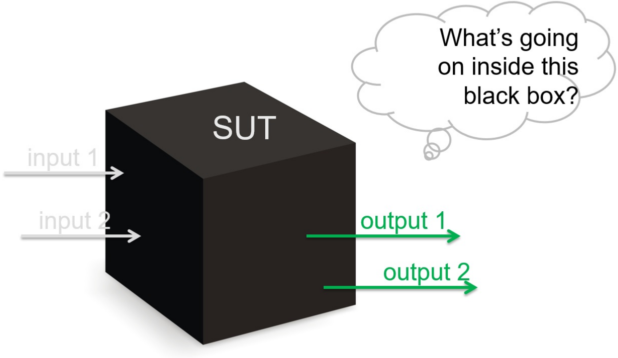
Learning Mealy Machines with Timers

Bengt Jonsson Frits Vaandrager

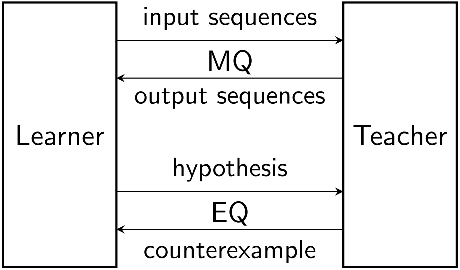
Uppsala University and Radboud University Nijmegen

IPA Fall Days, Nunspeet, November 2017

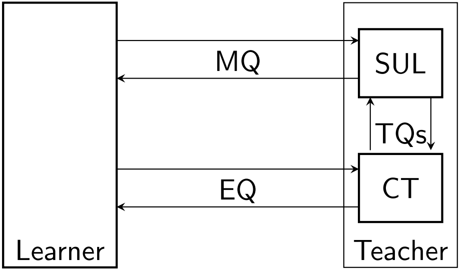
Goal active automaton learning



Minimally adequate teacher (Angluin)



Black box checking (Peled, Vardi & Yannakakis)



Learner: Formulate hypotheses
Conformance Tester (CT): Test correctness hypotheses

LearnLib

HOME NEWS DOWNLOADS FEATURES RESOURCES TEAM HELP

Welcome to the **LearnLib** home page! LearnLib is a *free, open-source* (Apache License 2.0) Java library for *active automata learning*. It is mainly being developed at the Chair for Programming Systems at TU Dortmund University, Germany; a complete list of contributors can be found on the team page.

Note: The open-source LearnLib is a from-scratch re-implementation of the former closed-source version. See the features page for a comparison of the feature sets of the two version.

