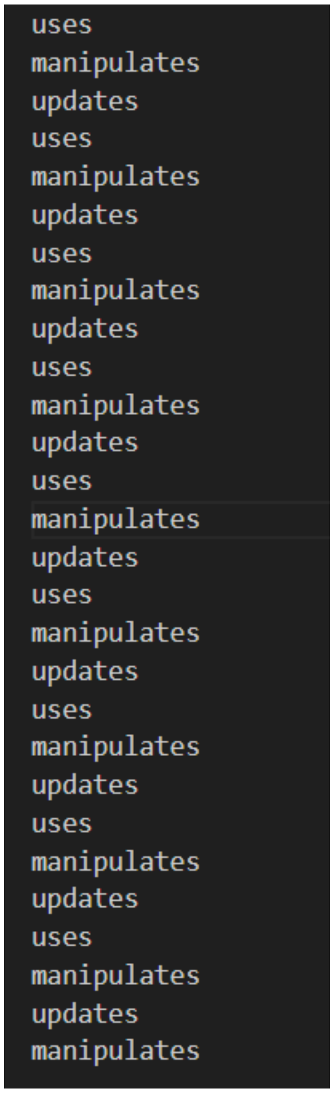


```
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
manipulates
```

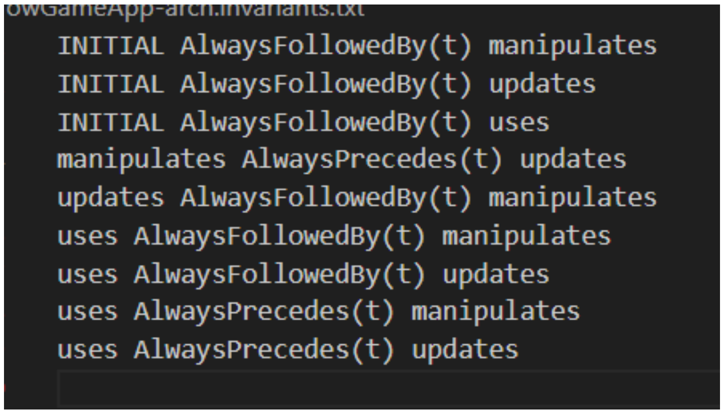


```
INITIAL AlwaysFollowedBy(t) manipulates
INITIAL AlwaysFollowedBy(t) updates
INITIAL AlwaysFollowedBy(t) uses
manipulates AlwaysPrecedes(t) updates
updates AlwaysFollowedBy(t) manipulates
uses AlwaysFollowedBy(t) manipulates
uses AlwaysFollowedBy(t) updates
uses AlwaysPrecedes(t) manipulates
uses AlwaysPrecedes(t) updates
```


manipulates
P: 1.00
updates
P: 1.00
INITIAL
P: 0.14
P: 0.86
P: 1.00
manipulates
uses
P: 1.00
TERMINAL

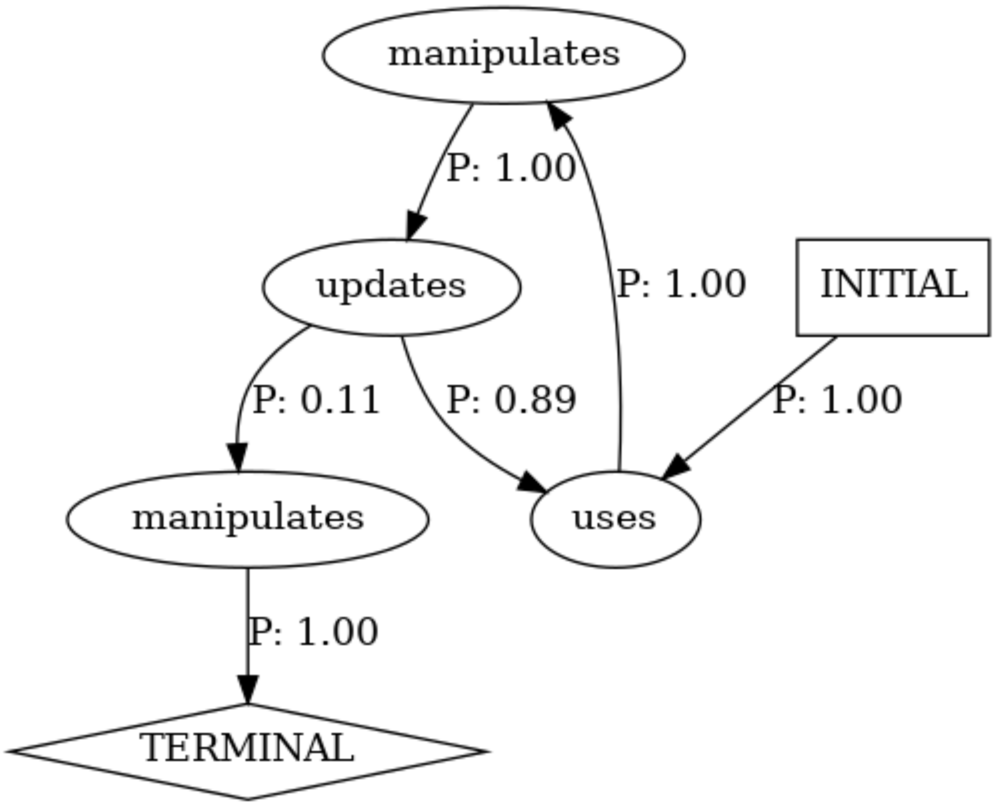


```
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
manipulates
```

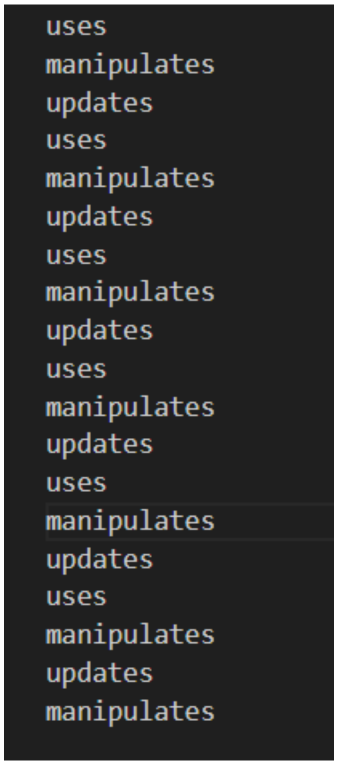


owGameApp-arch.invariants.txt

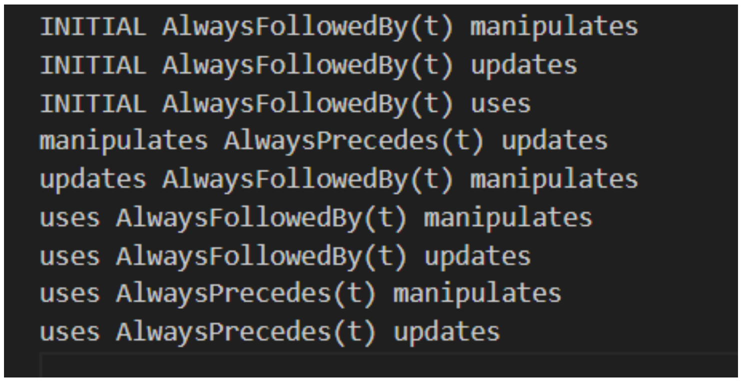
```
INITIAL AlwaysFollowedBy(t) manipulates
INITIAL AlwaysFollowedBy(t) updates
INITIAL AlwaysFollowedBy(t) uses
manipulates AlwaysPrecedes(t) updates
updates AlwaysFollowedBy(t) manipulates
uses AlwaysFollowedBy(t) manipulates
uses AlwaysFollowedBy(t) updates
uses AlwaysPrecedes(t) manipulates
uses AlwaysPrecedes(t) updates
```



