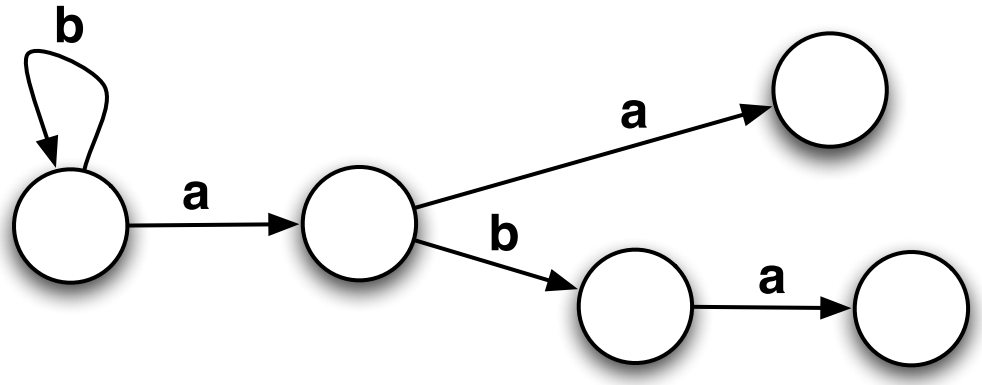
Efficiently learning timed systems

Sicco Verwer

TU Delft
Delft University of Technology

Deterministic Finite state Automata

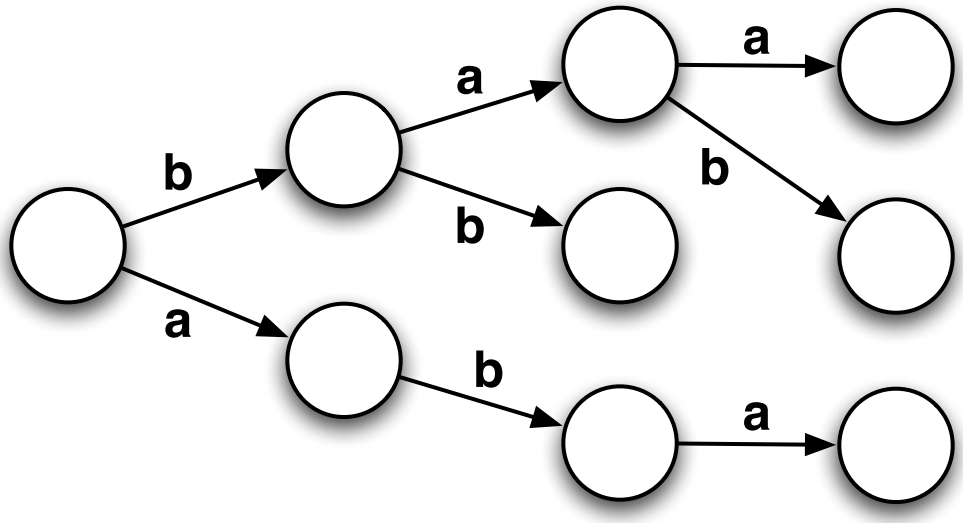


Learning a DFA from observations

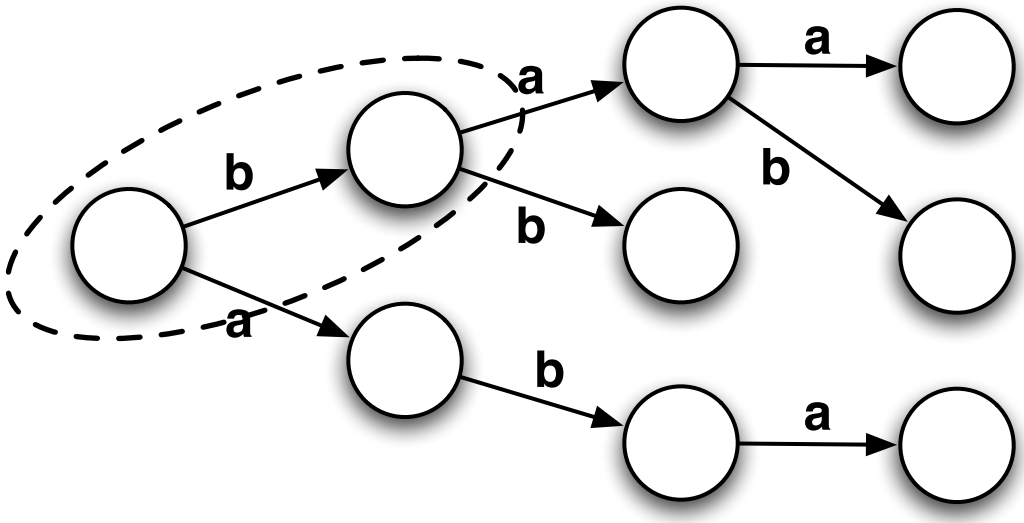
- The input is a finite set of words S:
  - { abab, aaabab, .. }

- The output is a DFA such that it is consistent with S:
  - Every prefix that leads to a state q, has the same distribution over futures in S.
- And preferably it has the least number of states among all possible automata consistent with S.

