Reverse engineering - traces to state machines

Neil Walkinshaw and Kirill Bogdanov [1]

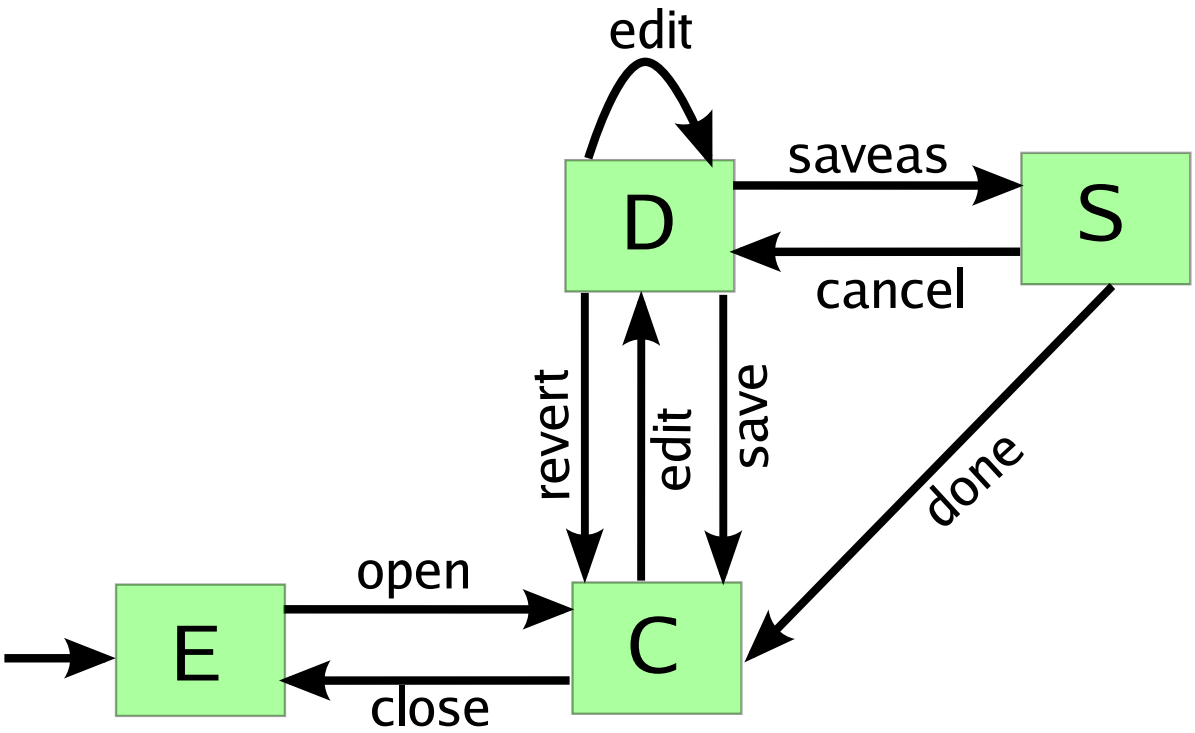
[1]Department of Computer Science
The University of Sheffield

TAIC PART, September 4, 2010

Outline

State-based models are useful

- For understanding software,
- Model-checking,
- Test generation.



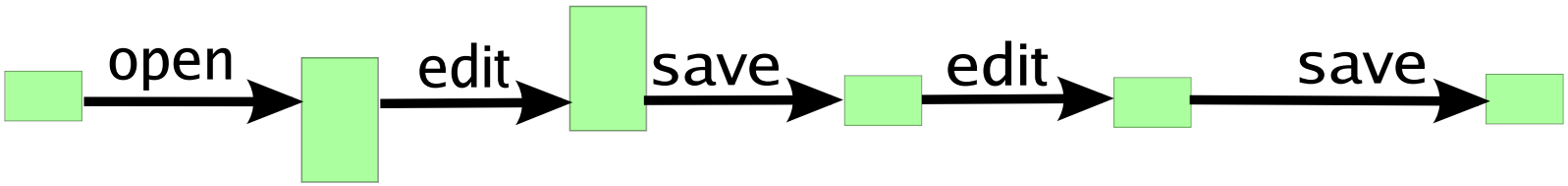
Maintenance can be difficult

- Legacy software tends to have no models associated with it,
- A failing test could indicate a fault in a model,
- Requirements-level defects have to be corrected in both.

