Graphical models for QuickCheck

Thomas Arts, Kirill Bogdanov, Alex Gerdes, John Hughes

This project has received funding from the
EU FP7 Collaborative project *PROWESS*, grant number 317820,
http://www.prowessproject.eu

Testing with QuickCheck

- QuickCheck permits one to write generators for test data and pre/postconditions.

- The expectation is that user provides a model, based on which test data is randomly generated.

- Illustration of testing a *write* operation:

```
write_args(_) -> [key(), int()].

write(Key, Value) -> lock:write(Key, Value).

write_post(_,[Key,Value],Res) -> eq(Res,ok).
```

Global state

operation name

this is a precondition

type of the global state

element of the global state

```
write_pre(S)  -> S#state.started

write_args(S) -> [  key(), int()  ].
```

precondition

generator for arguments

list of arguments to pass to operation *write* of the system under test

returns a generator for integers

returns a generator for keys

Global state is a record-type of type *state* with element *started*, passed as an argument to all operations.

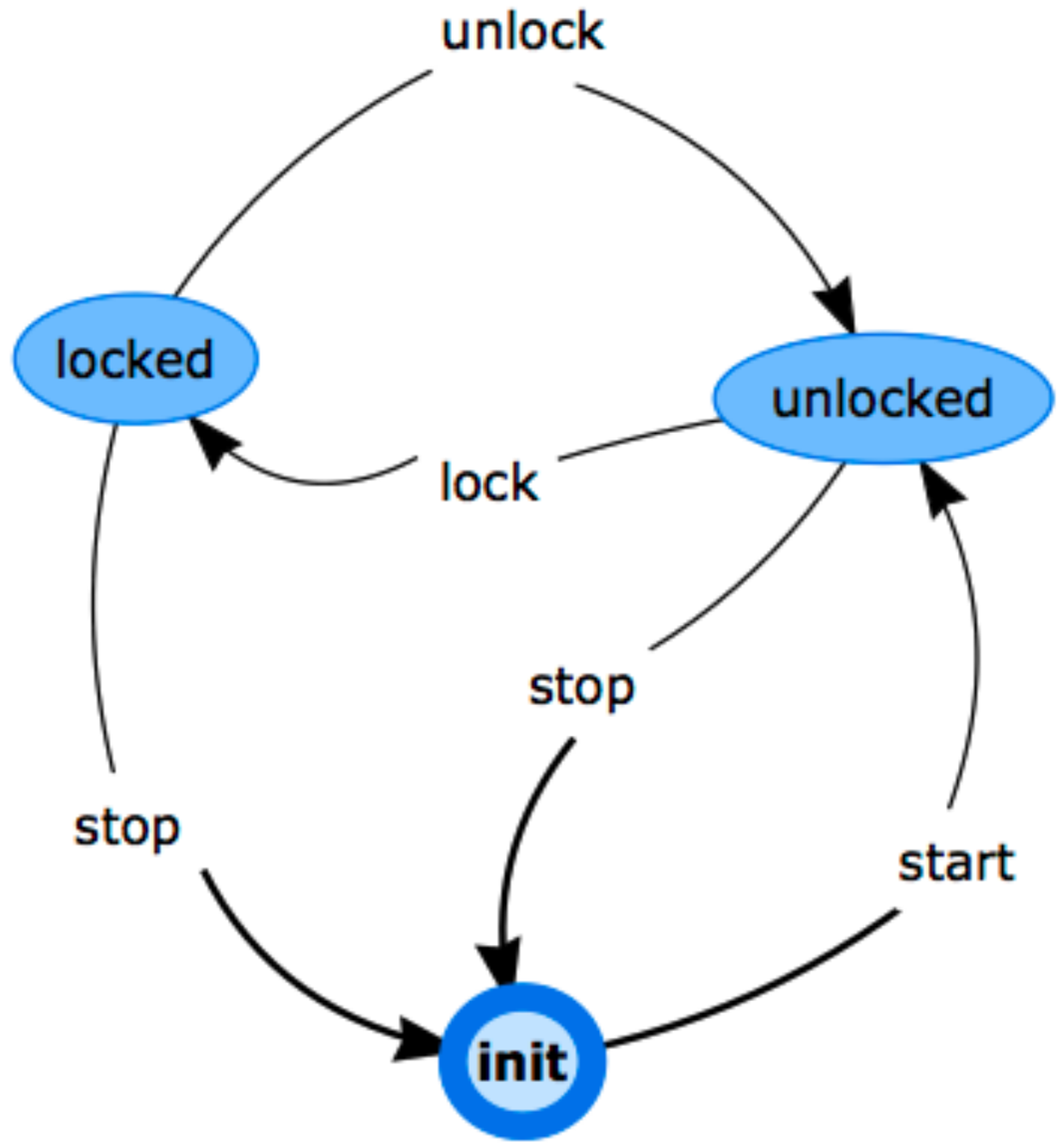
Testing *write* using global state

Assuming *started* is a boolean component of the global state reflecting if the system was started,

```
write_args(S) -> [key(), int()].

write(Key, Value) -> lock:write(Key, Value).

write_pre(S)  -> S#state.started

write_post(S,[Key,Value],Res) -> eq(Res,ok).

write_next(S, Res, [Key, Value]) ->
    S#state{kvs = [{Key,Value} |
        proplists:delete(Key,S#state.kvs)]}.
```

Locker example

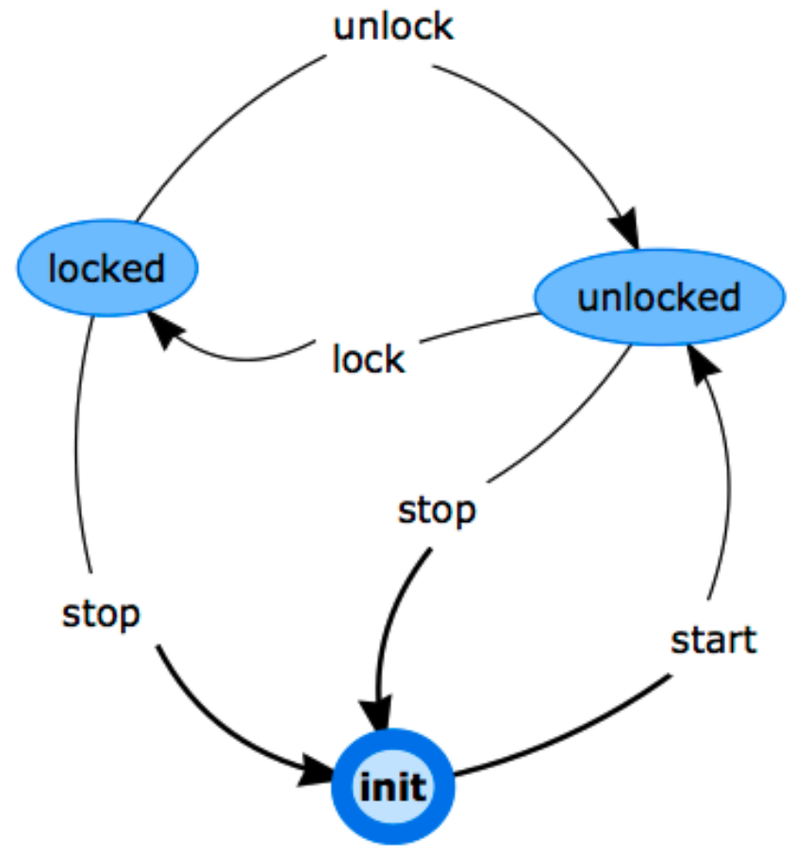
- Can be started/stopped
- Can be locked/unlocked
- Does not include read/write