Background

- Read some Papers on LearnLib
- Papers citing LearnLib at Google Scholar

Search

EXTERNAL LINKS

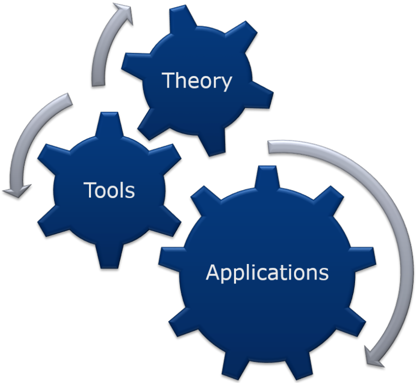
LearnLib @ GitHub
AutomataLib @ GitHub

RECENT POSTS

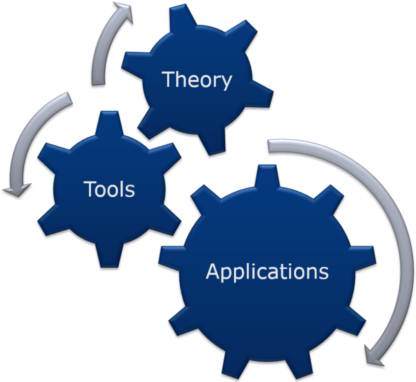
Open Source release of LearnLib

Prof. Dr. Bernhard Steffen

Research method

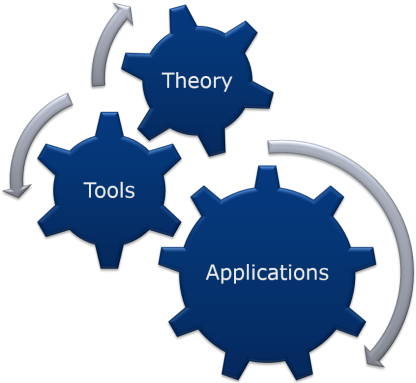


Research method



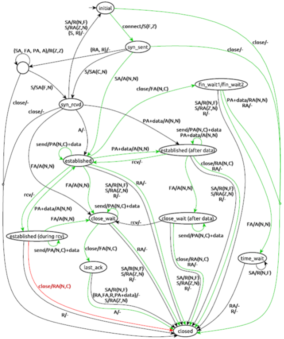
This talk: THEORY

Research method



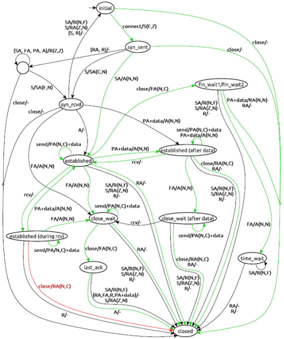
This talk: THEORY (motivated by earlier applications)

Bugs in protocol implementations



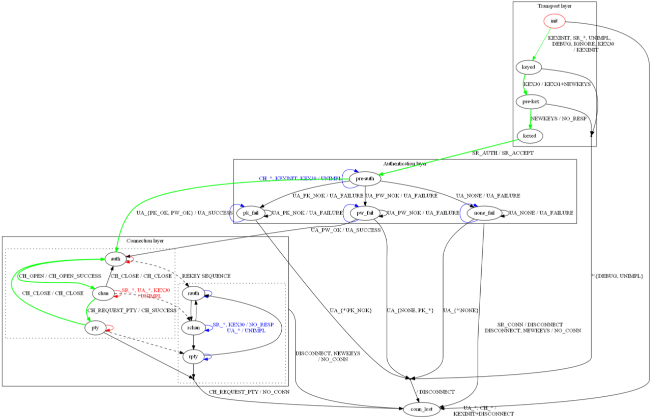
Standard violations found in implementations of major protocols, e.g., TCP (CAV'16, FMICS'17), TLS (Usenix Security'15), SSH (Spin'17).

Bugs in protocol implementations



Standard violations found in implementations of major protocols, e.g., TCP (CAV'16, FMICS'17), TLS (Usenix Security'15), SSH (Spin'17).
These findings led to several bug fixes in implementations.

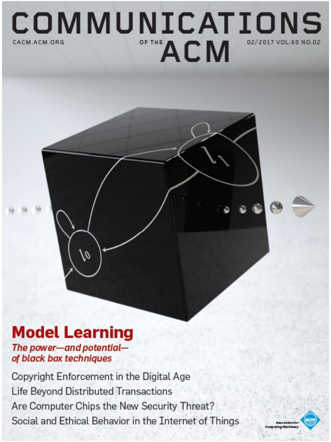
Learned model for SSH implementation



SSH model checking results

	Property	Key word	OpenSSH	Bitvise	DropBear
Security	Trans.		✓	✓	✓
	Auth.		✓	✓	✓
Rekey	Pre-auth.		X	✓	✓
	Auth.		✓	X	✓
Funct.	Prop. 6	MUST	✓	✓	✓
	Prop. 7	MUST	✓	✓	✓
	Prop. 8	MUST	X*	X	✓
	Prop. 9	MUST	✓	✓	✓
	Prop. 10	MUST	✓	✓	✓
	Prop. 11	SHOULD	X*	X*	✓
	Prop. 12	MUST	✓	✓	X