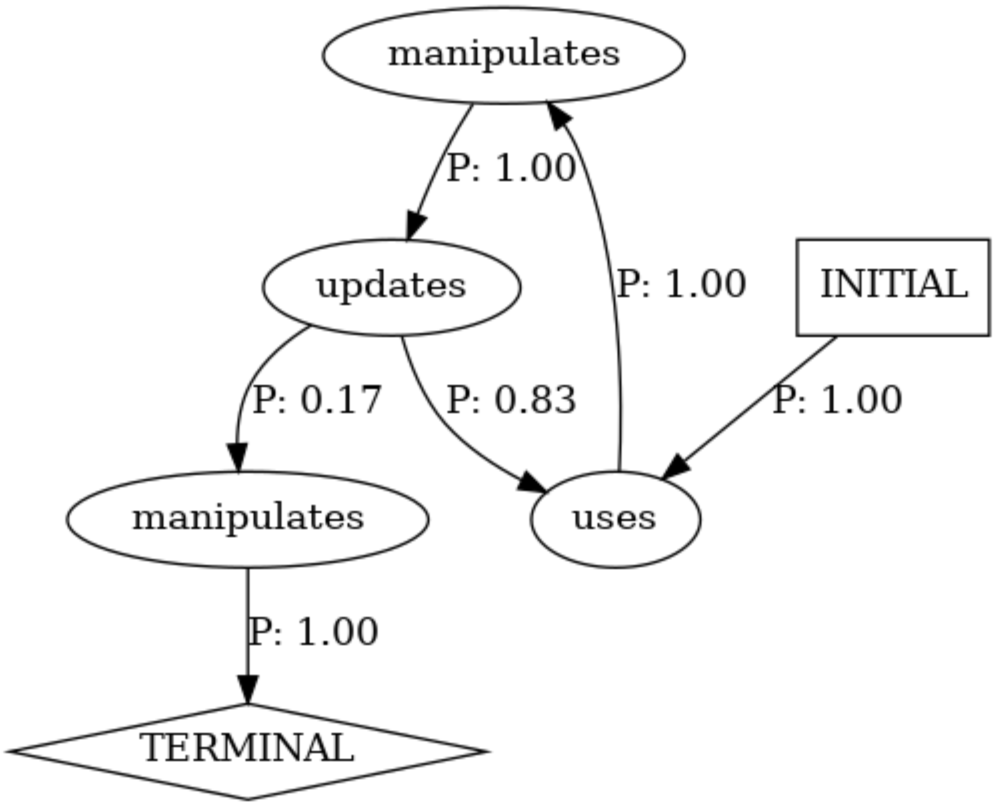
Player 2 Wins

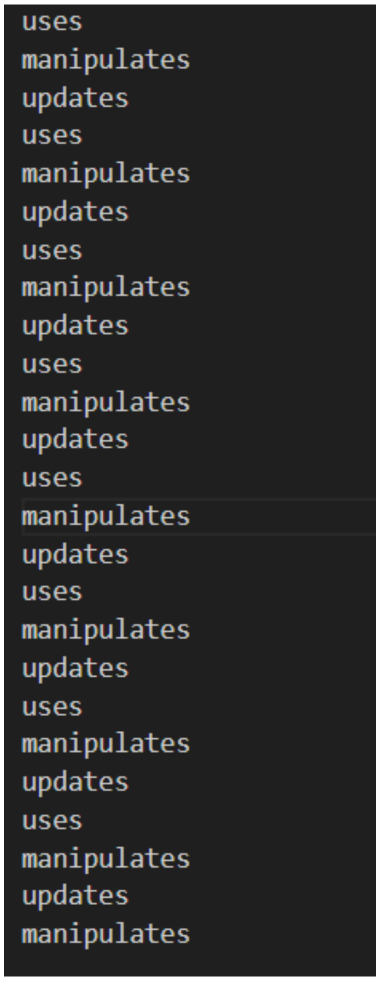


```
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
manipulates
```

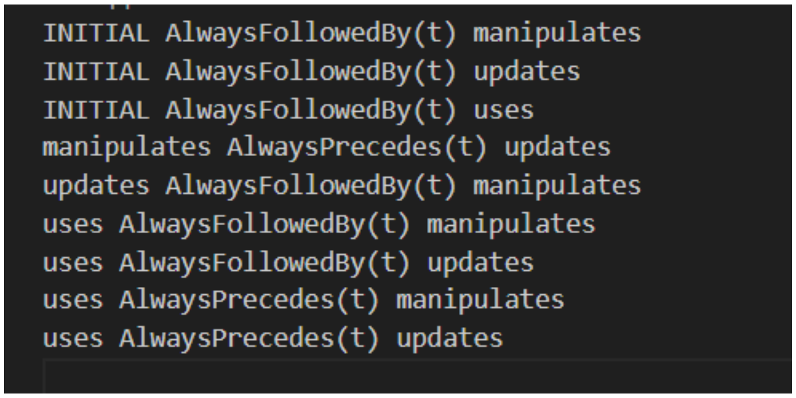


```
INITIAL AlwaysFollowedBy(t) manipulates
INITIAL AlwaysFollowedBy(t) updates
INITIAL AlwaysFollowedBy(t) uses
manipulates AlwaysPrecedes(t) updates
updates AlwaysFollowedBy(t) manipulates
uses AlwaysFollowedBy(t) manipulates
uses AlwaysFollowedBy(t) updates
uses AlwaysPrecedes(t) manipulates
uses AlwaysPrecedes(t) updates
```

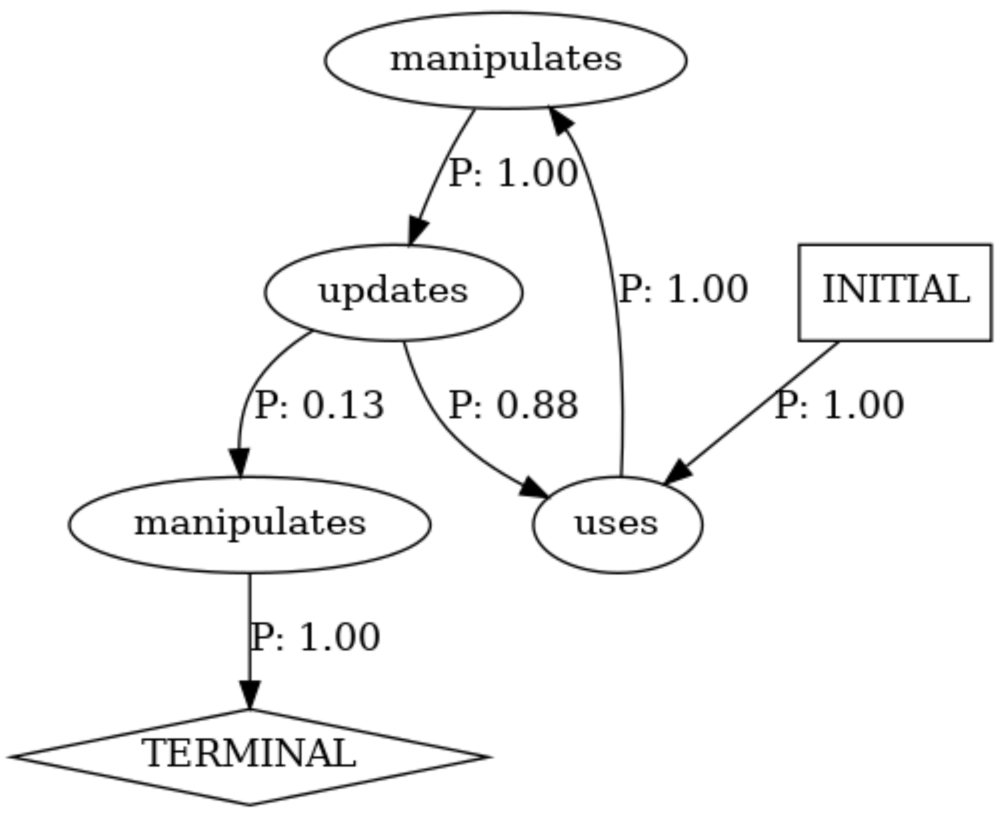

manipulates
P: 1.00
updates
P: 1.00
INITIAL
P: 0.17
P: 0.83
P: 1.00
manipulates
uses
P: 1.00
TERMINAL



```
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
manipulates
```



```
INITIAL AlwaysFollowedBy(t) manipulates
INITIAL AlwaysFollowedBy(t) updates
INITIAL AlwaysFollowedBy(t) uses
manipulates AlwaysPrecedes(t) updates
updates AlwaysFollowedBy(t) manipulates
uses AlwaysFollowedBy(t) manipulates
uses AlwaysFollowedBy(t) updates
uses AlwaysPrecedes(t) manipulates
uses AlwaysPrecedes(t) updates
```