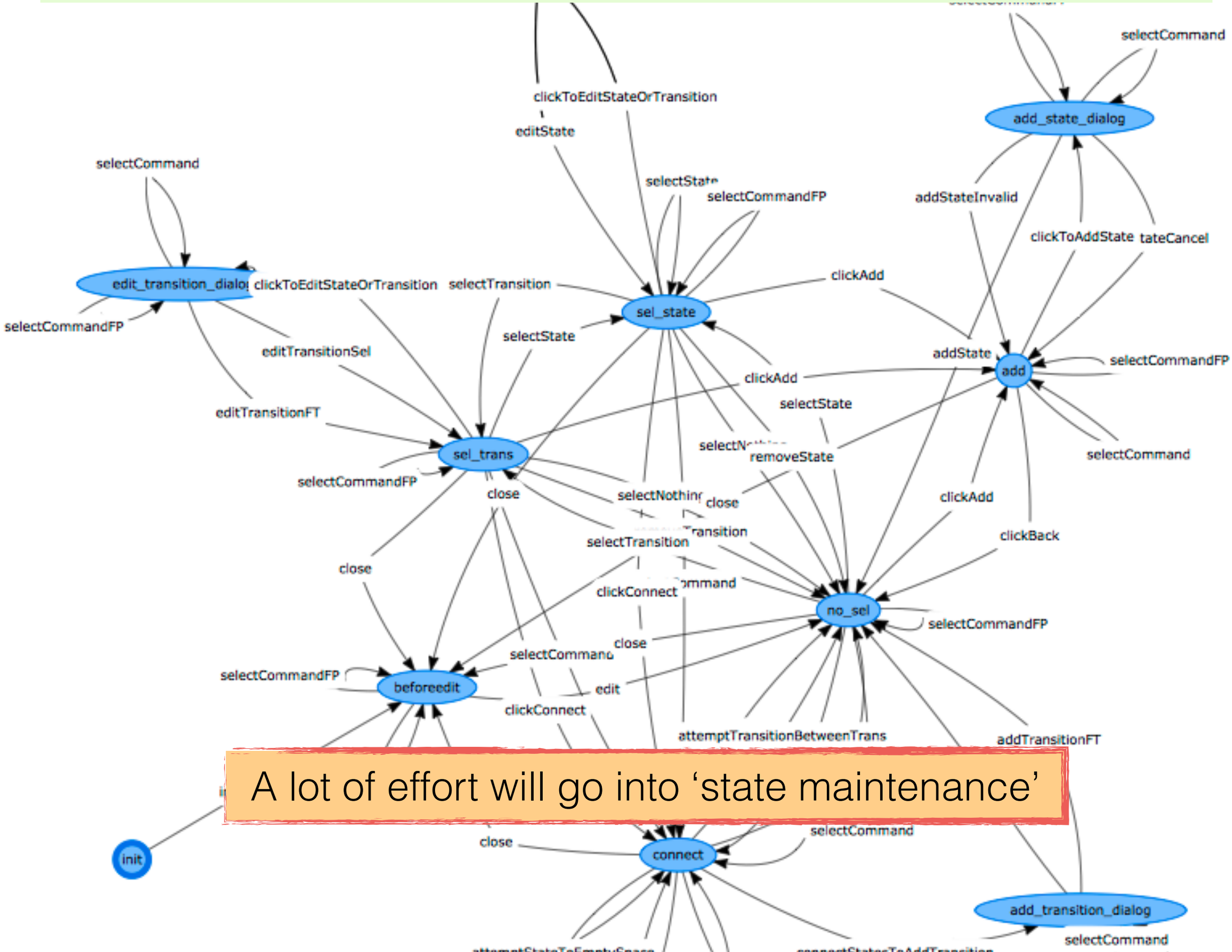
Part of this diagram in pure QuickCheck

```
lock_pre(S) -> S#state.started andalso not S#state.locked.
lock_args(S) -> [].
lock_next(S,Res,[])-> S#state{locked=true}.

unlock_pre(S) -> S#state.started andalso S#state.locked.
unlock_args(S) -> [].
unlock_next(S,Res,[])-> S#state{locked=false}.
```