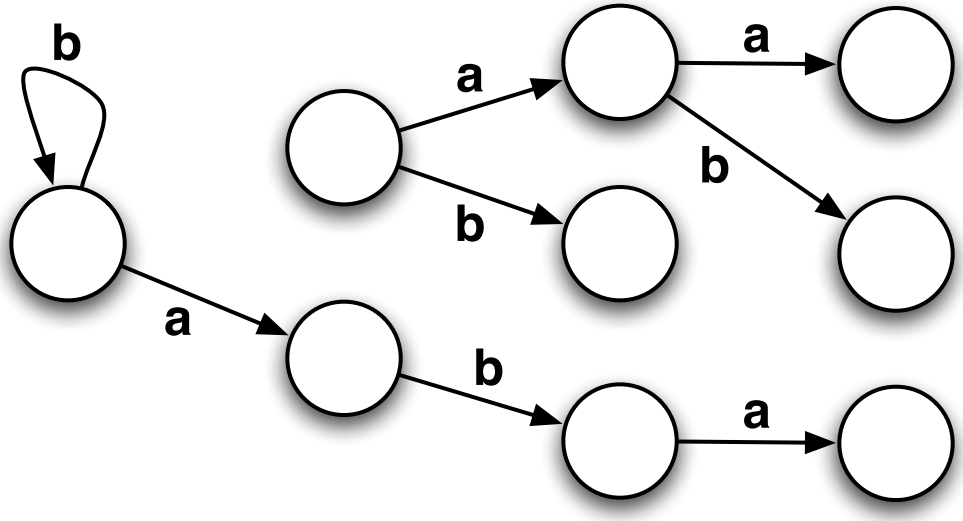
A prefix tree



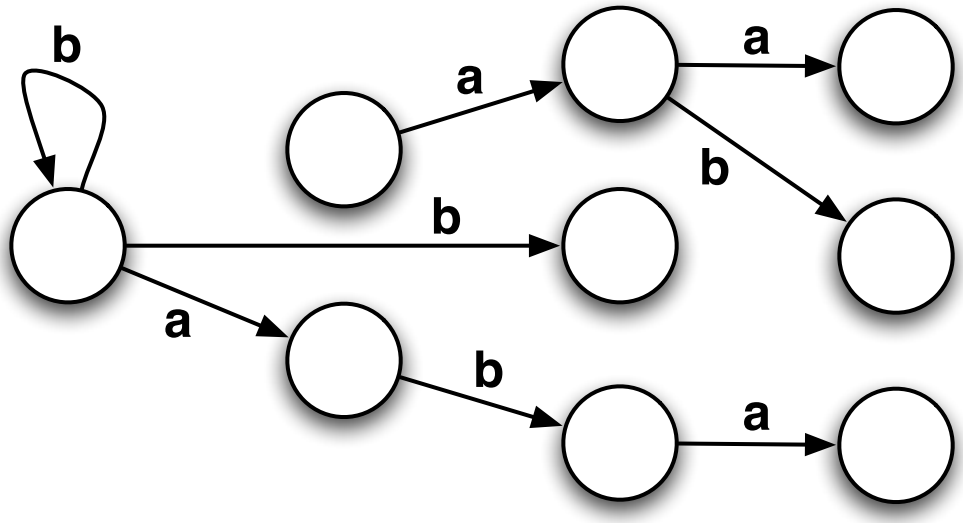
Merging a state



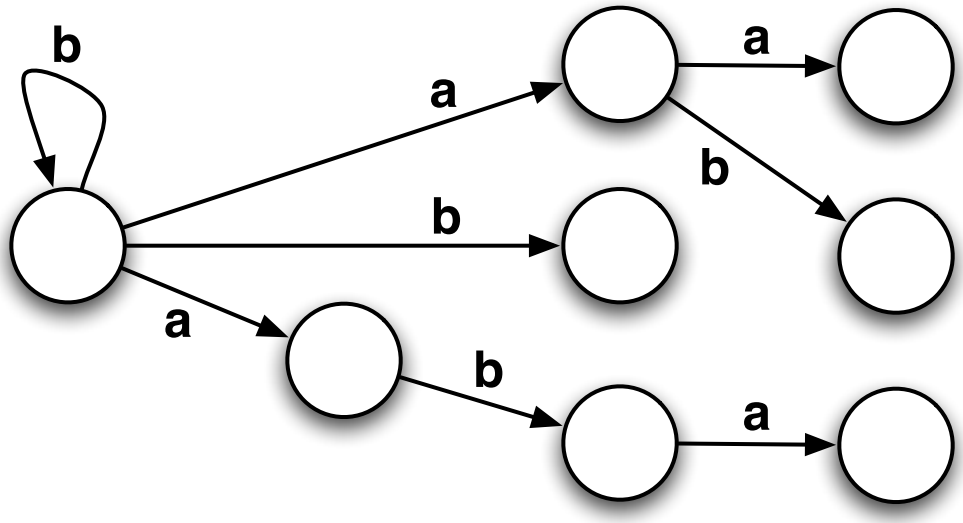
Merging a state



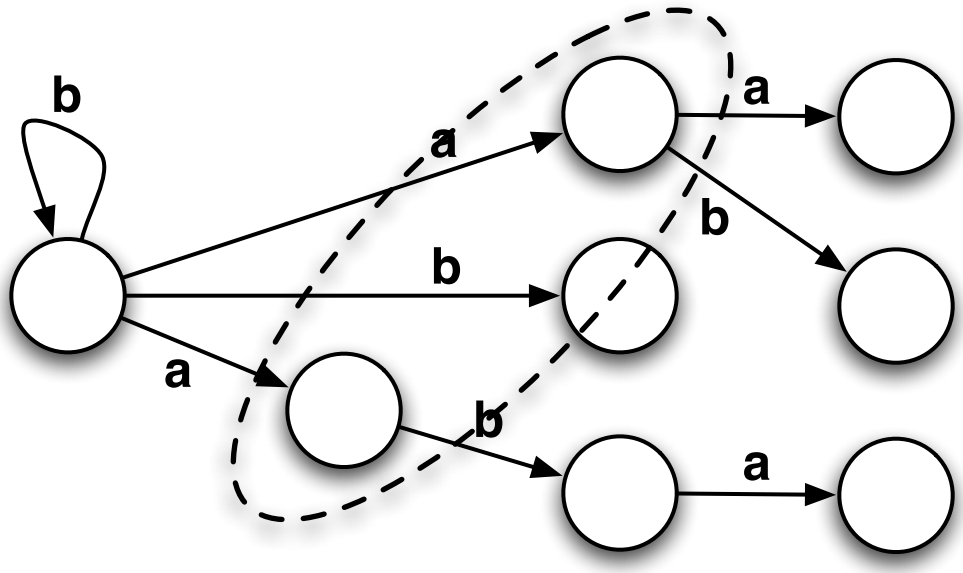
Merging a state



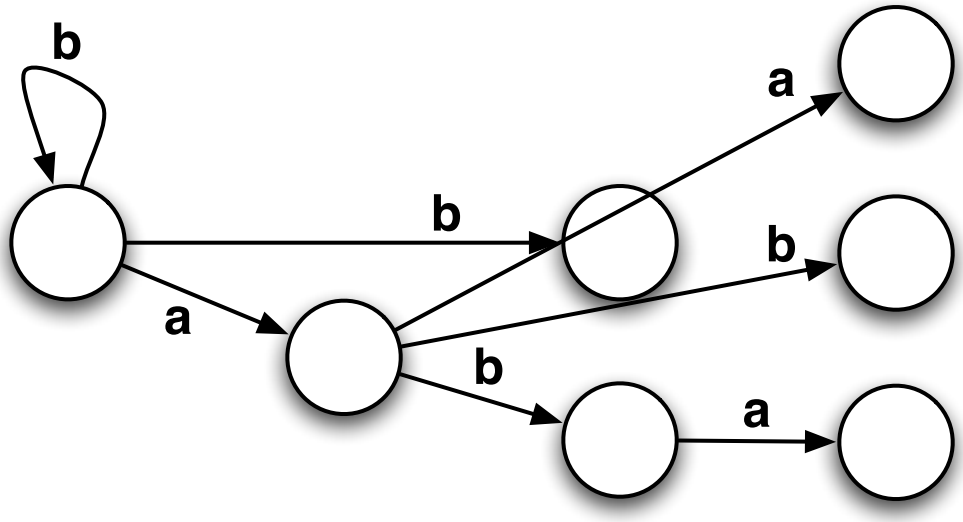