For background and applications see CACM review article



Motivation for work presented today

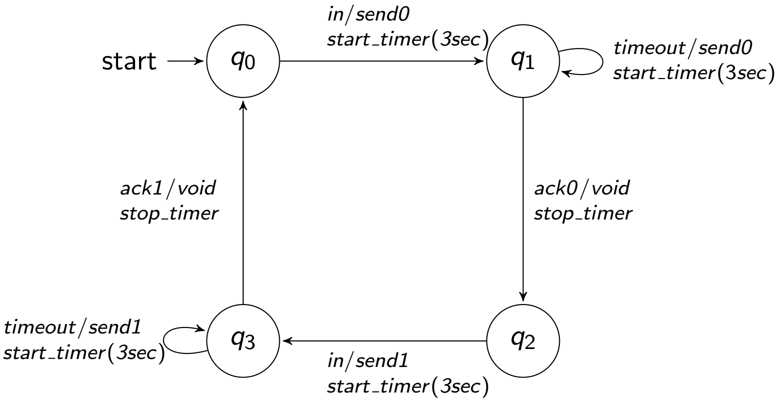
Timing behavior plays a crucial role in applications of model learning, but existing algorithms and tools cannot handle it.
There is some work on algorithms for learning timed systems:

- Grinchtein, Jonsson & Leucker.
  Learning of event-recording automata. TCS, 2010.
- Mens & Maler.
  Learning Regular Languages over Large Ordered Alphabets.
  LMCS, 2015.
- Caldwel, Cardell-Oliver & French.
  Learning time delay Mealy machines. IEEE TASE, 2016.

but this is not so practical because of high complexity and/or limited expressivity.

Timing Behavior in Network Protocols

Sender alternating-bit protocol, adapted from Kurose & Ross, Computer Networking:



Idea

Develop learning algorithm for Mealy machines with timers!!!

Idea

Develop learning algorithm for Mealy machines with timers!!!



Occurrence of timing dependent behavior fully determined by previous behavior

MMTs

Assume an unbounded set X of timers x, x_1, x_2, etc. For a set I, write $\hat{I} = I \cup \{to[x] \mid x \in X\}$.

Definition

A Mealy machine with timers (MMT) is a tuple $\mathcal{M} = (I, O, Q, q_0, \mathcal{X}, \delta, \lambda, \pi)$, where

- I and O are finite sets of input and output events
- Q is a finite set of states with $q_0 \in Q$ the initial state
- $\mathcal{X} : Q \to \mathcal{P}_{fin}(X)$, with $\mathcal{X}(q_0) = \emptyset$
- $\delta : Q \times \hat{I} \hookrightarrow Q$ is a transition function,
- $\lambda : Q \times \hat{I} \hookrightarrow O$ is an output function,
- $\pi : Q \times \hat{I} \hookrightarrow (X \hookrightarrow \mathbb{N}^{>0})$ is a timer update function

(satisfying some natural conditions)

Operations on timers

Write $q \xrightarrow{i/o,\rho} q'$ if $\delta(q,i) = q'$, $\lambda(q,i) = o$ and $\pi(q,i) = \rho$.
Basically, four things can happen:

1. If $x \in \mathcal{X}(q) \setminus \mathcal{X}(q')$ then input i stops timer x.
2. If $x \in \mathcal{X}(q') \setminus \mathcal{X}(q)$ then i starts timer x with value $\rho(x)$.
3. If $x \in \mathcal{X}(q) \cap \mathrm{dom}(\rho)$ then i restarts timer x with value $\rho(x)$.
4. Finally, if $x \in \mathcal{X}(q') \setminus \mathrm{dom}(\rho)$ then timer x is unaffected by i.

Timed Semantics (1)

A configuration of an MMT is a pair (q, κ) of a state q and a valuation $\kappa : \mathcal{X}(q) \to \mathbb{R}^{\geq 0}$ of its timers. When time advances, all timers decrease at the same rate; a timeout occurs when value of some timer becomes 0.