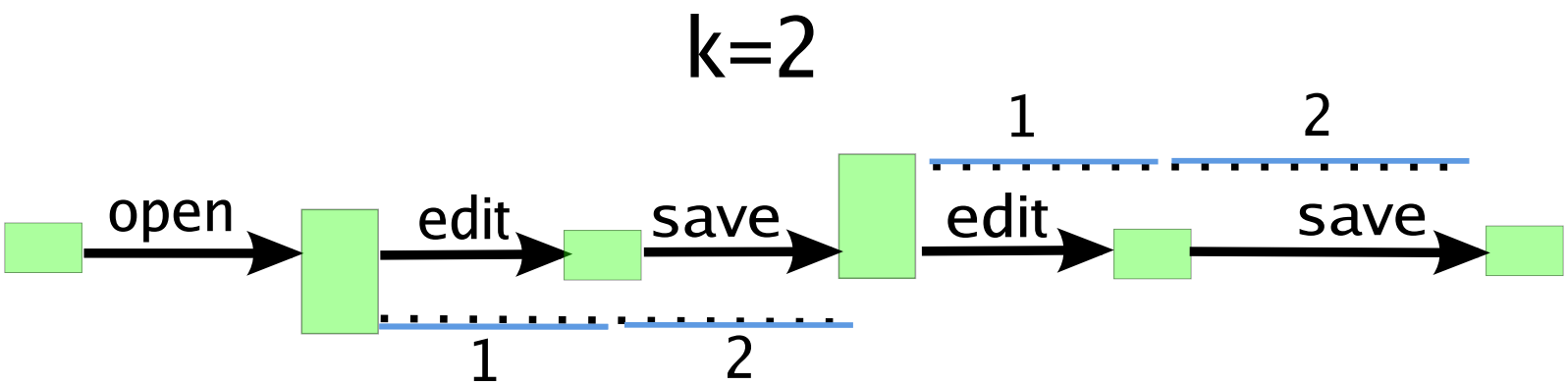
Grammar inference

- Assuming we know how to interpret traces from a program as sequences of events,
- and we know the overall pattern a model should obey (such as recognise a regular language)

The task is to learn models from event traces.