Very easy to make a mistake in one of the above expressions



Now if we are doing something more complex

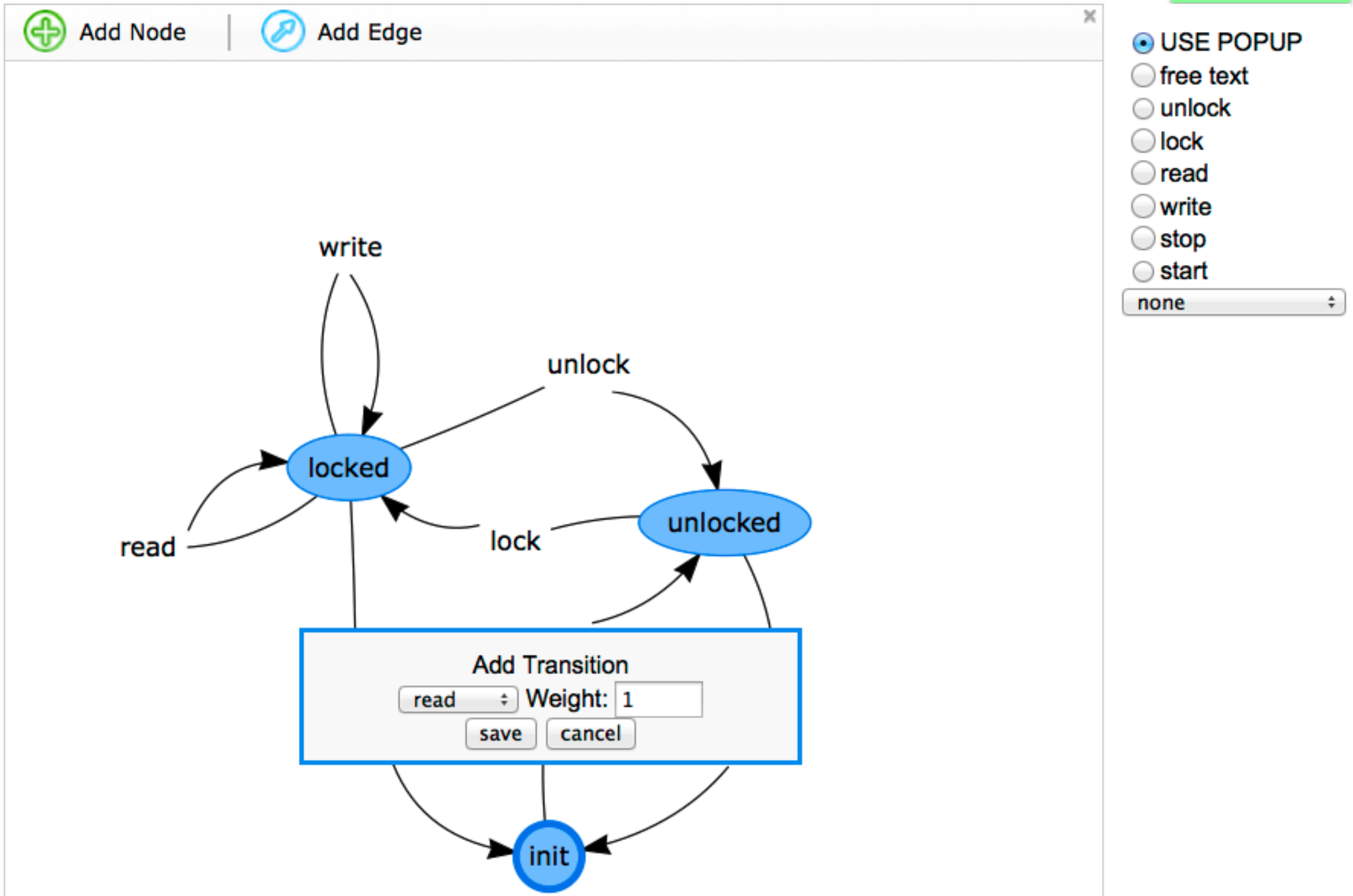


What we did

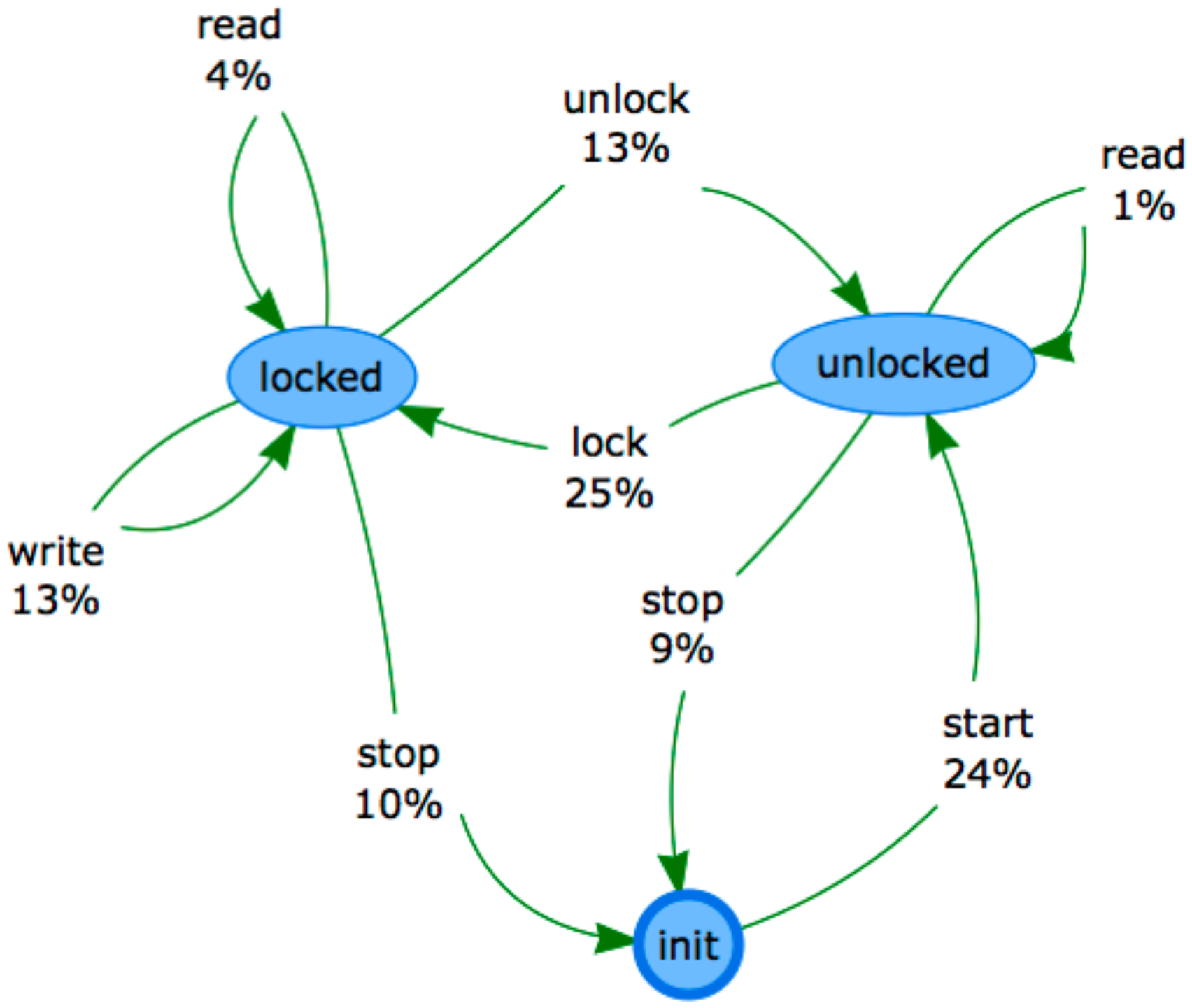
- Developed a tool to edit graphical models.
- Names of operations are extracted from Erlang code.
- For the above example, the resulting model is half the size of the traditional model …

  … and much easier to maintain.
- Test failures and frequencies are automatically extracted from results of test execution.

Addition of a *read* transition around *unlocked*.