A timed run of an MMT is a sequence

$$(q_0, \kappa_0) \xrightarrow{d_1} (q_0, \kappa_0') \xrightarrow{i_1/o_1} (q_1, \kappa_1) \xrightarrow{d_2} \cdots \xrightarrow{i_k/o_k} (q_k, \kappa_k)$$

of configurations, nonzero delays, and discrete transitions.

Timed Semantics (2)

A timed word describes an observation we can make on an MMT:

$$w \quad = \quad d_1 \ i_1 \ o_1 \ d_2 \ i_2 \ o_2 \cdots d_k \ i_k \ o_k,$$

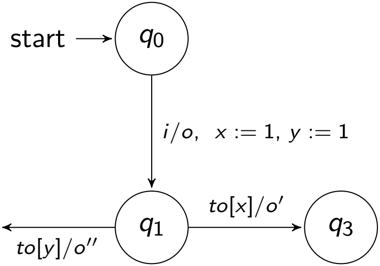
where $d_j \in \mathbb{R}^{>0}$, $i_j \in I \cup \{to\}$, and $o_j \in O$.

To each timed run α we associate a timed word $tw(\alpha)$ by forgetting the configurations and names of timers in timeouts.

Definition

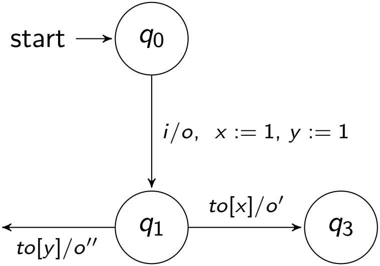
MMTs $\mathcal{M}$ and $\mathcal{N}$ are timed equivalent, denoted $\mathcal{M} \approx_{timed} \mathcal{N}$, iff they have the same timed words.

"Uncontrollable" Nondeterminism



Accepts timed words $1 \ i \ o \ 1 \ to \ o'$ and $1 \ i \ o \ 1 \ to \ o''$.

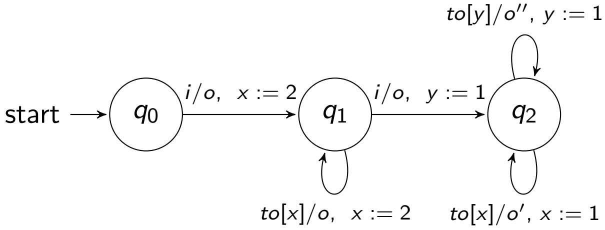
"Uncontrollable" Nondeterminism



Accepts timed words $1\ i\ o\ 1\ to\ o'$ and $1\ i\ o\ 1\ to\ o''$.

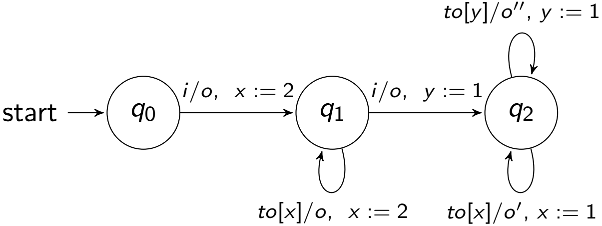
$\Rightarrow$ We assume at most one timer can be updated per transition.

"Controllable" Nondeterminism



Accepts timed words 7 i o 1 i o 1 to o' and 7 i o 1 i o 1 to o''.

“Controllable” Nondeterminism

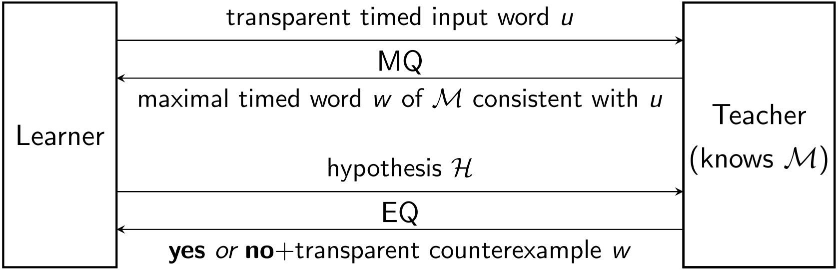


Accepts timed words $7 \; i \; o \; 1 \; i \; o \; 1 \; to \; o'$ and $7 \; i \; o \; 1 \; i \; o \; 1 \; to \; o''$.

