A likelihood–ratio test for identifying probabilistic deterministic real–time automata from positive data

Sicco Verwer\textsuperscript{1}, Mathijs de Weerdt\textsuperscript{2}, and Cees Witteveen\textsuperscript{2}
ICGI 2010

1. Eindhoven University of Technology
2. Delft University of Technology
Overview

- Real-time automata
  - Why learn them?
- Learning real-time automata
  - Labeled
  - Unlabeled
- A likelihood-ratio test
- Results
- Conclusions
Real-time automata (DRTAs)

Transitions contain guards on time values
DRTAs and events

Produces **timed strings**:
(a,6)(a,2)(a,3); (a,2)(b,1)(a,1)(b,8)
Why learn DRTAs?

- Suppose you observe:
  - (a,4)(b,7)(a,2)(a,10)…
- You can **transform** this into:
  - TTTTaTTTTTTTTbTTaTTTTTTTTTTTTa...
  - where T represents a time tick
- Then apply a **DFA learning algorithm**
- The resulting DFA is **language equivalent** to the target DRTA
Why learn DRTAs?

- The transformation goes from binary to unary
- Thus, the learned DFA is exponentially larger than the target DRTA
- DFAs are thus not efficiently learnable in this way from timed data

- DRTAs can be learned efficiently
- DRTAs result in compact representations
State-merging and transition-splitting:
- Start from a prefix tree
- Try all possible merges and splits
- If one scores good:
  - Perform the one that scores best
- Else
  - break
- Iterate
Transition splitting

A prefix tree
all guards are set to true, \([0, \infty)\)
Transition splitting:
choose a transition
Split the transition and **recalculate** the subsequent part of the prefix tree.
Transition splitting

Later, when merging states
Transition splitting

First split the transitions such that the guards match.
Labeled data

- Input: two sets \( S^+ \) and \( S^- \) of \textcolor{red}{\textit{positive}} and \textcolor{blue}{\textit{negative}} timed strings
- Use the state merging and transition splitting algorithm, but
  - \textbf{Allow} merges between positive and negative states
  - Maintain a core of inferred states using the red–blue framework
  - Do not allow positive–negative merges \textbf{in the core}
A core or red states, with a fringe of blue states only allow merges or red and blue states only split transitions to blue states.
When a blue state cannot be merged or split, this blue state is colored red.
Labeled data

Results in a positive-negative merge in the core
Labeled data

Does not results in a positive-negative merge in the core
Does not result in a positive-negative merge in the core
Labeled data

Does not results in a positive-negative merge in the core
The positive-negative merge can be resolved using a split.
The positive-negative merge can be resolved using a split
Unlabeled data

- Input: one set $S^+$ of positive timed strings
- Use the state merging and transition splitting algorithm, but
  - Maintain a core of inferred states using the red-blue framework
  - Only perform splits/merges of the transition/state that is visited by the most examples from $S^+$
Unlabeled data

There are only positive states
Unlabeled data

States should be merged if their future behavior is similar
This behavior is determined by the trees rooted by the two states.
We are only interested in the overlapping behavior, i.e., the behavior for which we have data.
A multinomial symbol distribution, and a histogram (uniform in every bin) time distribution. This is a semi-Markovian model.
DRTA Distributions

- The time and symbol distributions are independent
- The parameters are set by counting occurrences
- Bins are pooled for low occurrence counts
- Multiplying all time and symbol probabilities gives a distribution over timed strings
Similarity using likelihood ratio

- Two states and their future states define a distribution over timed strings
- Two states are similar if merging results in
  - a significantly smaller model,
  - with respect to the likelihood of $S^+$

- This is a likelihood ratio test
The likelihood ratio test:
Given two nested hypotheses $H$ and $H'$, then

$$LR = \frac{\text{likelihood}(S^+, H)}{\text{likelihood}(S^+, H')}$$

is compared to a chi-squared distribution with $n' - n$ degrees of freedom.

This tests whether the increase in likelihood (LR) is significant for $n' - n$ more parameters.

Fits nicely into state-merging since the models are always nested!
In the two states and the future states:

Determine the **maximized likelihood** of the data
Merge the two states, and computed the maximized likelihood of the data in the merged PDRTA.
The merged model can be created by posing linear constraints on the unmerged model, i.e., they are nested.
Compare the amount of reduced model parameters with the decrease in likelihood using LR.
Using similarity values

- Try all possible merges and splits of the transition to a blue state, compute their similarity p-values

- If the smallest split p-value is less than 0.05:
  - perform the split with the smallest p-value

- Else if the largest merge p-value is greater than 0.05:
  - perform the merge with the largest p-value

- Else color the blue state red
Can learn target DRTAs with 8 states and 8 splits from 2000 timed strings of average length 10
Conclusions

- We have an **efficient** algorithm for learning deterministic real-time automata (DRTA)s.
- The algorithm works both on **labeled** and **unlabeled** data.
- The algorithm is based on the best known algorithm for learning DFAs, state-merging.
- It shows **promising results** on artificial data.
Future work

- Prove convergence:
  - Is it efficient in the limit?
  - What is the convergence speed of the test?
- Try the likelihood ratio test to identify probabilistic DFAs
- Apply the algorithm in practice (more)
  - Code is available on my homepage