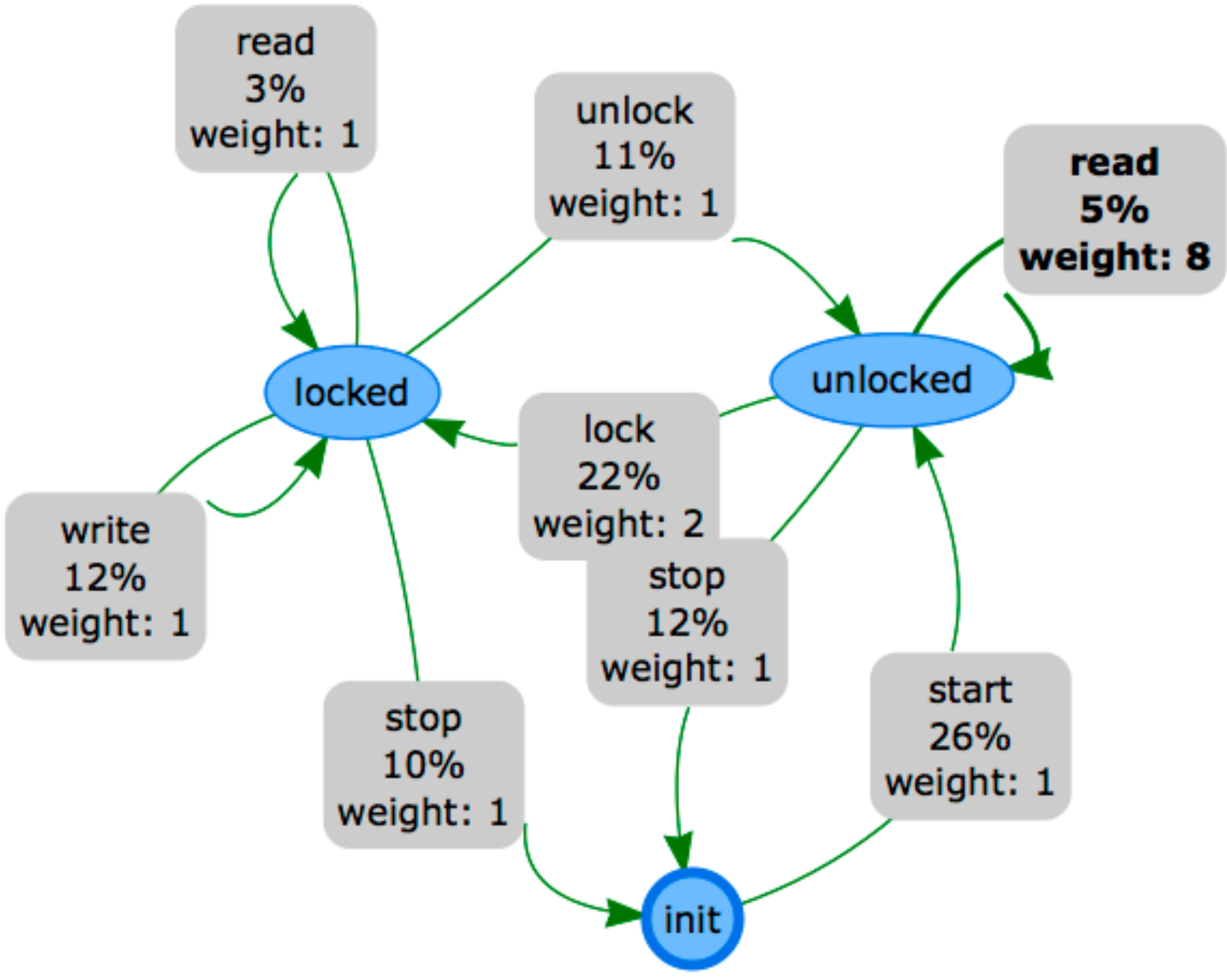


Frequencies



Running tests produces a distribution of transitions

Weights can be updated



Changing weights makes operations of interest run more frequently.

If you would like to try it

- You have access to both QuickCheck tool and the graphical editor online at http://quviq.de/euc2015

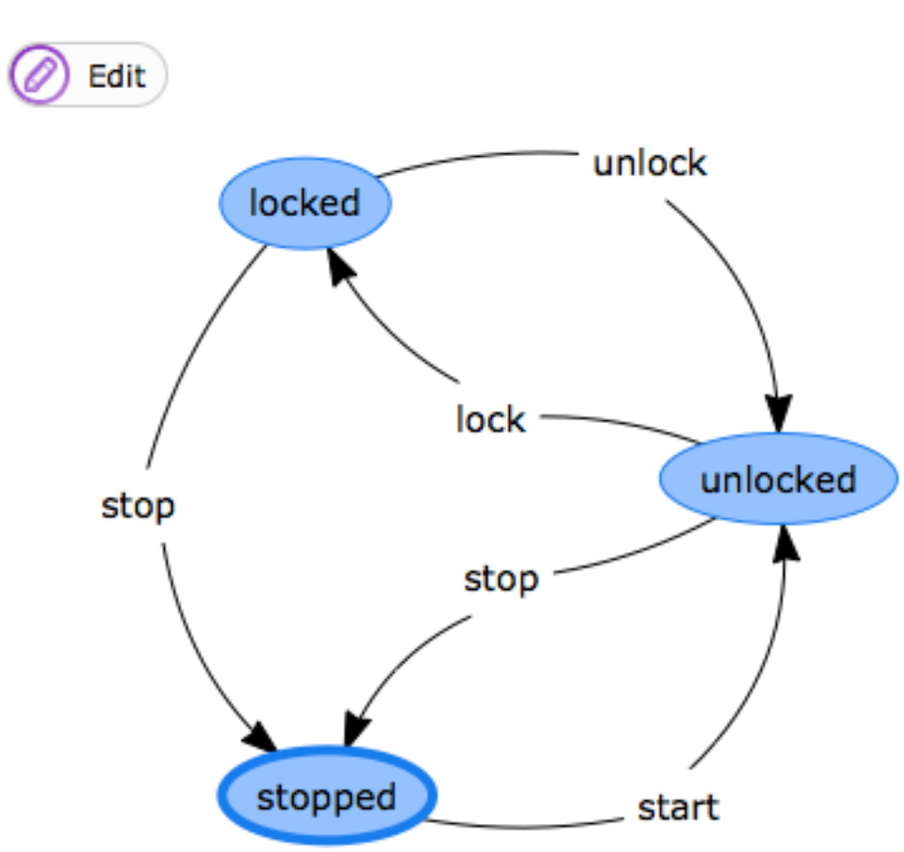
- The .zip file contains both eqc_graphedit (the graphical editing tool) and time-limited version of QuickCheck that you need to install first.