$\Rightarrow$ During learning we will simply avoid these race conditions.

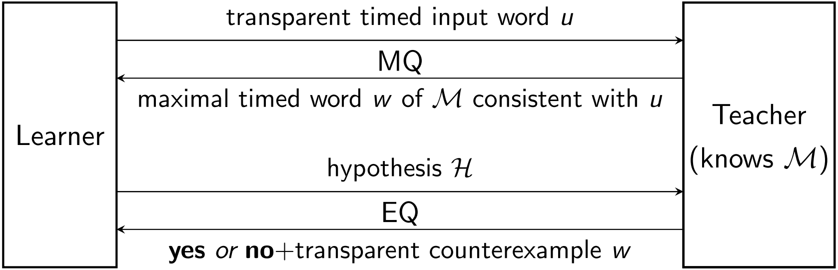
A timed MAT framework

A timed input word is a sequence $u = d_1 \ i_1 \cdots d_k \ i_k \ d_{k+1}$, with $d_j \in \mathbb{R}^{>0}$ and $i_j \in I$, for $j \leq k$, and $d_{k+1} \in \mathbb{R}^{\geq 0}$. A timed (input) word is transparent if inputs occur at different fractional times.



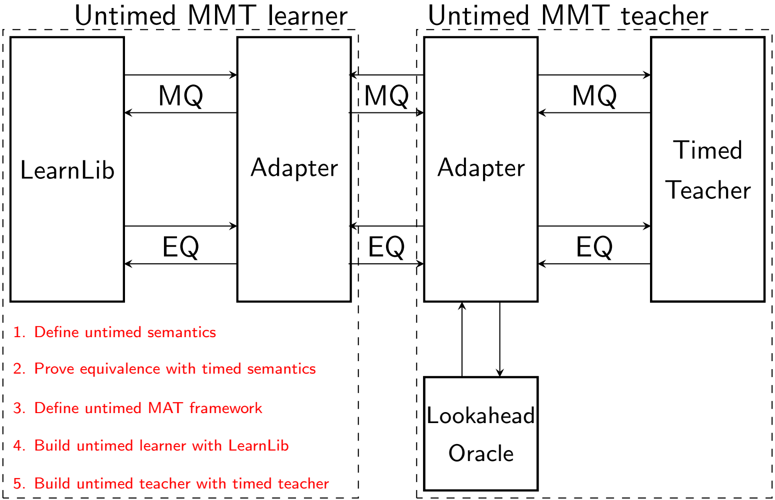
A timed MAT framework

A timed input word is a sequence $u = d_1\ i_1 \cdots d_k\ i_k\ d_{k+1}$, with $d_j \in \mathbb{R}^{>0}$ and $i_j \in I$, for $j \leq k$, and $d_{k+1} \in \mathbb{R}^{\geq 0}$. A timed (input) word is transparent if inputs occur at different fractional times.



Main contribution: algorithm allowing learner to construct MMT $\mathcal{N}$ that is timed equivalent to $\mathcal{M}$ (under mild restrictions).

Plan of attack



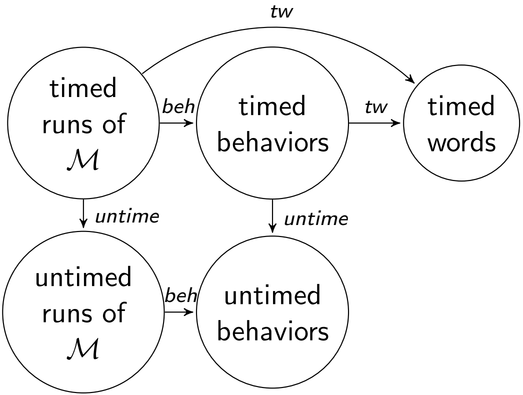
1. Define untimed semantics
2. Prove equivalence with timed semantics
3. Define untimed MAT framework
4. Build untimed learner with LearnLib
5. Build untimed teacher with timed teacher