No one Wins

uses

manipulates

updates

uses

manipulates

updates

uses

manipulates

updates

uses

manipulates

updates

uses

manipulates

updates

uses

manipulates

updates

uses

manipulates

updates

uses

manipulates

updates

uses

manipulates

updates

manipulates

INITIAL AlwaysFollowedBy(t) manipulates

INITIAL AlwaysFollowedBy(t) updates

INITIAL AlwaysFollowedBy(t) uses

manipulates AlwaysPrecedes(t) updates

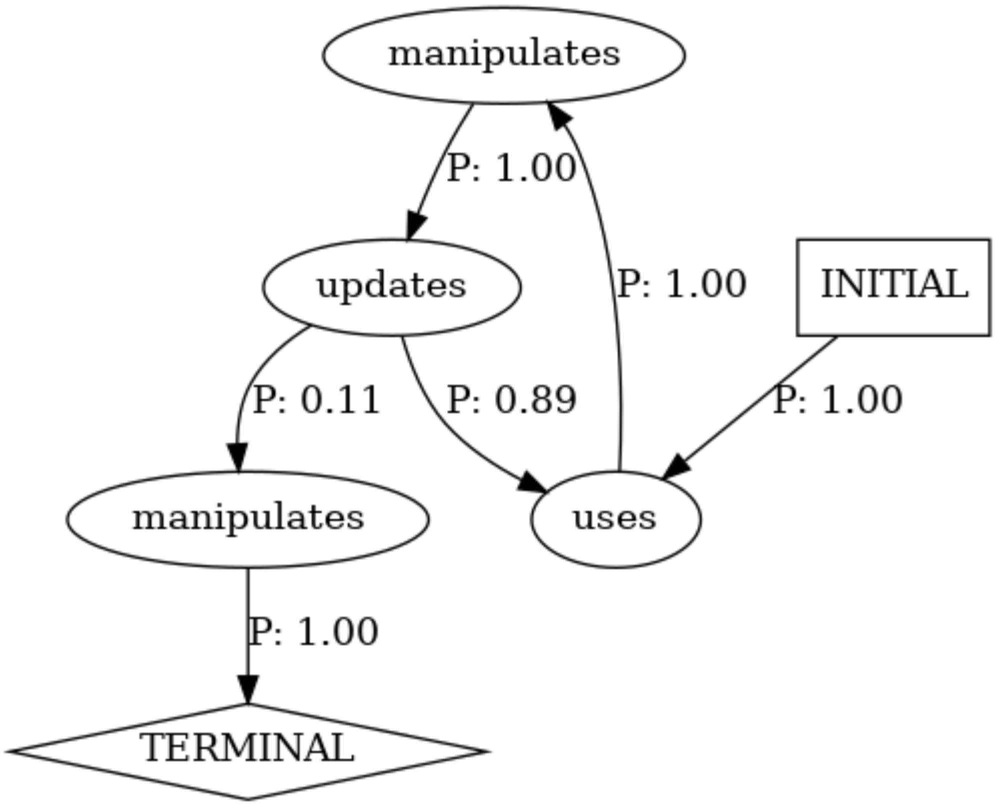
updates AlwaysFollowedBy(t) manipulates

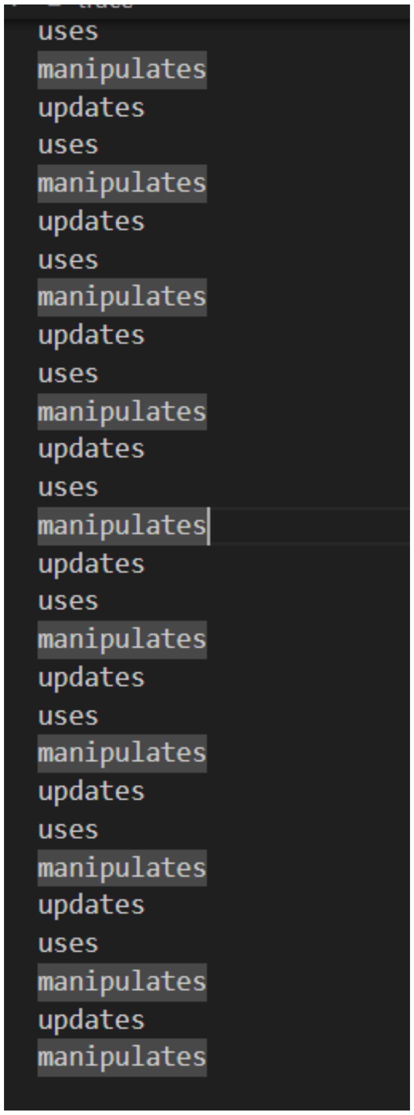
uses AlwaysFollowedBy(t) manipulates

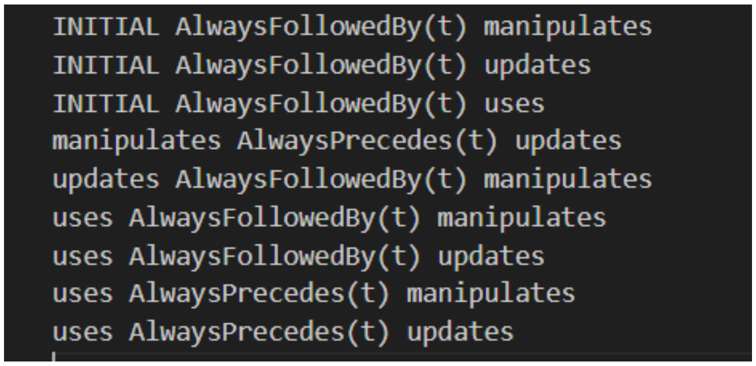
uses AlwaysFollowedBy(t) updates

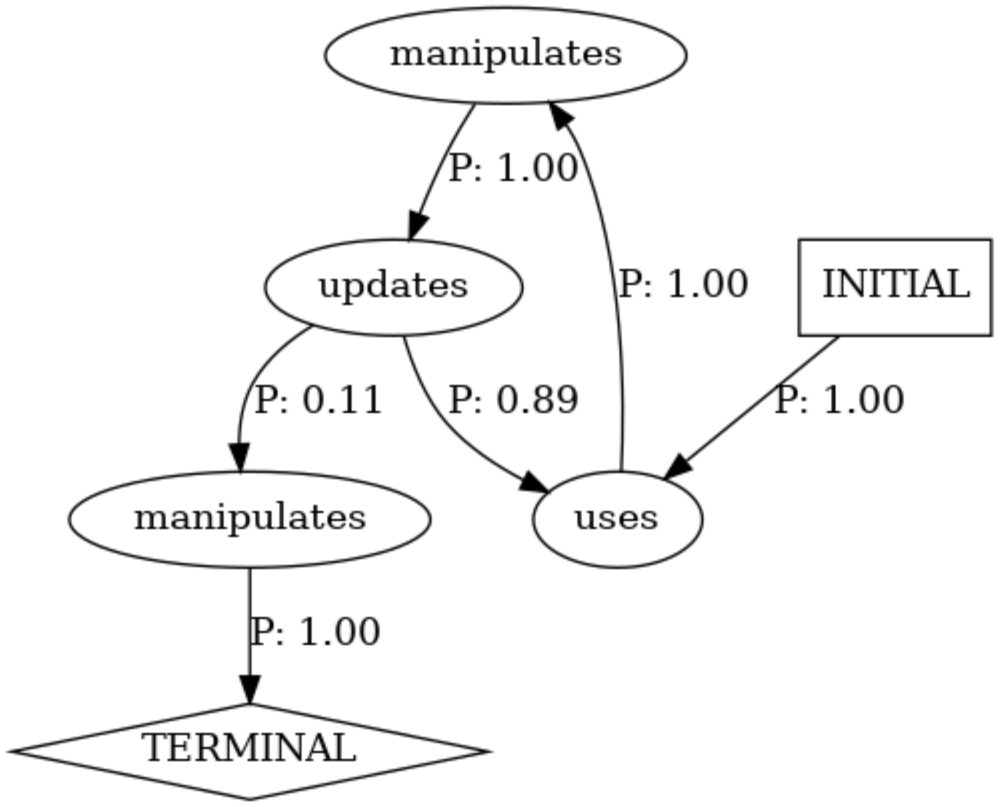
uses AlwaysPrecedes(t) manipulates

uses AlwaysPrecedes(t) updates


manipulates
P: 1.00
updates
P: 1.00
INITIAL
P: 0.11
P: 0.89
P: 1.00
manipulates
uses
P: 1.00
TERMINAL


```
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
manipulates
```


```
INITIAL AlwaysFollowedBy(t) manipulates
INITIAL AlwaysFollowedBy(t) updates
INITIAL AlwaysFollowedBy(t) uses
manipulates AlwaysPrecedes(t) updates
updates AlwaysFollowedBy(t) manipulates
uses AlwaysFollowedBy(t) manipulates
uses AlwaysFollowedBy(t) updates
uses AlwaysPrecedes(t) manipulates
uses AlwaysPrecedes(t) updates
```



Unexpected Behavior

```
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
```

```
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
manipulates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
```

```
uses
manipulates
updates
uses
manipulates
updates
uses
manipulates
updates
manipulates
```