Merging a state



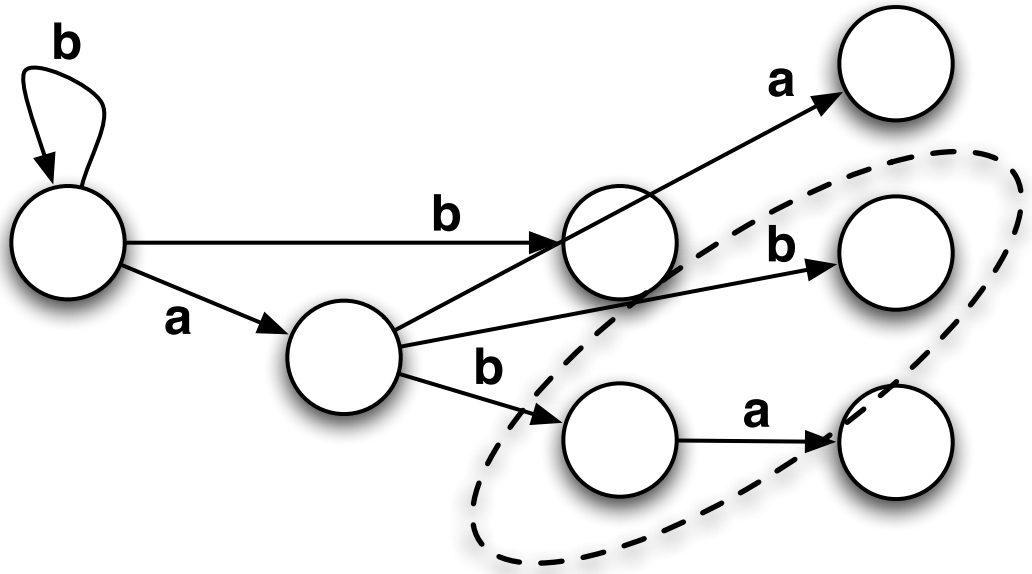
Determinizing a merge



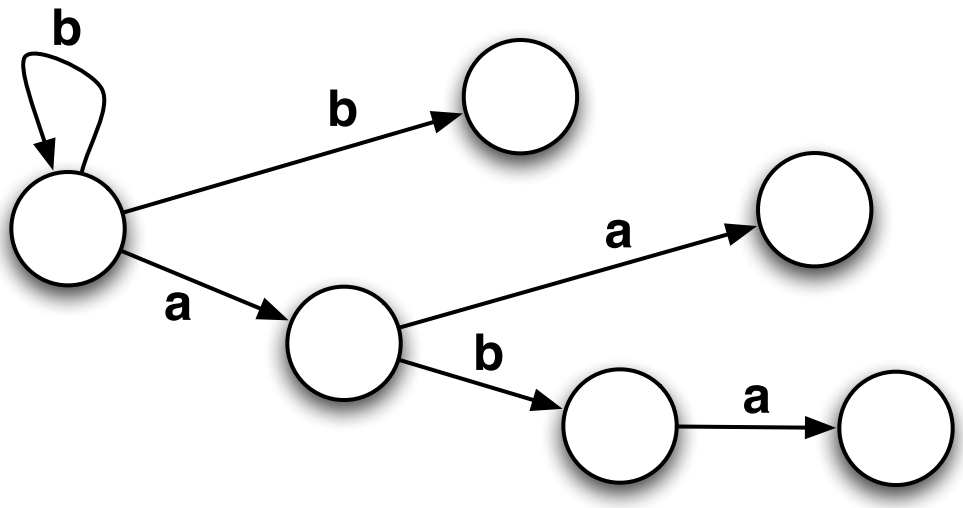
Determinizing a merge



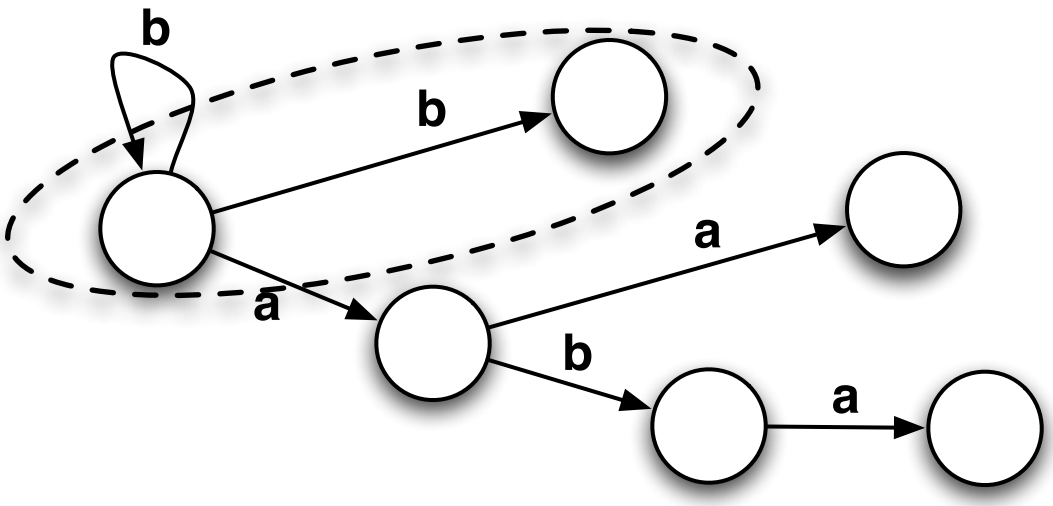
Determinizing a merge



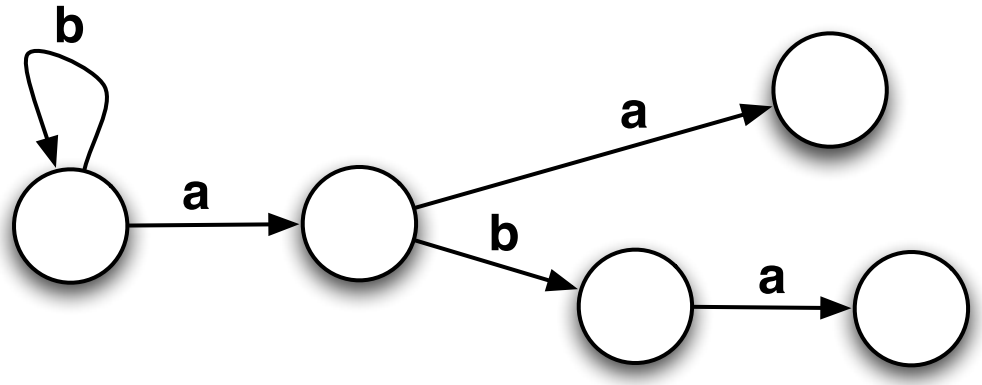
Determinizing a merge