Timed and Untimed Runs and Behaviors

$$
\begin{tikzcd}[column sep=tiny]
& (q_0, \kappa_0) \xrightarrow{d_1} (q_0, \kappa_0') \xrightarrow{i_1/o_1} (q_1, \kappa_1) \cdots (q_{k-1}, \kappa_{k-1}') \xrightarrow{i_k/o_k} (q_k, \kappa_k) \arrow[dl, "untime"'] \arrow[dr, "beh"] & \\
q_0 \xrightarrow{i_1/o_1, \rho_1} q_1 \cdots q_{k-1} \xrightarrow{i_k/o_k, \rho_k} q_k \arrow[dr, "beh"'] & & \kappa_0 \xrightarrow{d_1} \kappa_0' \xrightarrow{i_1/o_1, \rho_1} \kappa_1 \cdots \kappa_{k-1}' \xrightarrow{i_k/o_k, \rho_k} \kappa_k \arrow[dl, "untime"] \\
& X_0 \xrightarrow{i_1/o_1, \rho_1} X_1 \cdots X_{k-1} \xrightarrow{i_k/o_k, \rho_k} X_k &
\end{tikzcd}
$$

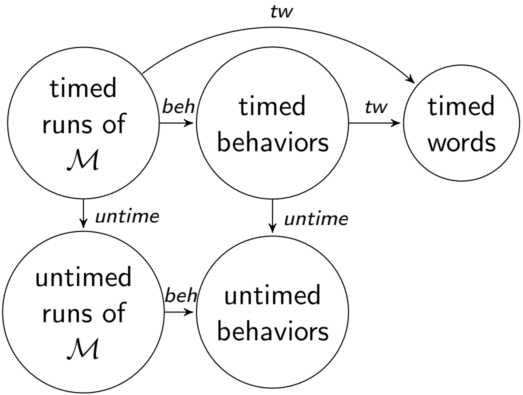
Timed and Untimed Runs and Behaviors

Diagram commutes and has a pullback:



Timed and Untimed Runs and Behaviors

Diagram commutes and has a pullback:



CAN WE DEFINE SEMANTICS MMTs IN TERMS OF UNTIMED BEHAVIORS??

Feasibility

Definition

An untimed behavior

$$\beta \ = \ X_0 \xrightarrow{i_1/o_1,\rho_1} X_1 \xrightarrow{i_2/o_2,\rho_2} X_2 \cdots \xrightarrow{i_k/o_k,\rho_k} X_k$$

is feasible if there is a timed behavior σ such that $untime(\sigma) = \beta$.

Example of untimed behavior that is *not* feasible:

$$\emptyset \xrightarrow{i_1/o_1,x:=1} \{x\} \xrightarrow{i_2/o_2,y:=100} \{x, y\} \xrightarrow{to[y]/o_3} \emptyset$$

Isomorphism

An isomorphism between untimed behaviors β and β' is a consistent renaming of timers:

$$
\begin{tikzcd}
\emptyset \arrow[r, "i_1/o_1{,}x{:=}2"] \arrow[d] & \{x\} \arrow[r, "i_2/o_2{,}y{:=}1"] \arrow[d] & \{x, y\} \arrow[r, "to[y]/o_3{,}y{:=}100"] \arrow[d] & \{x, y\} \arrow[d] \\
\emptyset \arrow[r, "i_1/o_1{,}x_1{:=}2"] & \{x_1\} \arrow[r, "i_2/o_2{,}x_2{:=}1"] & \{x_1, x_2\} \arrow[r, "to[x_2]/o_3{,}x_3{:=}100"] & \{x_1, x_3\}
\end{tikzcd}
$$