INITIAL AlwaysFollowedBy(t) manipulates

INITIAL AlwaysFollowedBy(t) updates

INITIAL AlwaysFollowedBy(t) uses

manipulates AlwaysPrecedes(t) updates

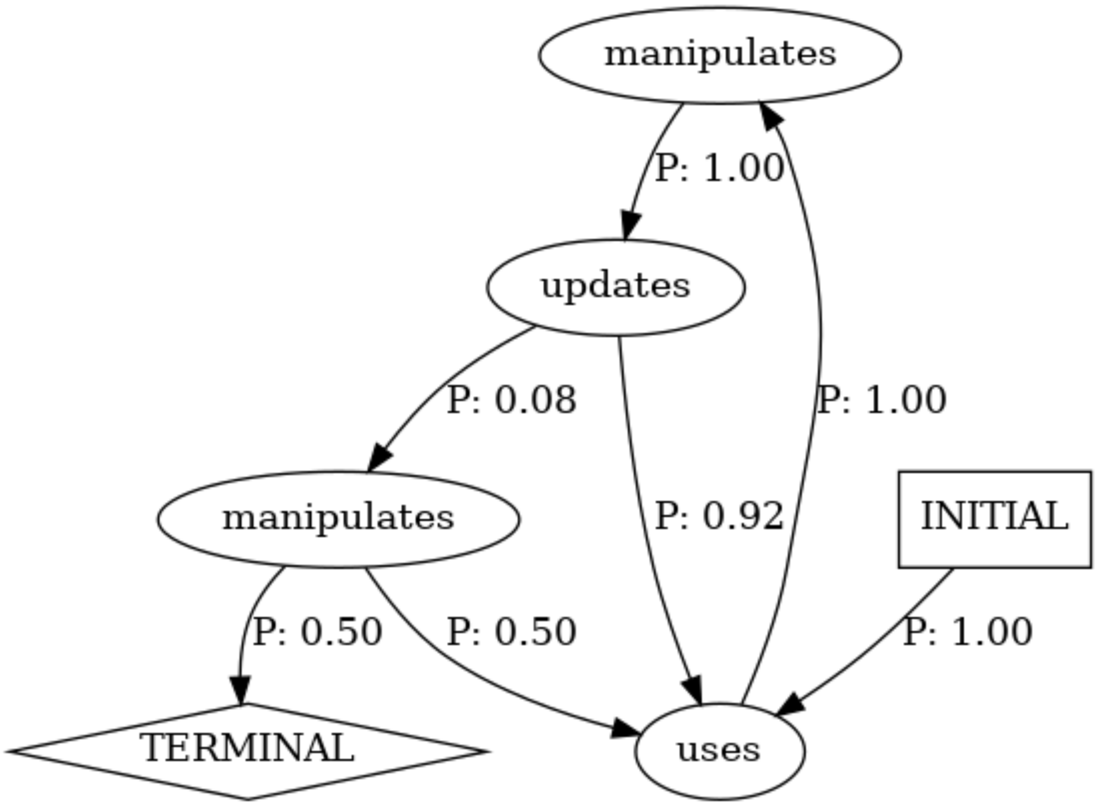
updates AlwaysFollowedBy(t) manipulates

uses AlwaysFollowedBy(t) manipulates

uses AlwaysFollowedBy(t) updates

uses AlwaysPrecedes(t) manipulates

uses AlwaysPrecedes(t) updates

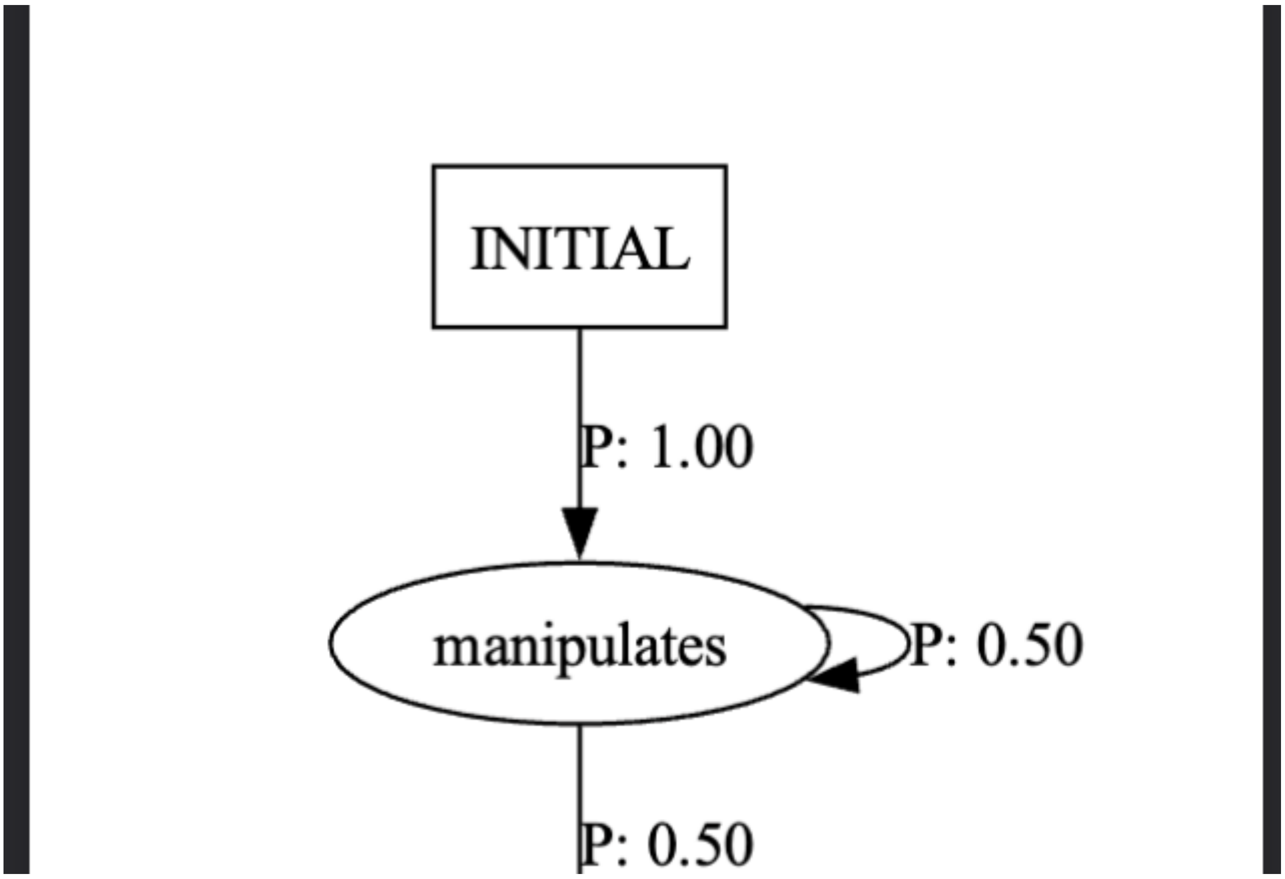

manipulates
P: 1.00
updates
P: 0.08
P: 1.00
P: 0.92
manipulates
INITIAL
P: 0.50
P: 0.50
P: 1.00
TERMINAL
uses

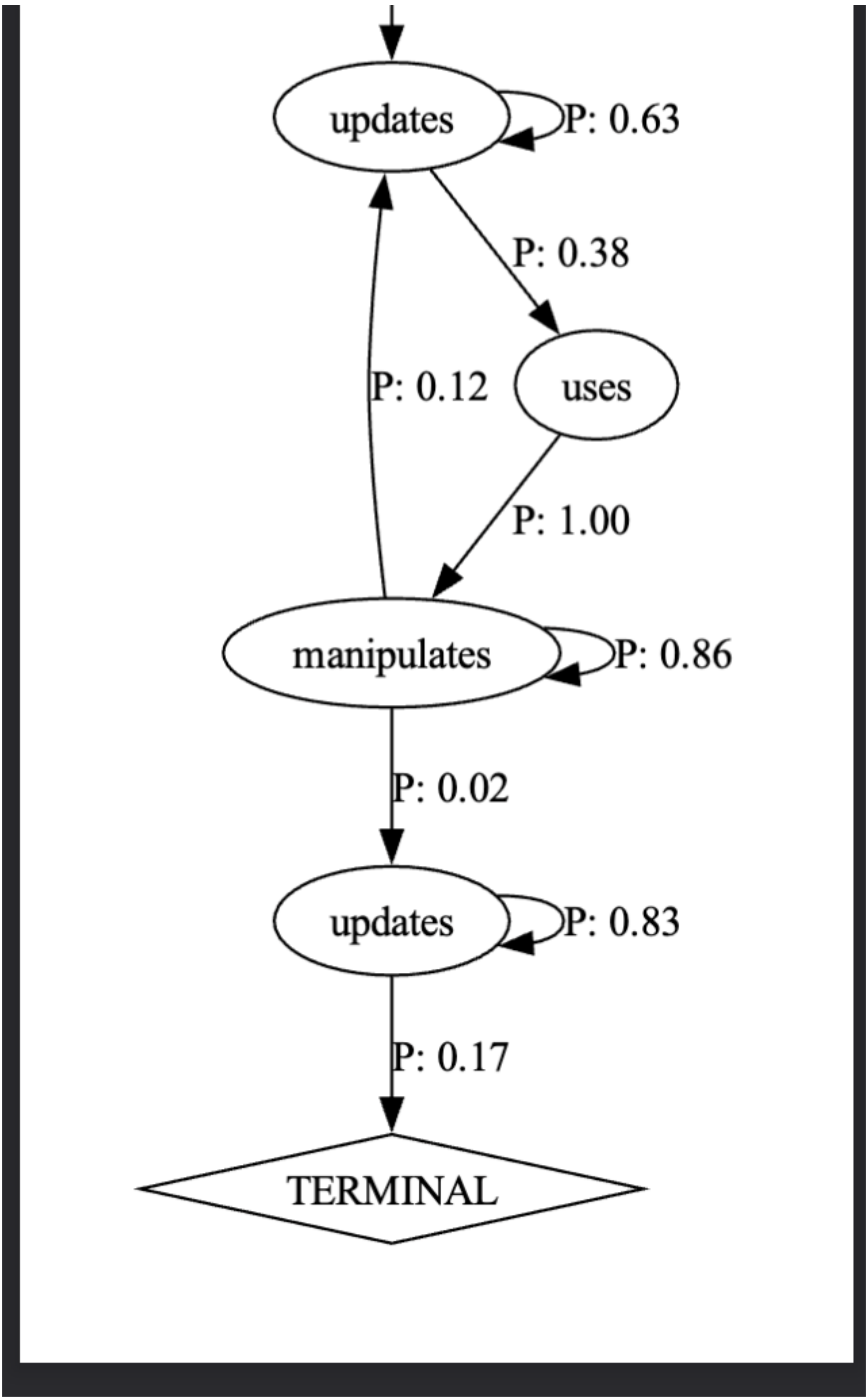
Version 2

trace-1 (traces would be attched in the folder)

player 1 won

```
digraph G {
  0 [label="manipulates"];
  1 [label="manipulates"];
  2 [label="updates"];
  3 [label="updates"];
  4 [label="uses"];
  5 [label="TERMINAL",shape=diamond];
  6 [label="INITIAL",shape=box];
0->0 [label="P: 0.50"];
0->3 [label="P: 0.50"];
1->1 [label="P: 0.86"];
1->2 [label="P: 0.02"];
1->3 [label="P: 0.12"];
2->2 [label="P: 0.83"];
2->5 [label="P: 0.17"];
3->3 [label="P: 0.63"];
3->4 [label="P: 0.38"];
4->1 [label="P: 1.00"];
6->0 [label="P: 1.00"];
}
```




updates
P: 0.63
P: 0.38
uses
P: 0.12
P: 1.00
manipulates
P: 0.86
P: 0.02
updates
P: 0.83
P: 0.17
TERMINAL