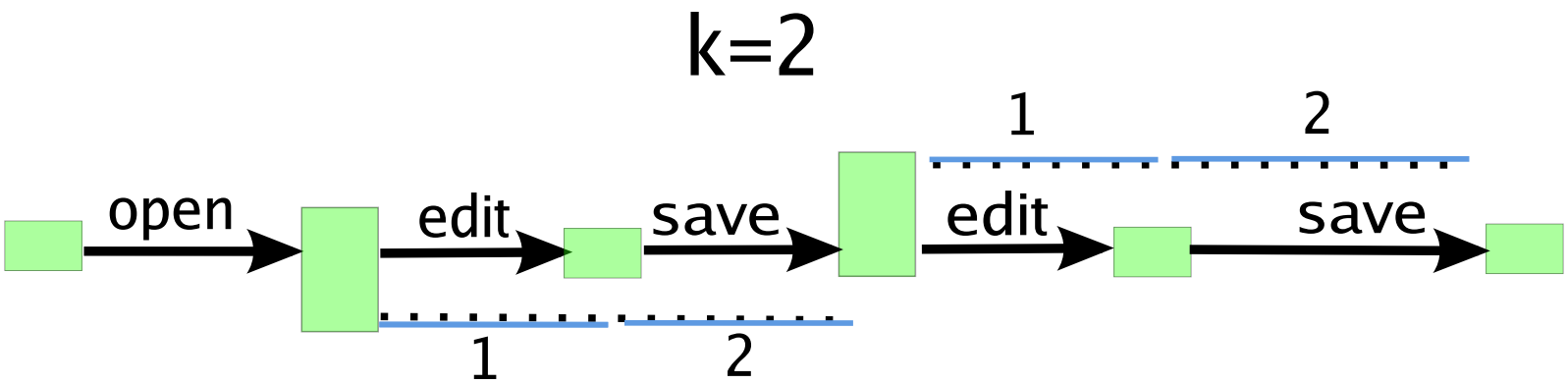


k-tails learner



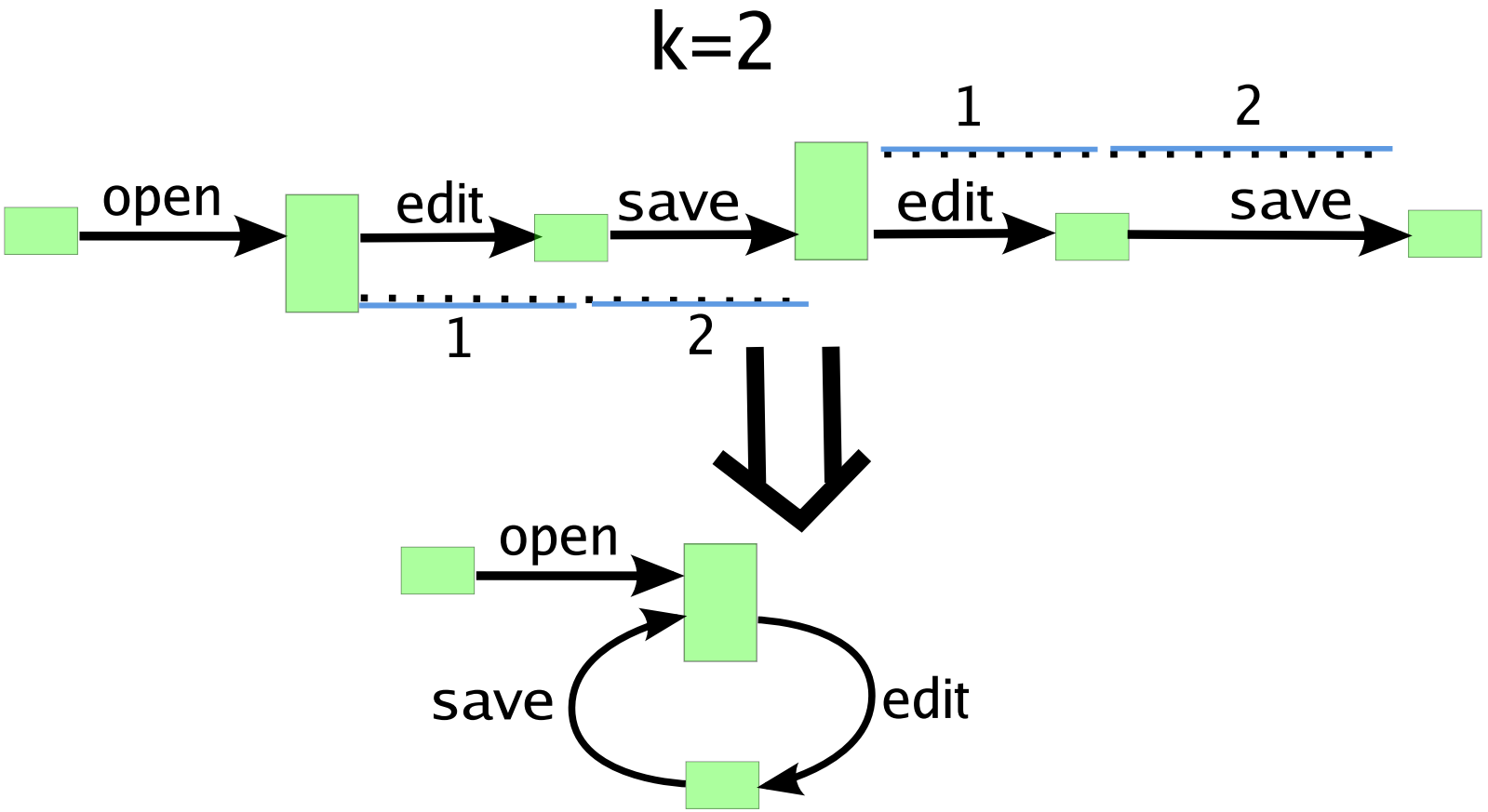
- Take traces and hypothesise what other traces should be possible or not . . .
- . . . assuming that some states in traces correspond to the same state in the model.
- k-tails assumes that if suffixes of length k are the same, so are the states.