Isomorphism

An isomorphism between untimed behaviors β and β' is a consistent renaming of timers:

$$\begin{tikzcd}
\emptyset \arrow[r, "i_1/o_1{,}x:=2"] \arrow[d] & \{x\} \arrow[r, "i_2/o_2{,}y:=1"] \arrow[d] & \{x,y\} \arrow[r, "to[y]/o_3{,}y:=100"] \arrow[d] & \{x,y\} \arrow[d] \\
\emptyset \arrow[r, "i_1/o_1{,}x_1:=2"] & \{x_1\} \arrow[r, "i_2/o_2{,}x_2:=1"] & \{x_1,x_2\} \arrow[r, "to[x_2]/o_3{,}x_3:=100"] & \{x_1,x_3\}
\end{tikzcd}$$

An untimed behavior is in canonical form if, for each j, the timer that is updated in the j-th event (if any) is equal to x_j.
Each untimed behavior is isomorphic to a unique untimed behavior in canonical form.

Untimed semantics

Definition

MMTs $\mathcal{M}$ and $\mathcal{N}$ are untimed equivalent, $\mathcal{M} \approx_{untimed} \mathcal{N}$, iff their sets of feasible untimed behaviors are isomorphic.

Untimed semantics

Definition

MMTs $\mathcal{M}$ and $\mathcal{N}$ are untimed equivalent, $\mathcal{M} \approx_{untimed} \mathcal{N}$, iff their sets of feasible untimed behaviors are isomorphic.

Theorem

$\mathcal{M} \approx_{untimed} \mathcal{N}$ implies $\mathcal{M} \approx_{timed} \mathcal{N}$.

Untimed semantics

Definition

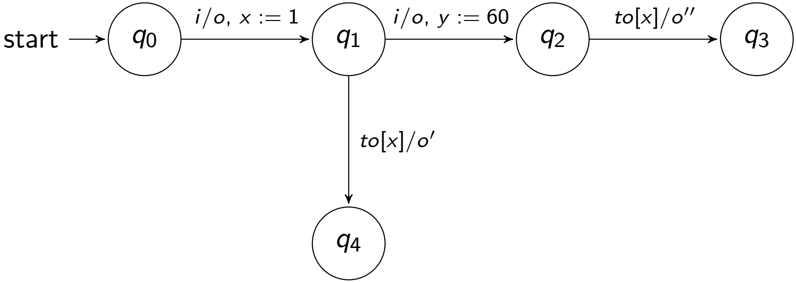
MMTs $\mathcal{M}$ and $\mathcal{N}$ are untimed equivalent, $\mathcal{M} \approx_{untimed} \mathcal{N}$, iff their sets of feasible untimed behaviors are isomorphic.

Theorem

$\mathcal{M} \approx_{untimed} \mathcal{N}$ implies $\mathcal{M} \approx_{timed} \mathcal{N}$.

Converse implication does not hold in general.

Ghost timers



Equivalence of Timed and Untimed Semantics

Theorem

Suppose that $\mathcal{M}$ and $\mathcal{N}$ are MMTs without ghost timers in which at most one timer is started on each transition.
Then $\mathcal{M} \approx_{timed} \mathcal{N}$ implies $\mathcal{M} \approx_{untimed} \mathcal{N}$.

Equivalence of Timed and Untimed Semantics

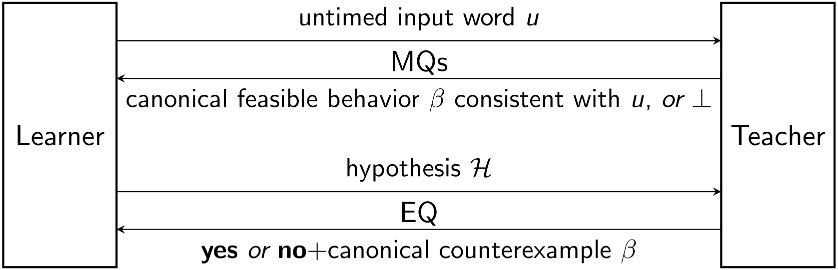
Theorem

Suppose that $\mathcal{M}$ and $\mathcal{N}$ are MMTs without ghost timers in which at most one timer is started on each transition.
Then $\mathcal{M} \approx_{timed} \mathcal{N}$ implies $\mathcal{M} \approx_{untimed} \mathcal{N}$.