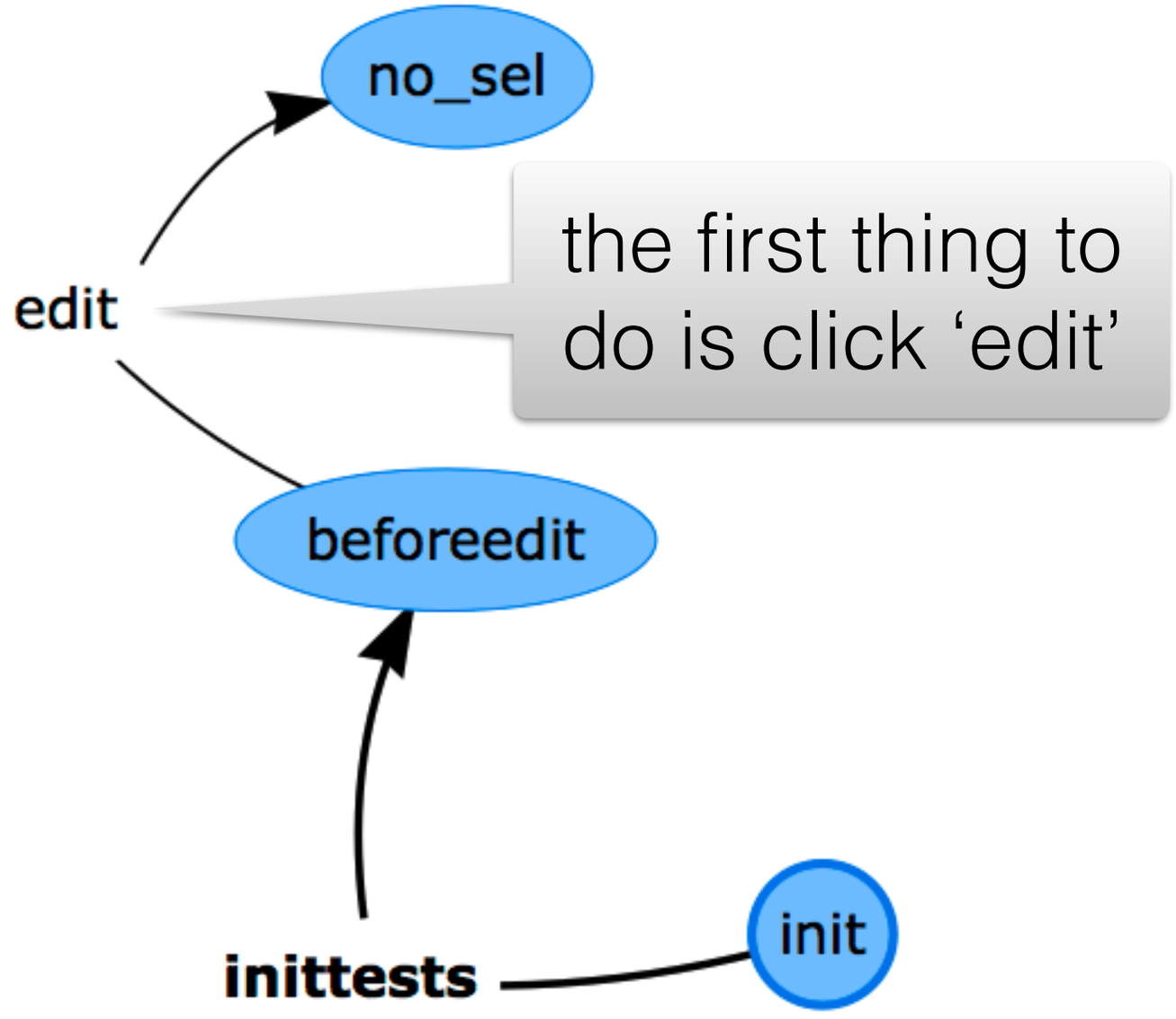
- lock.erl is the module we are testing

- lock_eqc.erl is the QuickCheck model for testing lock.erl

- I'll do the demo how to use the editor.