k-tails learner

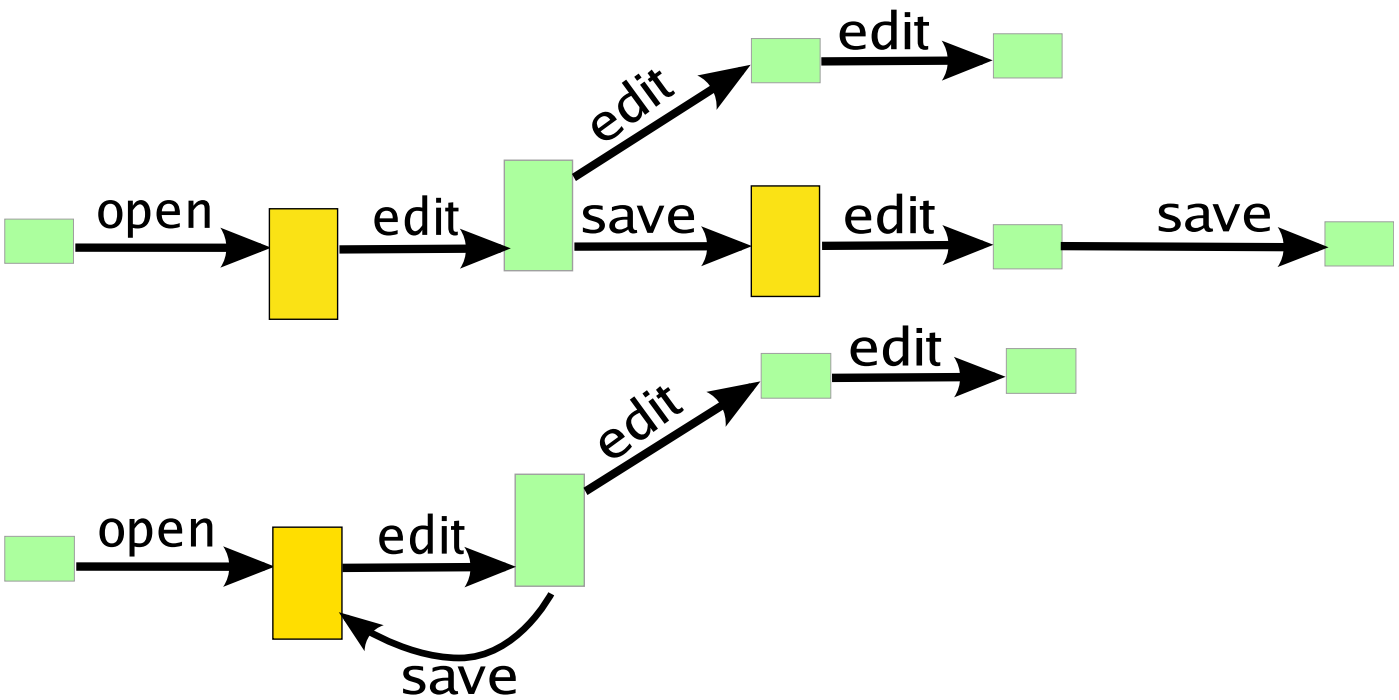


- Take traces and hypothesise what other traces should be possible or not . . .
- . . . assuming that some states in traces correspond to the same state in the model.
- k-tails assumes that if suffixes of length k are the same, so are the states.