Determinizing a merge



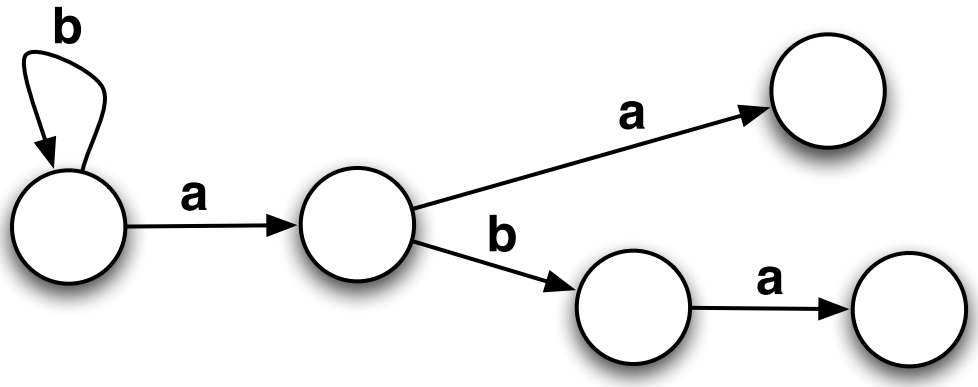
Determinizing a merge



State merging

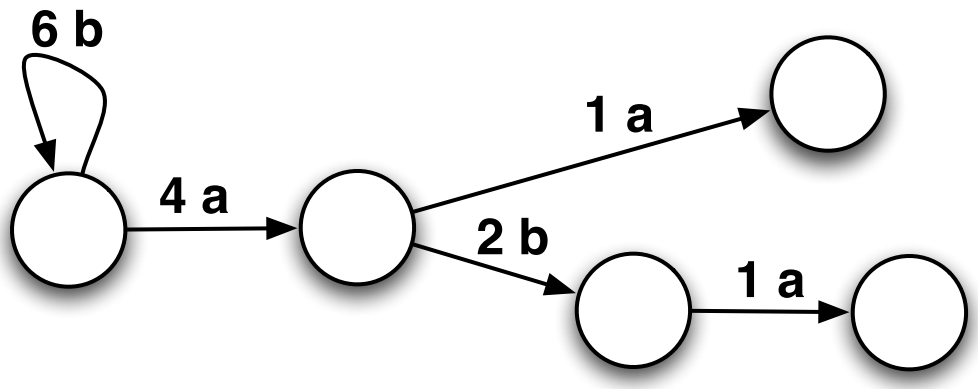
- Construct a prefix tree from S:
  - A tree-shaped DFA where each prefix of a string from S leads to a unique state.
- This DFA is consistent with S.
- Merge consistent states of the DFA until no more consistent merges are possible:
  - Two states q and q' can be merged if the distribution over futures in q and q' are equivalent.
- Optionally backtrack or make use some other search mechanism