How to model the graphical editor using itself

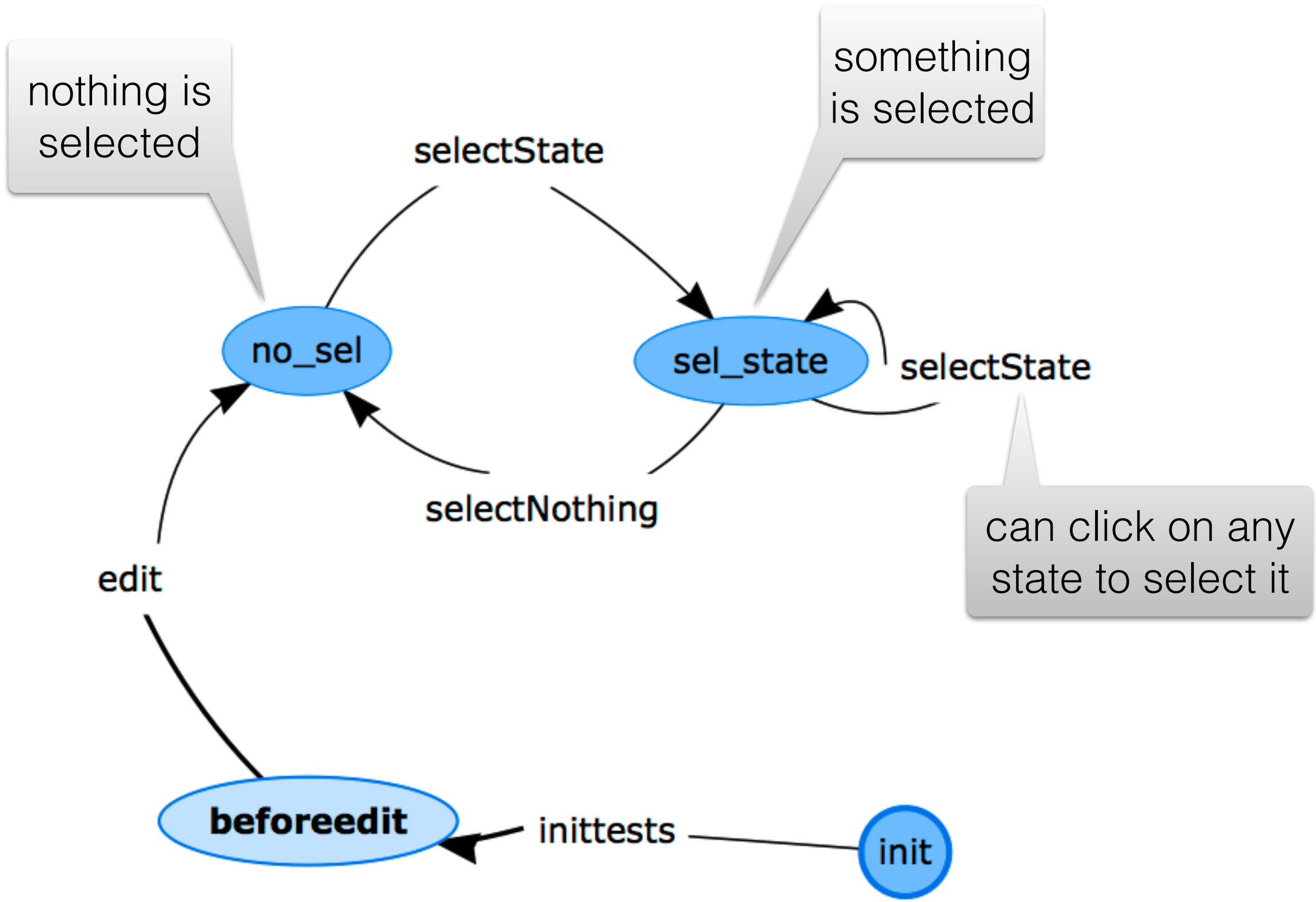




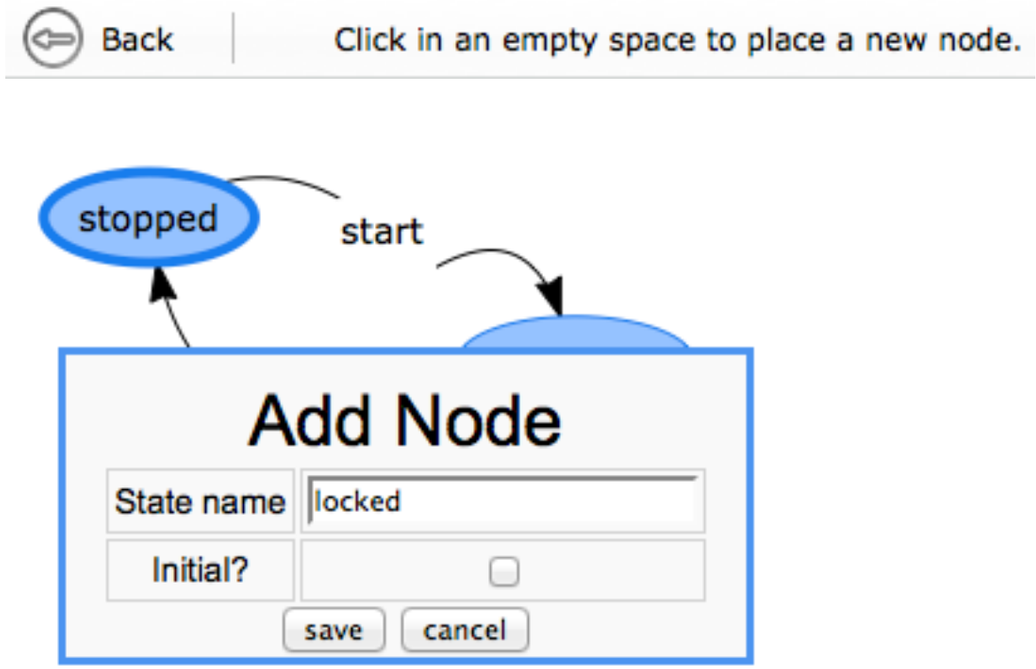
the first thing to do is click ‘edit’