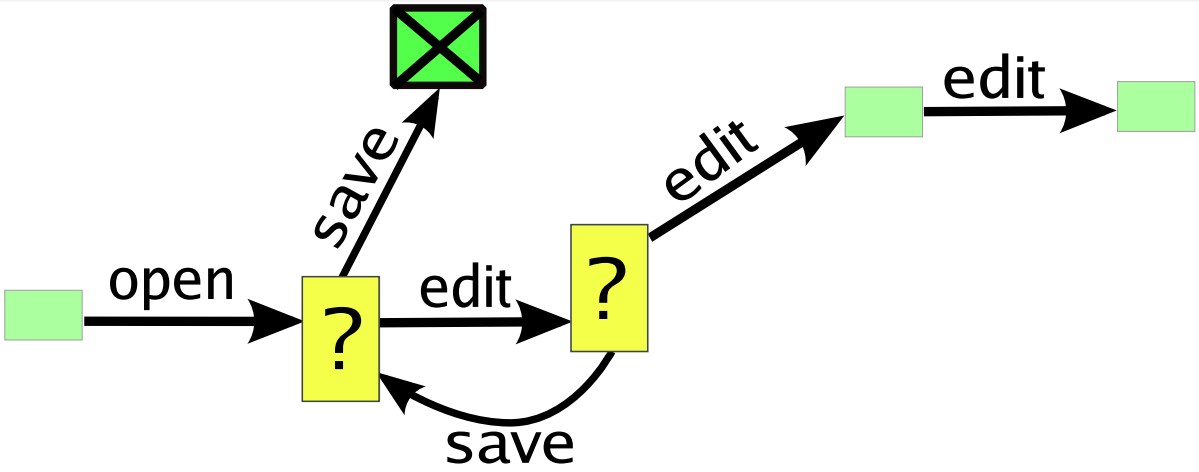
k-tails learner



- A lot of work was done by the Grammar Inference community on passive learners - no feedback from a user.
- If the initial PTA has "enough" positive and negative sequences, the correct FSM will be learnt.



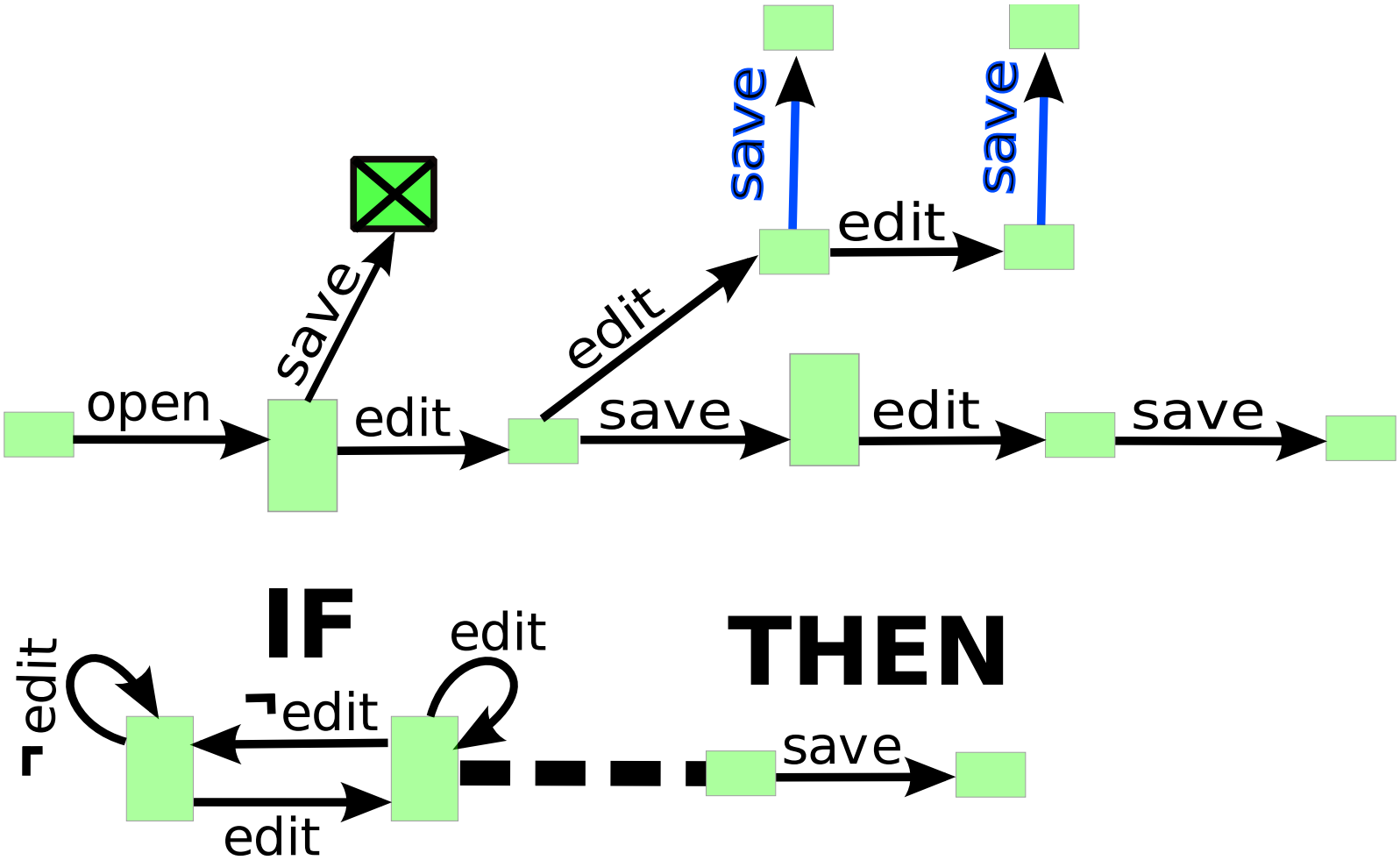
- Starting from the initial node, pairs of states are considered and merged in the order of their *compatibility score*
- An outcome of merging has to be validated - there is a new path $\langle$`open`, `edit`, `save`, `edit`, `edit`$\rangle$ which is not in the original tree.