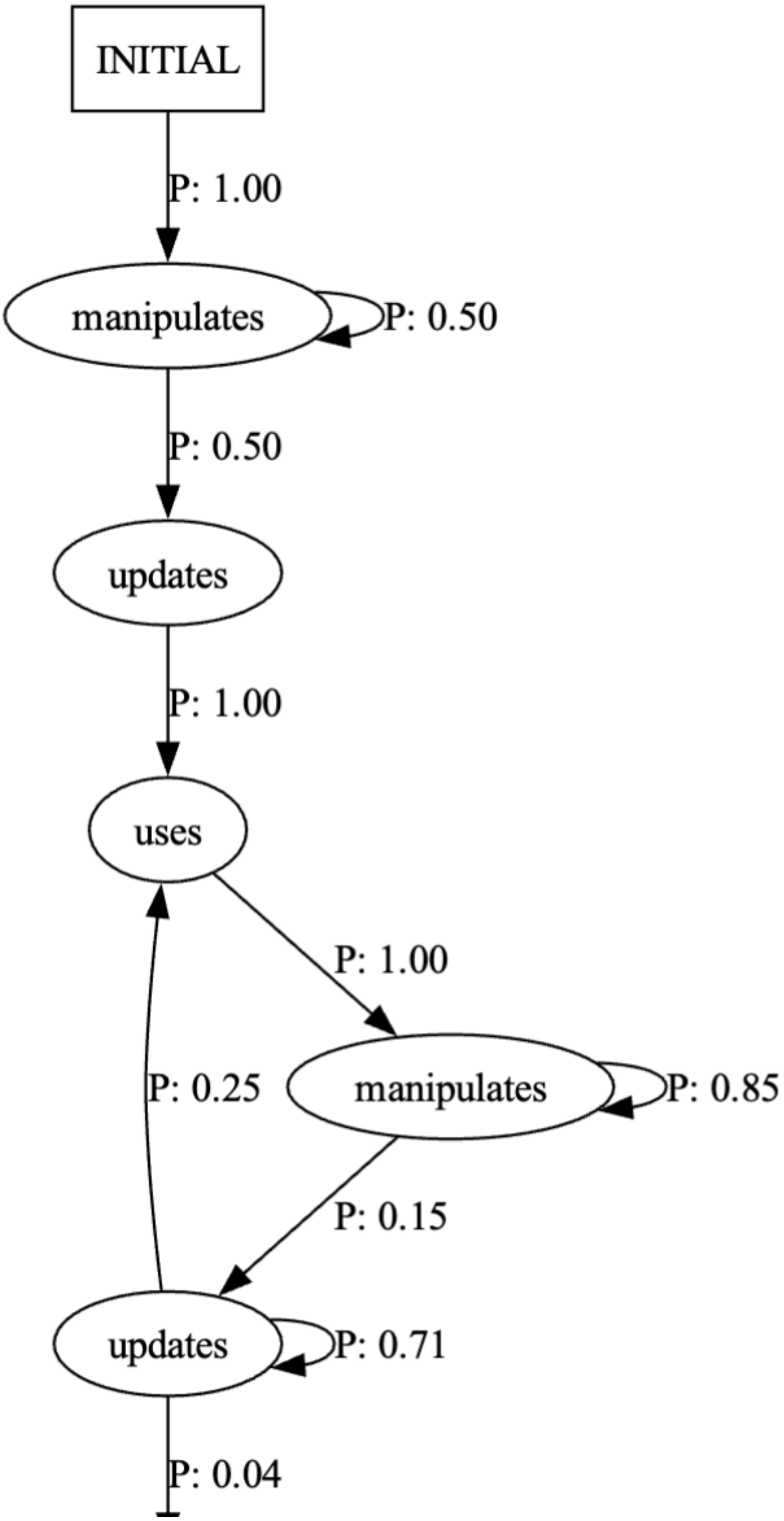
```
INITIAL AlwaysFollowedBy(t) manipulates
INITIAL AlwaysFollowedBy(t) updates
INITIAL AlwaysFollowedBy(t) uses
manipulates AlwaysFollowedBy(t) updates
manipulates AlwaysPrecedes(t) updates
manipulates AlwaysPrecedes(t) uses
updates AlwaysPrecedes(t) uses
uses AlwaysFollowedBy(t) manipulates
uses AlwaysFollowedBy(t) updates
```

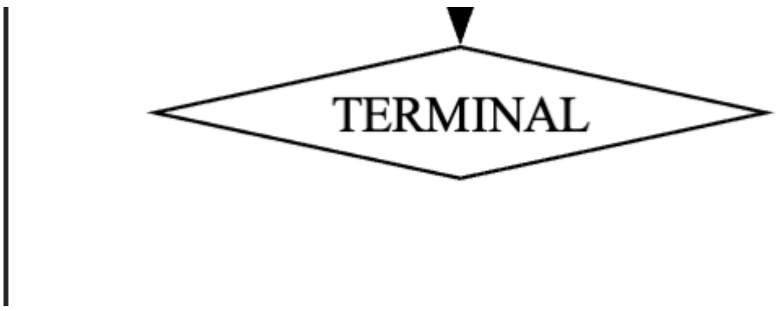
player 2 win

trace 2

```
digraph G {
  0 [label="manipulates"];
  1 [label="manipulates"];
  2 [label="updates"];
  3 [label="updates"];
  4 [label="uses"];
  5 [label="TERMINAL",shape=diamond];
  6 [label="INITIAL",shape=box];
0->0 [label="P: 0.50"];
0->2 [label="P: 0.50"];
1->1 [label="P: 0.85"];
1->3 [label="P: 0.15"];
2->4 [label="P: 1.00"];
3->3 [label="P: 0.71"];
3->4 [label="P: 0.25"];
3->5 [label="P: 0.04"];
4->1 [label="P: 1.00"];
6->0 [label="P: 1.00"];
}
```

```
INITIAL AlwaysFollowedBy(t) manipulates
INITIAL AlwaysFollowedBy(t) updates
INITIAL AlwaysFollowedBy(t) uses
manipulates AlwaysFollowedBy(t) updates
manipulates AlwaysPrecedes(t) updates
manipulates AlwaysPrecedes(t) uses
updates AlwaysPrecedes(t) uses
uses AlwaysFollowedBy(t) manipulates
uses AlwaysFollowedBy(t) updates
```