we need to start with a state machine, hence generate one at random

Selecting states

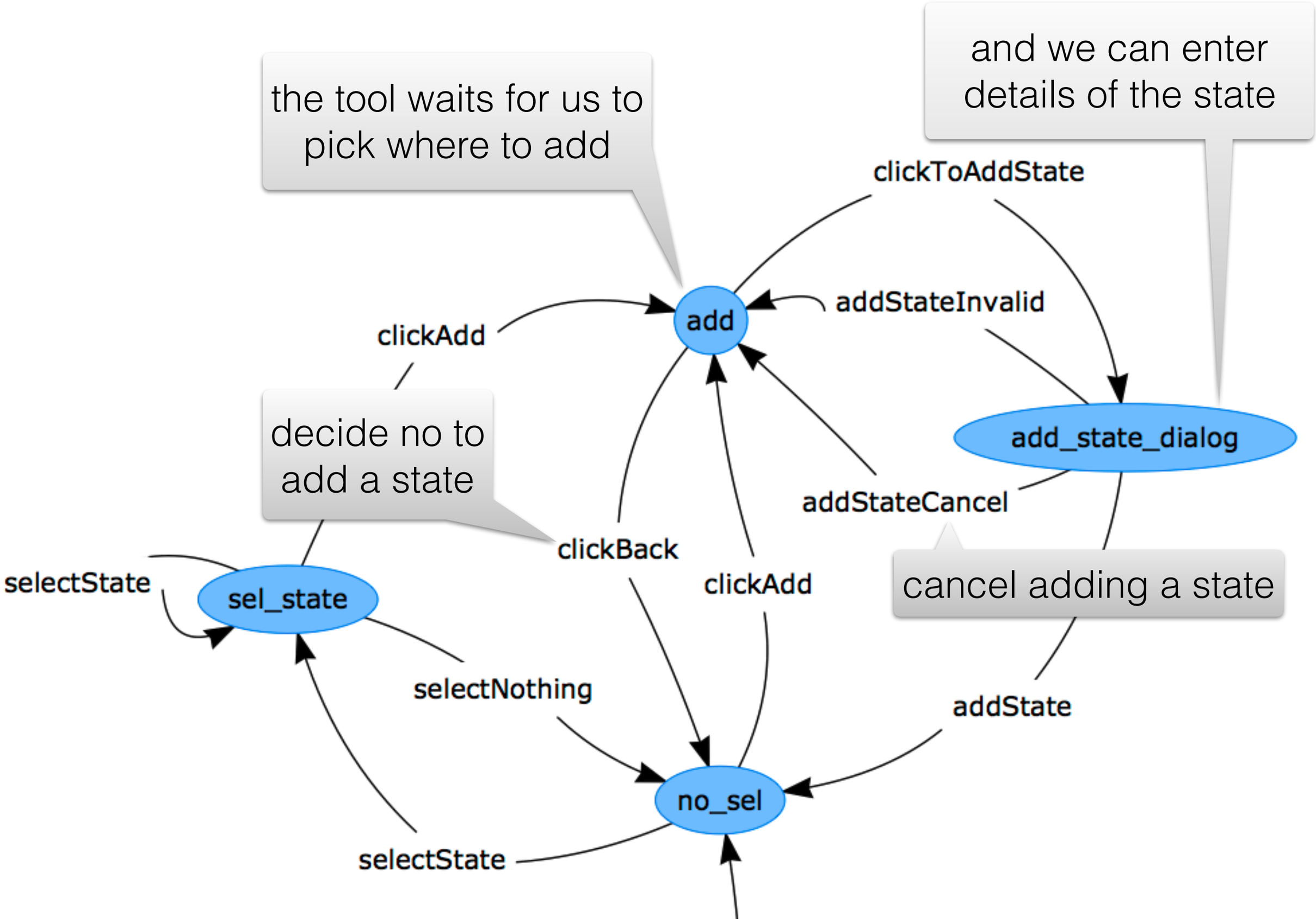


Example: adding the ‘lock’ state



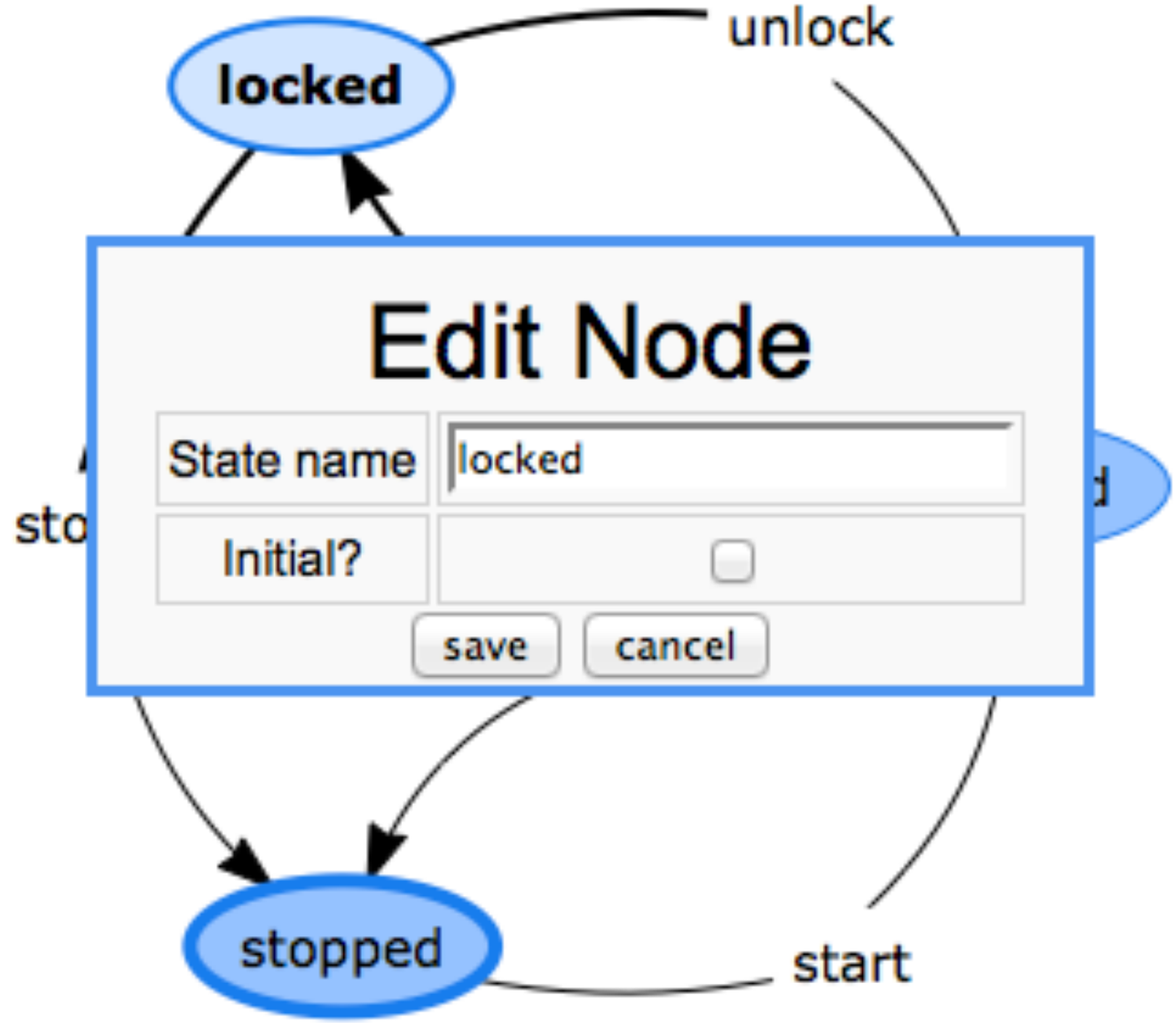
Adding states

Demo running tests

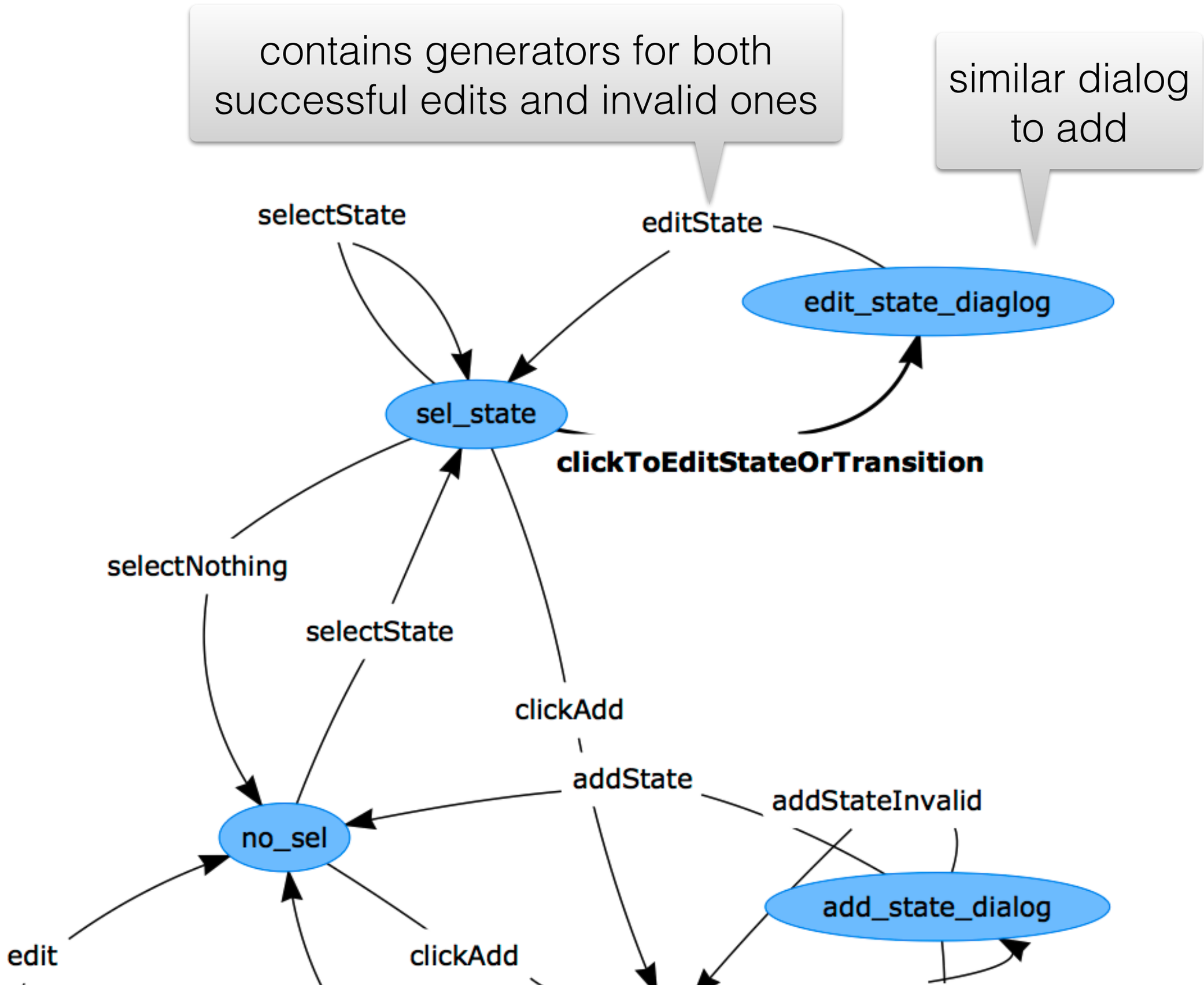


Editing states

- State has to be selected.
- Edit has to be clicked.
- If ‘initial?’ is ticked it cannot be cleared.