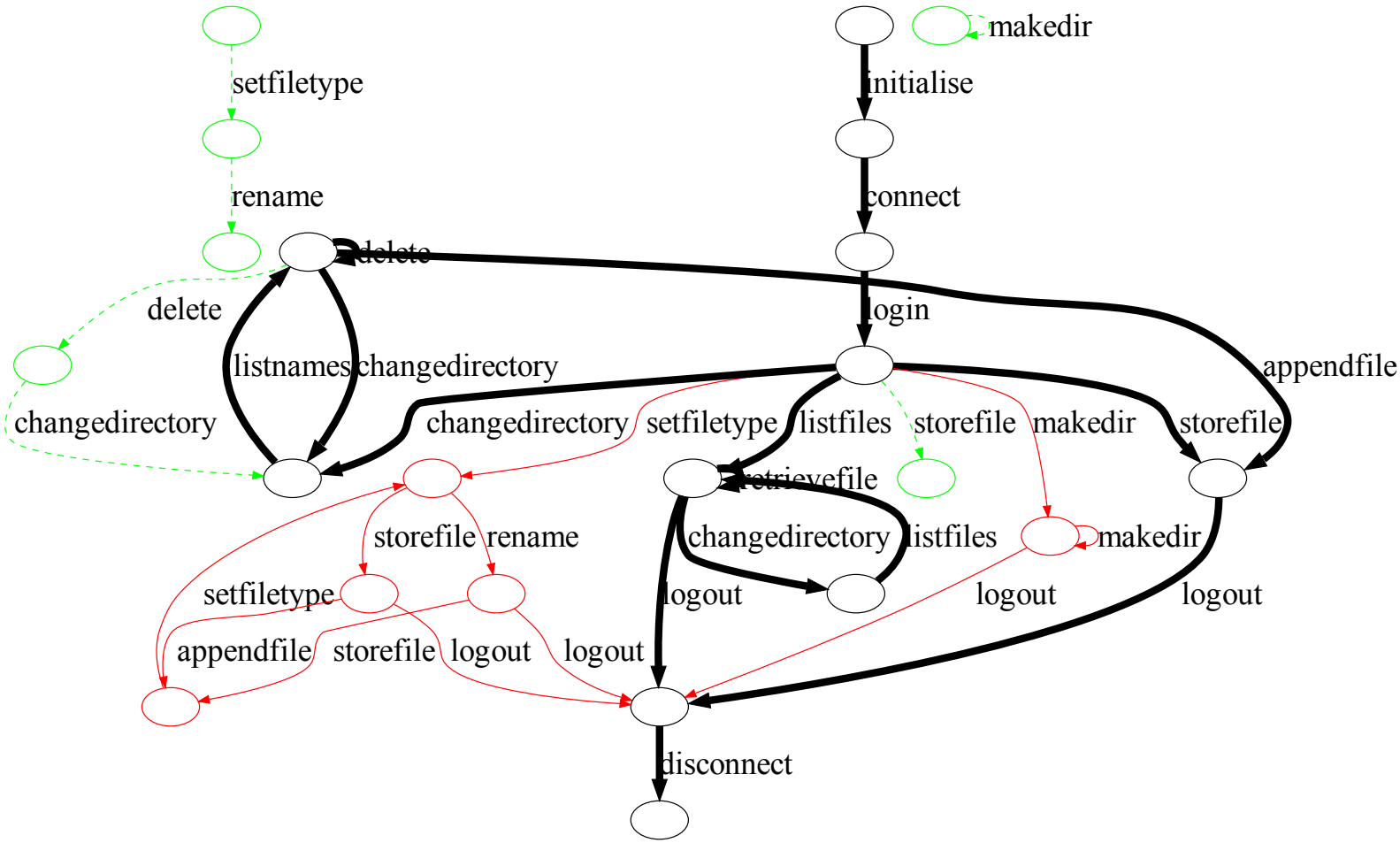
- Since dynamic analysis does not give "enough" traces, feedback is used to validate mergers.
- The two marked states cannot be merged - if a learner attempts to merge them, a user will say that $\langle \texttt{open}, \texttt{save} \rangle$ cannot be performed, hence a *reject*-node is added.
- Experimental results: if we always merge states with a high score (such as 3), we can get 10x reduction in the number of questions and around 10% reduction in the quality of the learnt machine.