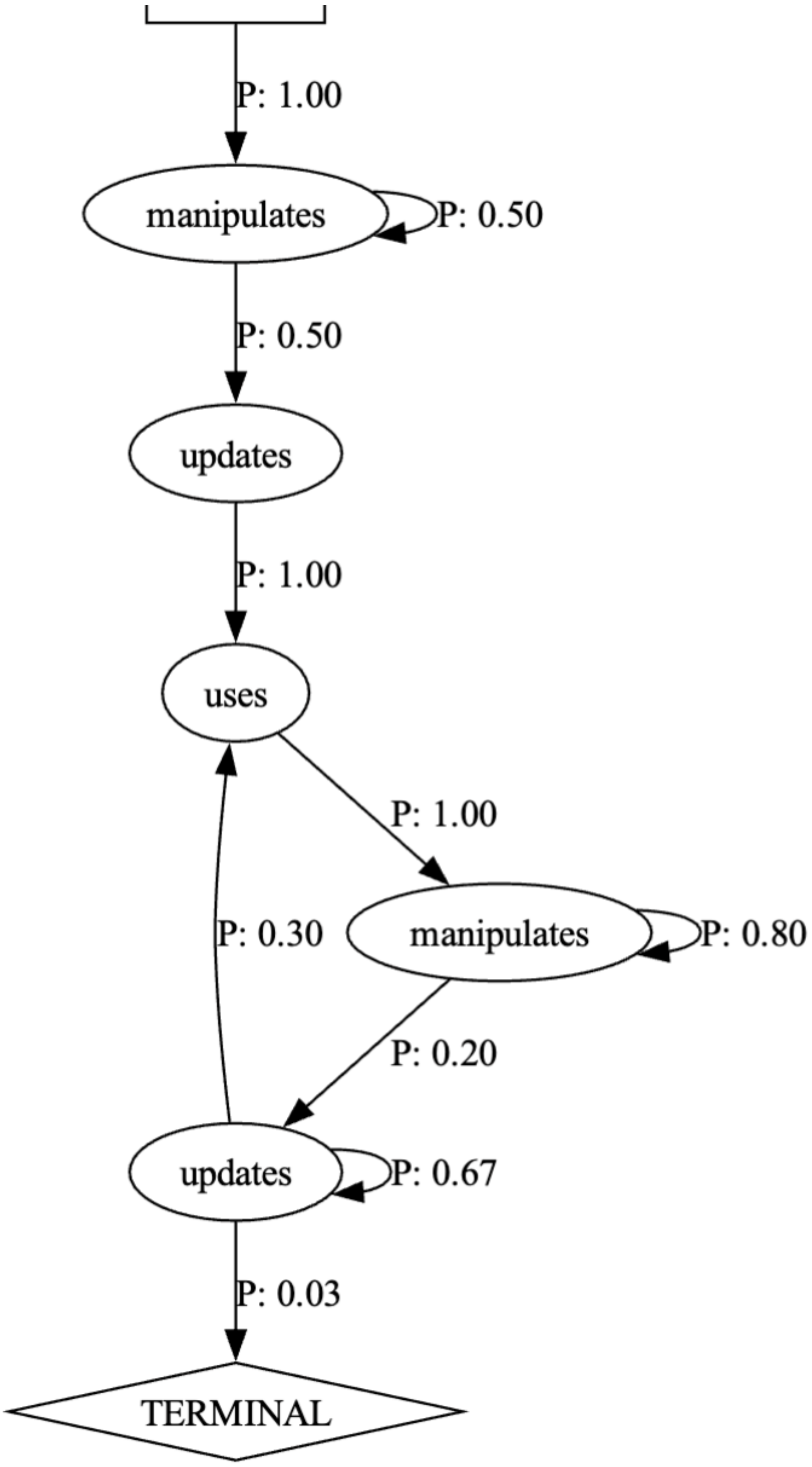

INITIAL
P: 1.00
manipulates
P: 0.50
P: 0.50
updates
P: 1.00
uses
P: 1.00
P: 0.25
manipulates
P: 0.85
P: 0.15
updates
P: 0.71
P: 0.04



tie

```
digraph G {
  0 [label="manipulates"];
  1 [label="manipulates"];
  2 [label="updates"];
  3 [label="updates"];
  4 [label="uses"];
  5 [label="TERMINAL",shape=diamond];
  6 [label="INITIAL",shape=box];
0->0 [label="P: 0.50"];
0->2 [label="P: 0.50"];
1->1 [label="P: 0.80"];
1->3 [label="P: 0.20"];
2->4 [label="P: 1.00"];
3->3 [label="P: 0.67"];
3->4 [label="P: 0.30"];
3->5 [label="P: 0.03"];
4->1 [label="P: 1.00"];
6->0 [label="P: 1.00"];
}
```

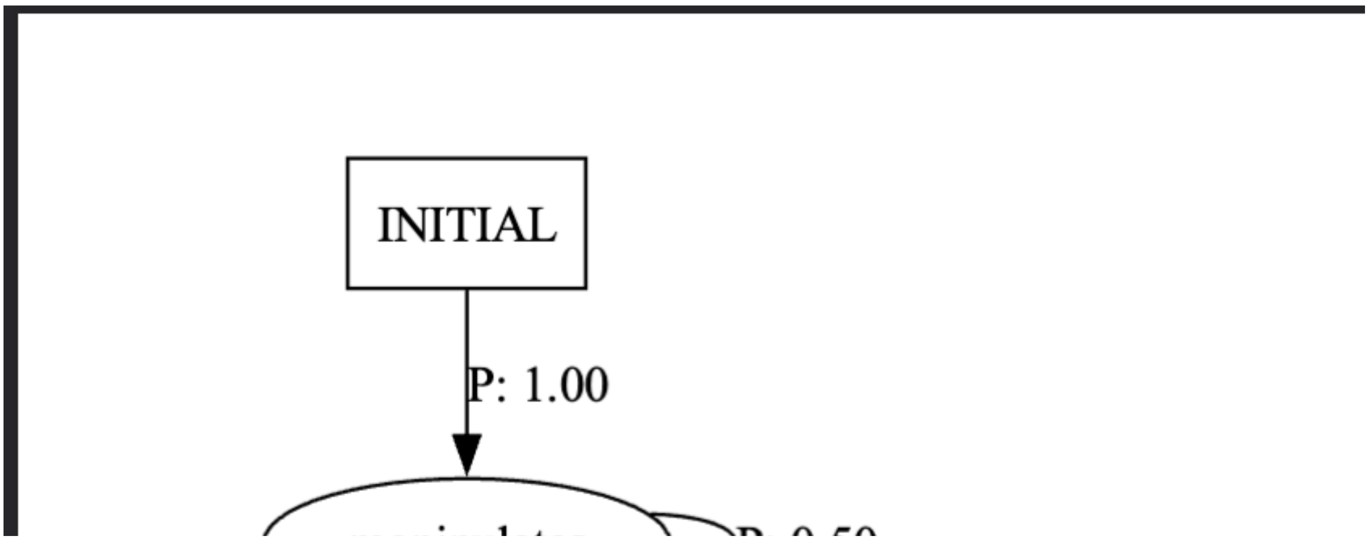
```
INITIAL AlwaysFollowedBy(t) manipulates
INITIAL AlwaysFollowedBy(t) updates
INITIAL AlwaysFollowedBy(t) uses
manipulates AlwaysFollowedBy(t) updates
manipulates AlwaysPrecedes(t) updates
manipulates AlwaysPrecedes(t) uses
updates AlwaysPrecedes(t) uses
uses AlwaysFollowedBy(t) manipulates
uses AlwaysFollowedBy(t) updates
```