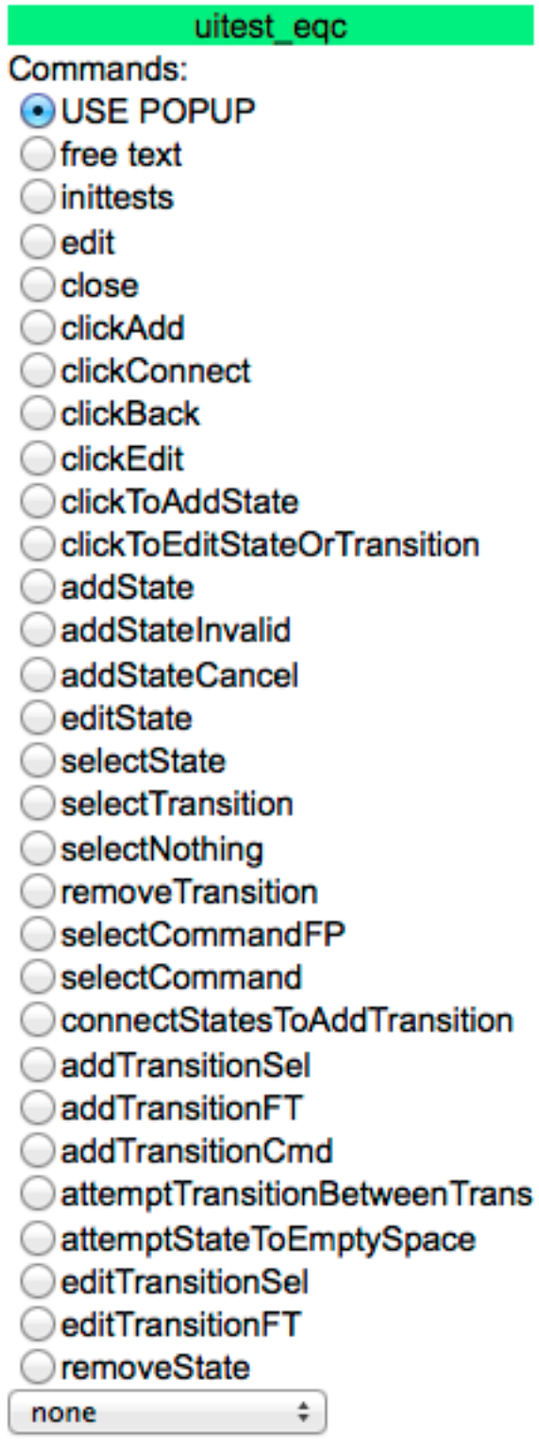


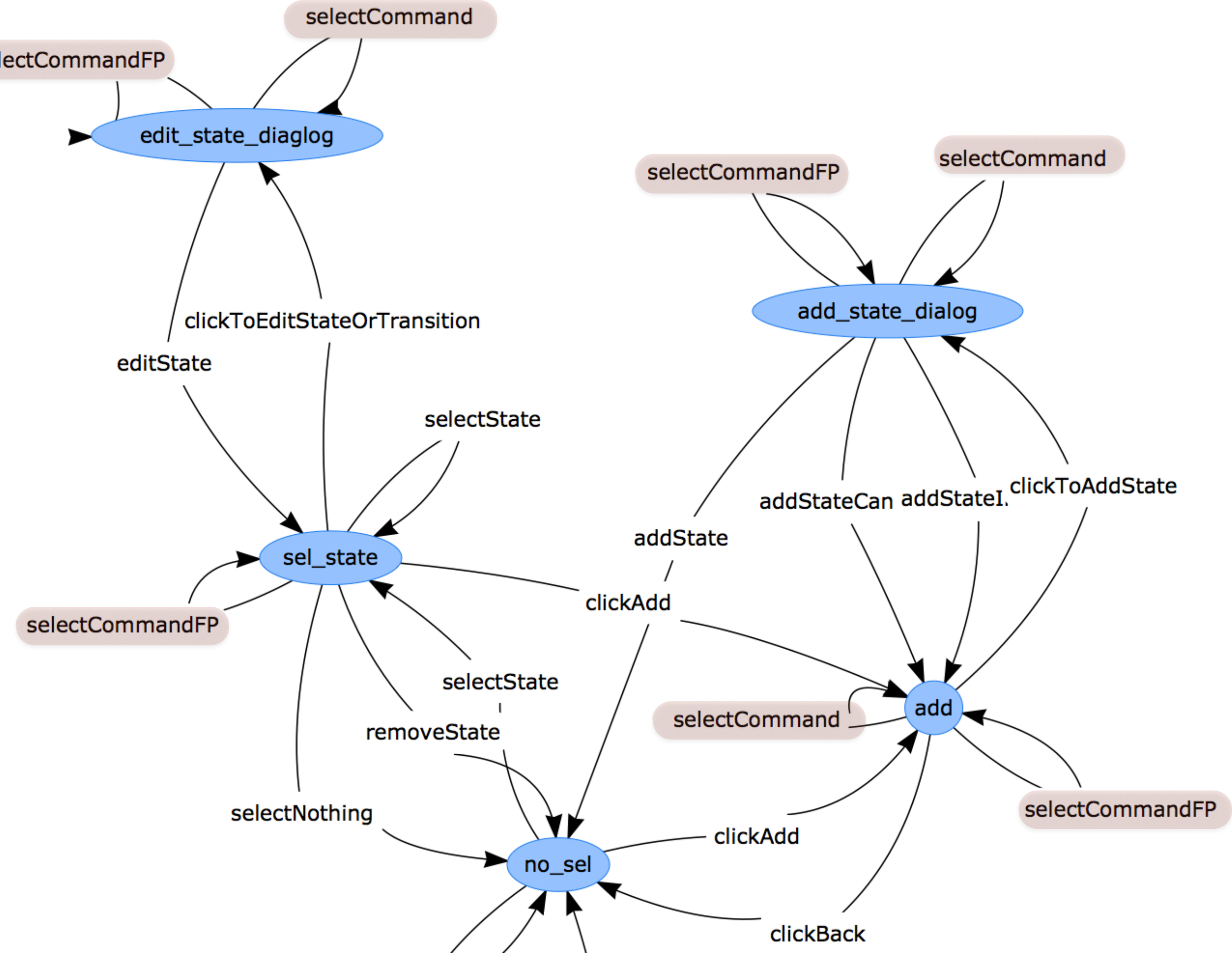
Editing states



Commands on the right-hand side

- Used to add transitions: choose a command, then drag a transition.
- Can be clicked at any time.
- Consequently, the corresponding transitions have to be added to each state.




selectCommand
selectCommandFP
edit_state_diaglog
clickToEditStateOrTransition
editState
selectState
sel_state
selectCommandFP
selectCommandFP
selectCommand
add_state_dialog
addStateCan
addStateI.
clickToAddState
addState
clickAdd
selectState
removeState
selectCommand
add
selectCommandFP
selectNothing
clickAdd
no_sel
clickBack

Conclusions

- Existing QuickCheck models are hard to develop for complex state-transition diagrams.
- Developed interface to edit such diagrams.
- Part of the most recent version of QuickCheck.
- Tested using itself.