Distribution over futures in S



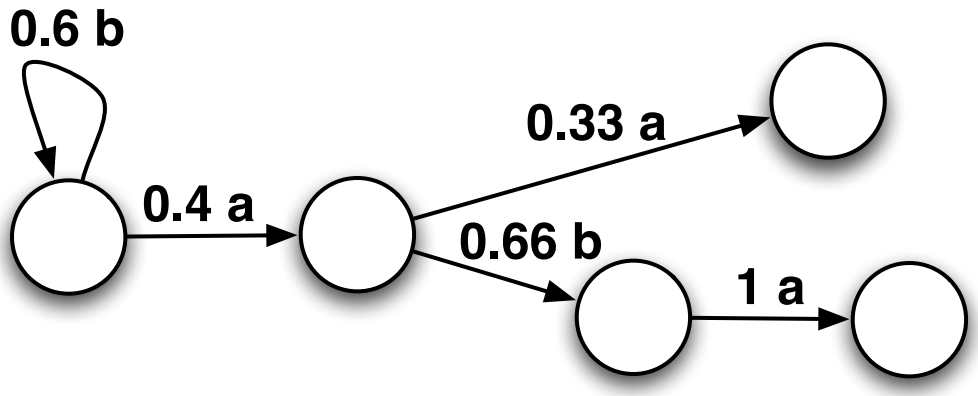
{aa, aba, bab, bba, bbb}

Distribution over futures in S



{aa, aba, bab, bba, bbb}

Distribution over futures in S



{aa, aba, bab, bba, bbb}

State merging with multinomial tests

- Each state defines a distribution over futures
  - P : strings $\Rightarrow [0,1]$, $\sum_s P(s) = 1$.
- We treat two distributions P and P' as being equal if P and P' are not significantly different.
- We would like to test this using the Chi-square test.
- The probabilities of these futures are dependent:
  - For example P(aaba) is dependent on P(aab).
- The Chi-square test can only be applied to independent probabilities, i.e. to fixed length futures.

Many independent tests

- Each state defines a next symbol distribution:
  - A fixed length future of length one.
- The distribution over longer futures is defined by the next symbol distributions of future states.
- This results in many independent tests for the difference of P and P'.
- We the p-values of these tests using a multiple hypothesis testing method.

Learning an RTA

- The input is a finite set of timed words S:
  - {(a,1.0)(b,3.4).., (a,0.2)(a,0.5)..., ... }

- The output is a real-time automaton such that it is consistent with S.
- It preferably has the least number of transitions amongst all possible automata consistent with S

The value of time

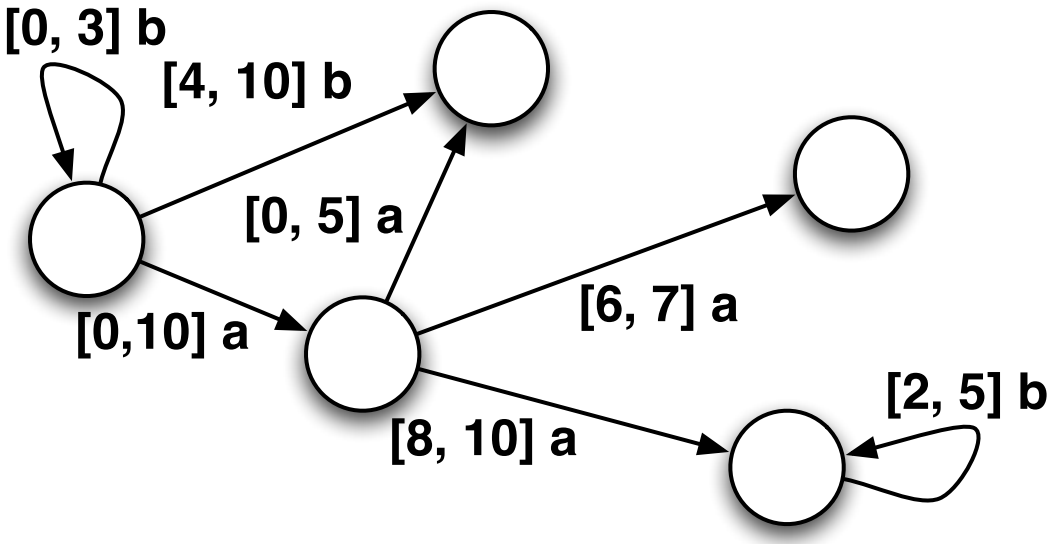
- Timed automata represent time explicitly, i.e. with numbers.
- Instead time can be represented implicitly, i.e. using states.
- These two are equivalent:
  - A set of time values can be represented by a single state.
- However, this results in an exponential blow-up of both the required data set and the resulting automaton.
- Hence, any algorithm used to learn this automaton is inefficient by definition.
- Using the time values explicitly can avoid this inefficiency.