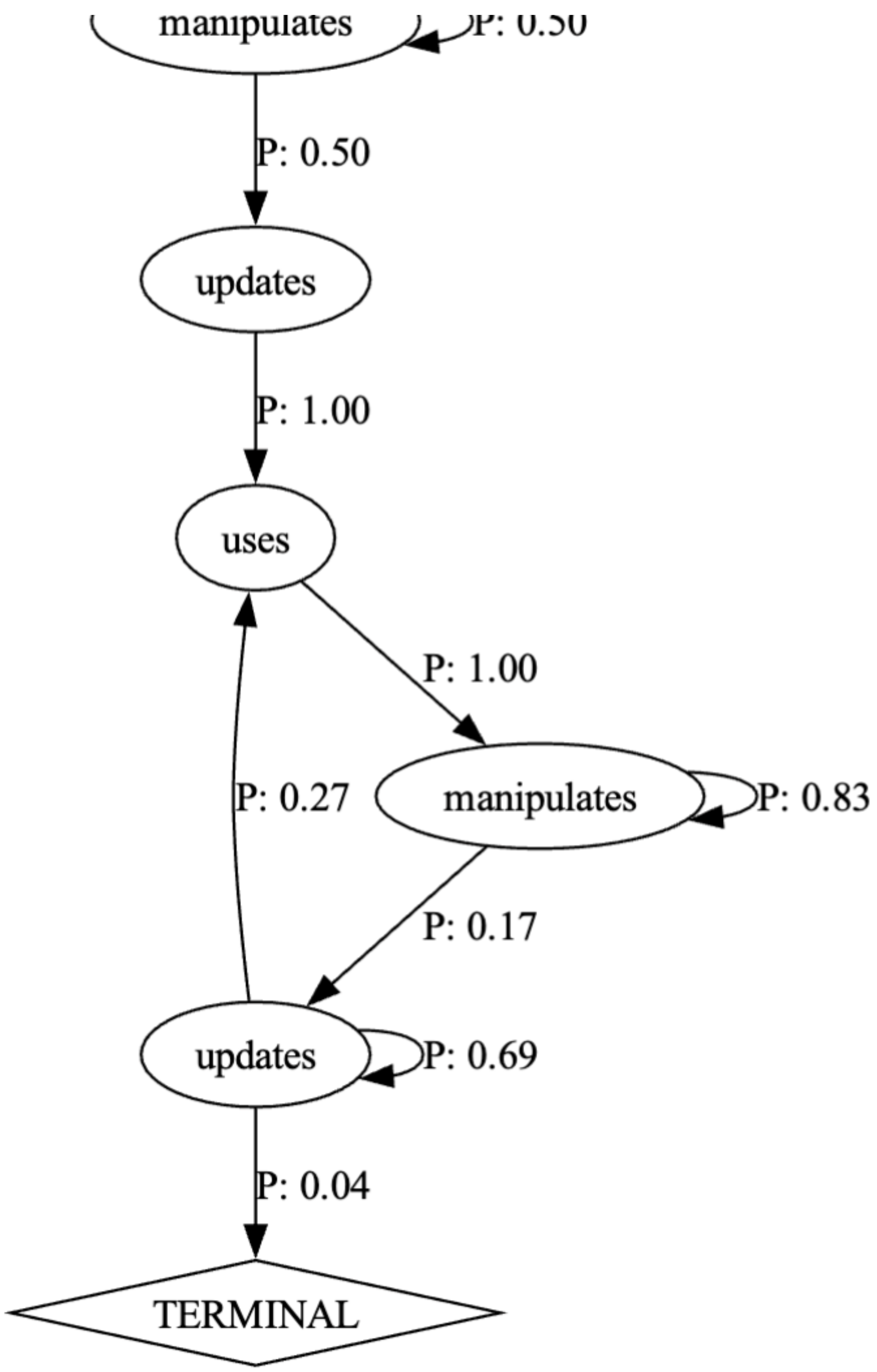

P: 1.00
manipulates
P: 0.50
P: 0.50
updates
P: 1.00
uses
P: 1.00
P: 0.30
manipulates
P: 0.80
P: 0.20
updates
P: 0.67
P: 0.03
TERMINAL

final

```
digraph G {
  0 [label="manipulates"];
  1 [label="manipulates"];
  2 [label="updates"];
  3 [label="updates"];
  4 [label="uses"];
  5 [label="TERMINAL",shape=diamond];
  6 [label="INITIAL",shape=box];
0->0 [label="P: 0.50"];
0->2 [label="P: 0.50"];
1->1 [label="P: 0.83"];
1->3 [label="P: 0.17"];
2->4 [label="P: 1.00"];
3->3 [label="P: 0.69"];
3->4 [label="P: 0.27"];
3->5 [label="P: 0.04"];
4->1 [label="P: 1.00"];
6->0 [label="P: 1.00"];
}
```

```
INITIAL AlwaysFollowedBy(t) manipulates
INITIAL AlwaysFollowedBy(t) updates
INITIAL AlwaysFollowedBy(t) uses
manipulates AlwaysFollowedBy(t) updates
manipulates AlwaysPrecedes(t) updates
manipulates AlwaysPrecedes(t) uses
updates AlwaysPrecedes(t) uses
uses AlwaysFollowedBy(t) manipulates
uses AlwaysFollowedBy(t) updates
```





Report questions:

```
1. For the Synoptic build (https://github.com/ModelInference/synoptic),
identify 2 design principles or best programming practices that are
satisfied. For each design principle or best practice satisfied,
illustrate with an example from the build.
```

Modularity: The repository separates its features into several chunks, including CSight, InvariMint and Perfume, instead of having them all in one class.
Single responsibility: Methods in the repository are all each in responsibility of only one job.

```
2. Briefly explain how you added the tracing statements to the first
version of the RowGameApp (Step 1 above).
```

In this non-MVC implementation of the application, we have tried to split the program on a higher level into model and view - where we only considered the blocks to be part of the view and rest everything is part of the model. Therefore, any text appearing while playing the game in the bottom TextArea is to be considered as part of the model and only the changes in the blocks is considered to be part of the view. Accordingly we added “manipulates” log whenever we were using a setText or setContents, whereas an ‘updates’ log when the updateBlock method was called.

```
3. Briefly explain your technique for sampling the traces of the first
version of the RowGameApp (Step 2 above). This explanation should include
how you decided to stop sampling.
```

Our strategy was to get as many traces as possible under different conditions. So we started with traces which guaranteed that Player 1 will win - under this condition we sampled many traces which were of varying length (meaning sometimes player 1 wins within 5 moves which is the fastest one could win in the game, and for others this number of moves before win increased). This same strategy was applied for the case where Player 2 will win. Lastly we sampled traces for the case where there is a draw between the players and the game finishes. We observed that the Finite State Automata (FSA) model that was being generated was of the same structure for all these cases, and this is the point where we stopped sampling traces (We also got an exceptional trace, which was generated when we overwrite the moves of other players - more details of this case are present in next answer).

```
4. For your inferred model of the first version of the RowGameApp, briefly
describe one expected behavior. Briefly describe one unexpected behavior.
```

One expected behavior (conditioned on our definition of logs) is that for each move there is a cycle of 3 logs - uses, manipulates and updates (in the exact order mentioned). This group keeps getting repeated until the game is terminated (no one wins/any one player wins) - which leads to a manipulates statement at the very end (which makes changes in the TextArea specifying which player won/ no one won and the game ended in a draw).

The unexpected behavior was observed when we tried to click again on a block which has a move (displays ‘X’ or ‘O’) - allowed to overwrite other player’s moves or even skip a chance by clicking again on a cell you have marked before. In this case an additional edge is placed in the FSA, which brings it from the manipulates state (present when we want to display game is over) back to ‘uses’ state.

```
5. Was it easier to add the tracing statements to the second version of
the RowGameApp (Step 1 above). Why or why not?
```

The second version of the App is definitely easier to add statements. Since version 2 is DRYer (less repeat lines) than version 1 using MVC architecture. Instead of checking behavior in every if statement in the TicTacToeGame.java in version 1, in version 2 we could just jump to the model/ and view/ folders and add statements in the desired functions.

```
6. Your inferred model of the second version of the RowGameApp should be
the MVC architecture pattern. Did you need to change your technique for
sampling the traces of the RowGameApp (Step 2 above) to produce that
inferred model? Briefly explain why or why not.
```

We stay using the same sampling strategy. The sampling strategy focuses on capturing all the possible outcomes: player1 win, player2 win and tie, which is the same for version 1 and 2 . It is independent of the internal design pattern.

```
7. Propose a semi-automated of fully automated technique for sampling the
traces of the RowGameApp (Step 2 above).
```

A fully automated way that in expectation guarantees to go through all the cases can go as follows - \

- Create a list of all possible blocks we can make a move on.
- At each action step, randomly sample a block from this list.
- Make the move and remove the state from the list.
- This entire process (Steps 2-3) continues until the program ends.

This technique will work for generating various lengths and forms of traces for the RowGameApp application. (However this will not cover the unexpected behavior case - which can be modeled by removing 2nd part of Step 3 - remove the selected state from the list)

```
8. What is the primary disadvantage of any such trace sampling technique?
```

As described earlier these fully automated techniques are not suitable for getting traces of the unexpected behavior. We have to make modifications in them, just to get traces for unexpected behavior. Moreover, there is no guarantee in how many iterations it will be able to generate all the various forms of traces required for full satisfaction.

```
9. Propose one technique for illustrating the differences between two
inferred models (represented as FSAs).
```

The inferred models are stored in the forms of a graph containing 2 components - Labeled states (state number associated with the log statement) and labeled transitions (transitions with probability associated with it). We can directly compare 4 aspects if we are getting differences between 2 inferred models:\

- Number of states
- Number of transitions (edges)
- Type of each state
- Probability of transitions

We can use these 4 aspects to illustrate differences between 2 different models.

```
10.   That is one use case where model inference would be helpful to the
developers?
```

In a complex system, debugging by manually scanning through all the logs isn't practical. Model Inference is useful in identifying unexpected patterns in a system during development. Like the question 4 above, without the model inference, developers have to monitor all the output logs to identify which part in the program is the cause of the unexpected behavior, while using model inference this process becomes markedly more effective and time-saving.