Main proof technique: wiggling of timed behaviors to ensure that fractional starting times of different inputs are different.

An untimed MAT framework

An untimed input word is a sequence $u = i_1 \cdots i_k$ over $\hat{I}$ such that $i_j = to[x_l]$ implies $l < j$, and each timer expires at most once.



Nerode congruence

Definition

Let S be a set of feasible untimed behaviors. Behaviors $\beta, \beta' \in S$ are equivalent, notation $\beta \equiv_S \beta'$, iff for any untimed behavior γ, $\beta \cdot \gamma \in S \Leftrightarrow \beta' \cdot \gamma \in S$.

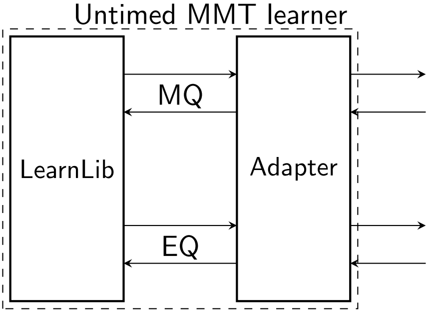
Myhill-Nerode theorem

Theorem

Let S be a set of feasible untimed behaviors over finite sets of inputs I and outputs O. Then S is the set of feasible untimed behaviors of an MMT $\mathcal{M}$ iff

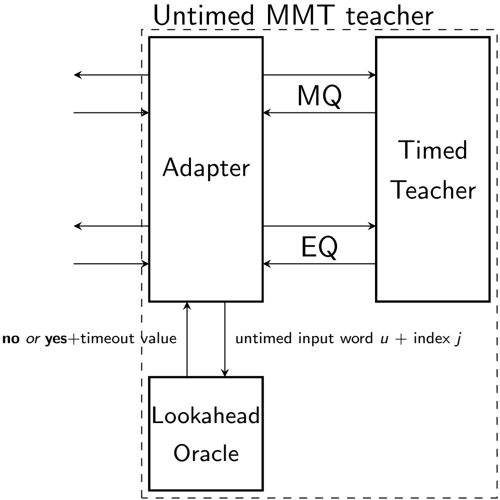
1. S is nonempty,
2. all behaviors in S start with the empty set of timers,
3. the set of timers that occur in S is finite,
4. S is prefix closed,
5. S is behavior deterministic,
6. S is input complete,
7. S is timeout complete, and
8. $\equiv_S$ has only finitely many equivalence classes (finite index).

Building untimed MMT learner with Mealy machine learner



We assume learner knows bound n on the number of timers that can be active in a state. Adapter uses function *uncan* to translate canonical behaviors to behaviors involving at most n clocks.

Building an untimed MMT teacher with a timed teacher



Query complexity

Number of queries polynomial in size canonical MMT $\mathcal{N}$ produced by Myhill-Nerode construction.

This MMT may be exponentially bigger (in the number of clocks) than original MMT $\mathcal{M}$ of the teacher (cf register automata).

For MMTs with single timer, learning is easy: all untimed behaviors are feasible, lookahead oracle is trivial if we assume learner knows bound on maximal timer value (just wait), and complexity is the same as for Mealy machine with the same size.

Conclusions

Our work consitutes a major step towards a practical approach for active learning of timed systems.

Just like timed automata paved the way to extend model checking to a timed setting, we expect that MMTs will make it possible to lift model learning to a timed setting.

Future Work

1. Implement equivalence oracle
2. Implement lookahead oracle (inspired by Tomte tool)
3. Handle non transparent counterexamples
4. Deal with timing uncertainty in real applications
5. Implement our algorithm and apply to practical case studies
6. Many theoretical questions left!