Real-time automata (RTA)

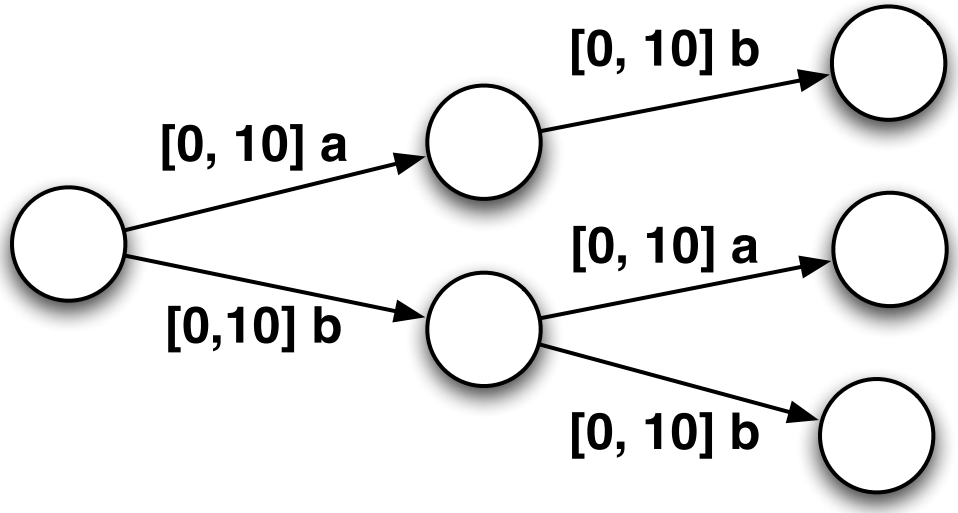
- A state transition can depend on the time delay d between two consecutive events:
  - state transitions optionally contain a guard: $d \in [t,t']$
  - a transition can fire only if its guard is satisfied