- Questions can be executed on a system, checked using static analysis or presented as questions to a developer.
- State merging performs no systematic exploration.
- In order to make analysis more complete, static analysis can be used to compute an underapproximation on infeasible paths, hence a better-quality tree without extra queries.

IF-THEN properties



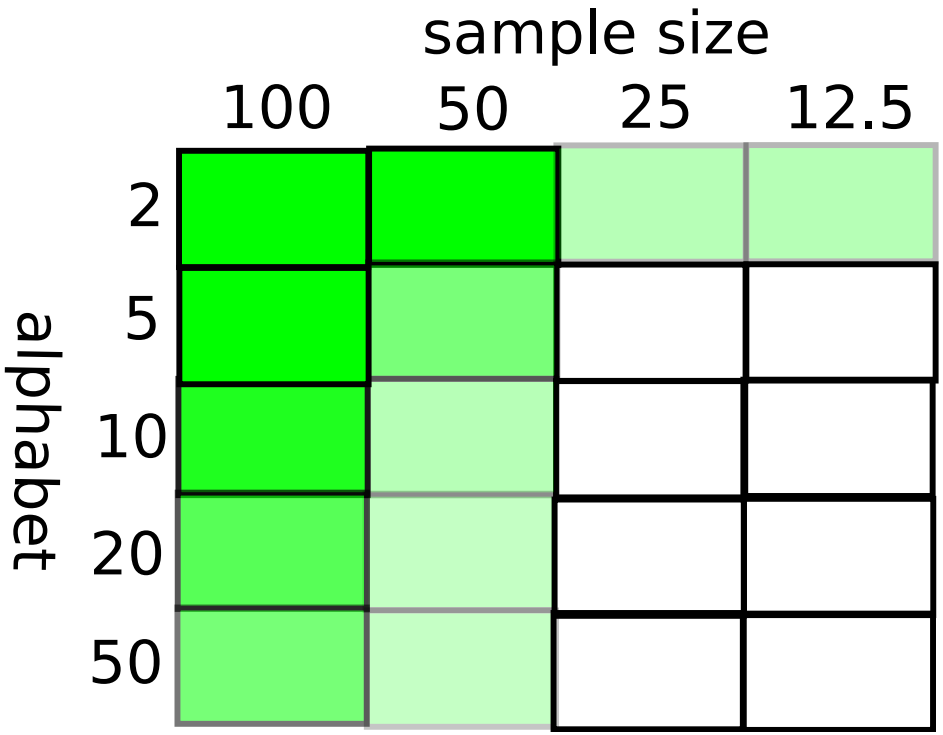
Graph comparison



Competition

- Existing techniques tend to be evaluated on an alphabet of 2,
- Not necessarily sparse automata,
- With an uncertain transition structure, software models
  - tend to have hub-based structure
  - more states tends to mean larger depth
- The idea is to start a competition where one would aim to learn state machines typical of software.

Participate



- *http://stamina.chefbe.net/*
- Download sequences, upload labelling of tests,
- USD 1053 prise money,
- Special issue of Journal of Empirical Software Engineering.

PostDoc position open

PostDoc position open at the Unversity of Sheffield, UK, for up to 2 years from now.