- In normal timed automata there can be a guard between any two events

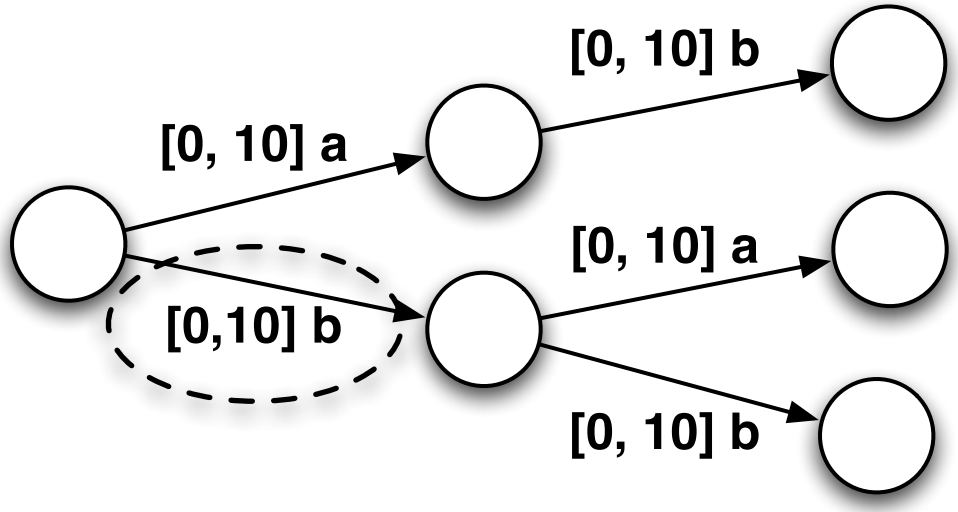
Real-time automaton



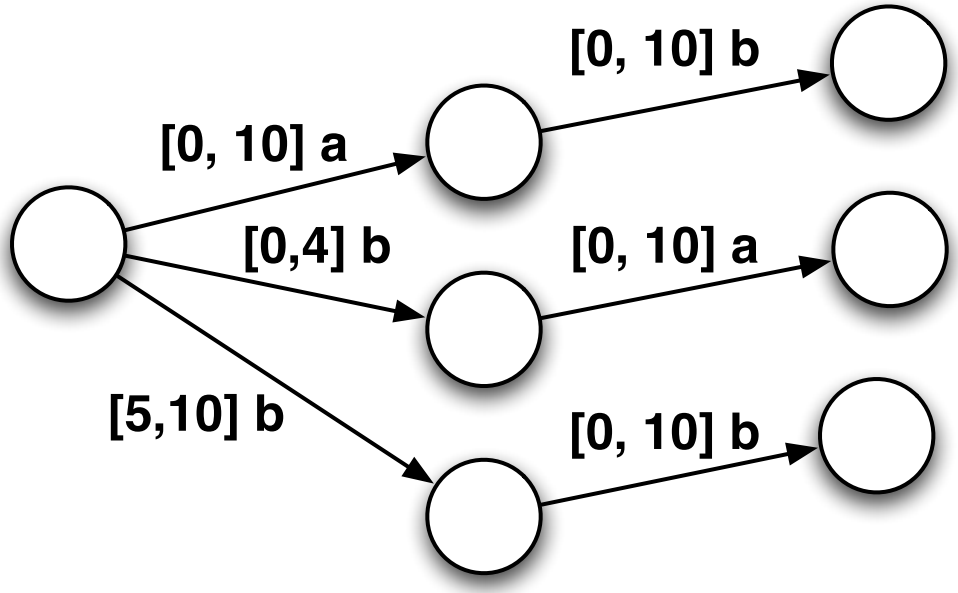
Timed prefix tree



Splitting a transition



Splitting a transition



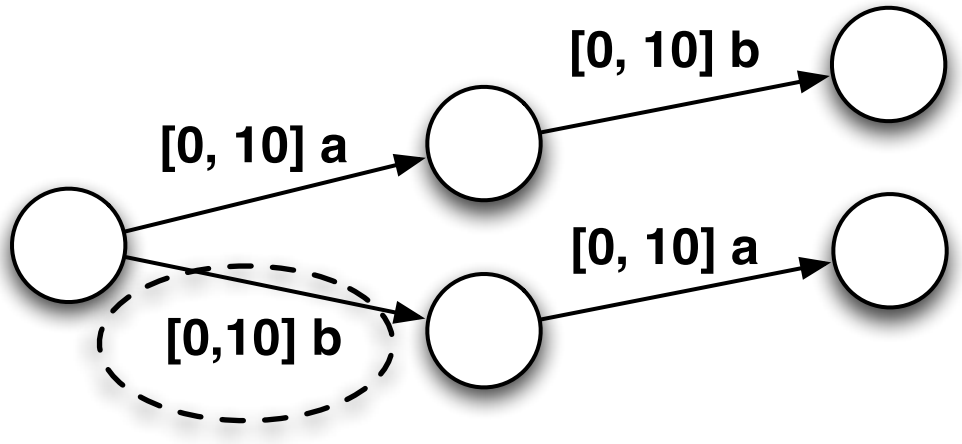
A timed statistical test

- In addition to a distribution over the next symbol, each state in an RTA defines a distribution over time:
- For each symbol an ordered distribution over time values.
- The KS-test can be used to test whether two such distributions are significantly different.

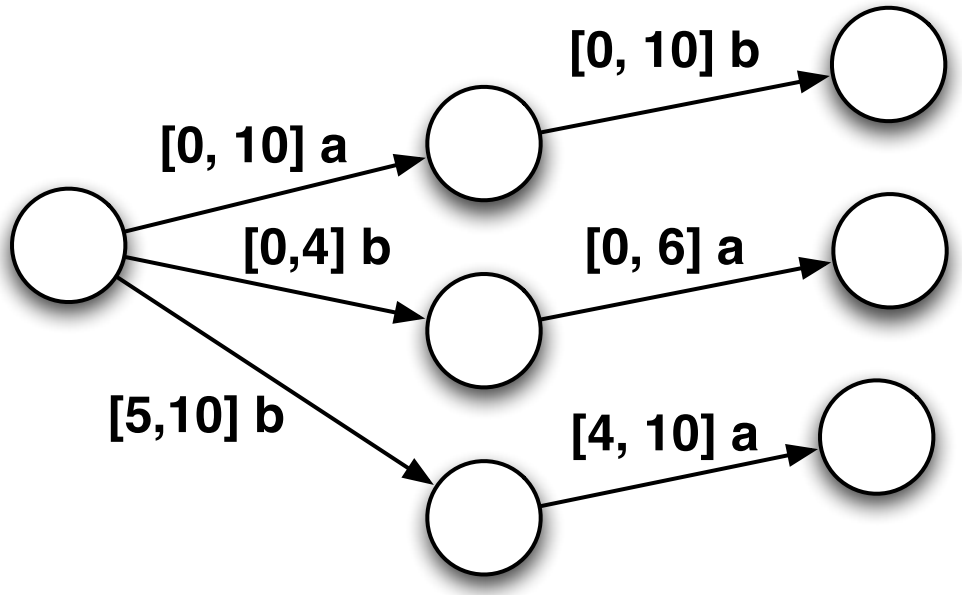
Many independent tests

- Each symbol defines a time distribution in every state for futures of length one.
- The time distribution over longer futures is defined by the time distributions in future states.
- This results in many independent tests for the difference of P and P'.
- We the p-values of these tests using a multiple hypothesis testing method.

Splitting a transition



Splitting a transition



Learning an RTA

- Construct a timed prefix tree.
- Merge states of the automaton.
- Split transitions of the automaton into two:
  - $[t, t'] \rightarrow [t,t''], [t'' + 1, t']$
- Continue until no more consistent merges or splits can be performed.
- Optionally backtrack or make use some other search mechanism.

Conclusions

- Timed systems can be learned efficiently from unlabeled data.
- If the data is timed in nature, representing time explicitly reduces the complexity.

- Soon we want to apply our learning algorithm to real-world problems.