On the Construction of Monitors for Temporal Logic Properties

Marc Geilen
Information and Communication Systems Group
Faculty of Electrical Engineering
Eindhoven University of Technology
P.O.Box 513, 5600 MB Eindhoven
The Netherlands
E-mail: M.C.W.Geilen@tue.nl

July 23, 2001
Overview

1. Introduction
2. Runtime Verification of Temporal Logic Properties
3. Informative Execution Traces
4. Informative Tableaux
5. Construction of Monitors from the Tableaux
6. Conclusions
1. Introduction

- Complexity of verification; run-time verification
- Verifying (detailed) simulation models or (instrumented) hardware or software realisations
- Properties expressed in linear temporal logic
- Finite traces and temporal logic properties
- Incremental, deterministic
2. Labelled Transition Systems

System can be viewed as LTS

Set $Prop$ of observable (boolean) propositions $(p, q)$

Observable states are subsets of $Prop$. 
2. Exploring a Labelled Transition System

Simulating or running the system generates a path through the LTS

A trace $\bar{\sigma} \in (2^{Prop})^\omega$ is an infinite sequence of states

A prefix $\bar{\tau} \in (2^{Prop})^*$ of a trace is a finite sequence of states
2. Linear Temporal Logic

To specify properties, we use LTL formulas (in positive form) \((p \in Prop)\):

\[
\varphi ::= \text{true} \mid \text{false} \mid p \mid \neg p \mid \varphi_1 \lor \varphi_2 \mid \varphi_1 \land \varphi_2 \mid \square \varphi \mid \varphi_1 U \varphi_2 \mid \varphi_1 V \varphi_2.
\]

LTL formulas are interpreted on infinite traces in the standard way.

A run of the system incrementally generates a prefix of a trace of increasing length.

To relate the logic to prefixes we consider the set of all infinite extensions of a given prefix \(\overline{\tau}\).
2. Infinite Traces

For a formula $\varphi$, all (infinite) execution traces can be classified as good (satisfying $\varphi$) or bad (not satisfying $\varphi$).
2. Infinite Traces

A prefix is associated with a set of traces

All extensions of good prefixes are good traces

All extensions of bad prefixes are bad traces

Some prefixes have both good and bad extensions
2. Infinite Traces

Any (finite) extension of a good prefix is a good prefix.

Any (finite) extension of a bad prefix is a bad prefix.
2. Finite Traces

Prefixes can be classified as **good**, **bad** and **indecisive** ones.

As they get longer, **indecisive** ones may become **good** or **bad**.
3. Informativeness

It is possible to effectively construct for a given formula $\varphi$, an automaton $A_\varphi$ that recognises all bad (good) prefixes of a formula $\varphi$ [KV99].

However, not every bad prefix “tells the whole story” about why the formula does (not) hold for all extensions of the prefix.

E.g., if $\psi$ is not satisfiable, then every prefix $\tau$ is a bad prefix for $\diamond \psi$.

The state sequence $\{q\}\{q\}\{q, p\}$ on the other hand is informative for the formula $p \lor q$.

Therefore, we concentrate on informative prefixes [KV99] and we construct tableau automata based upon informativeness.
3. Informativeness

A set \( \Phi \) of formulas is called **locally informative** if

- false \( \notin \Phi \);
- if \( \varphi_1 \lor \varphi_2 \in \Phi \) then \( \varphi_1 \in \Phi \) or \( \varphi_2 \in \Phi \);
- if \( \varphi_1 \land \varphi_2 \in \Phi \) then \( \varphi_1 \in \Phi \) and \( \varphi_2 \in \Phi \);
- if \( \varphi_1 \lor \varphi_2 \in \Phi \) then \( \varphi_1 \in \Phi \) or \( \varphi_2 \in \Phi \);
- if \( \varphi_1 \lor \varphi_2 \in \Phi \) then \( \varphi_2 \in \Phi \).
3. Informativeness

Local informativeness is not enough, we need constraints on the next moment as well.

A set $\Phi$ of formulas is non-trivial for

- the Until formula $\varphi_1 \mathbf{U} \varphi_2$, if $\varphi_1 \mathbf{U} \varphi_2 \in \Phi$ and $\varphi_2 \notin \Phi$, let
  $$\text{Next}(\varphi_1 \mathbf{U} \varphi_2) = \varphi_1 \mathbf{U} \varphi_2;$$

- the Release formula $\varphi_1 \mathbf{V} \varphi_2$, if $\varphi_1 \mathbf{V} \varphi_2 \in \Phi$ and $\varphi_1 \notin \Phi$, let
  $$\text{Next}(\varphi_1 \mathbf{V} \varphi_2) = \varphi_1 \mathbf{V} \varphi_2;$$

- the formula $\Box \varphi$, if $\Box \varphi \in \Phi$, let $\text{Next}(\Box \varphi) = \varphi$. 
3. Informativeness

Let $\Phi$ be a set of formulas, then the set $\text{Next}(\Phi)$ of temporal informativeness constraints is the set

$$\{\text{Next}(\psi) \mid \psi \in \Phi \text{ such that } \Phi \text{ is non-trivial for } \psi\}$$

$\Phi'$ is a temporally informative successor of $\Phi$ if $\text{Next}(\Phi) \subseteq \Phi'$

$$\{p \lor q, p\} \rightarrow \{p \lor q, q\}$$

but

$$\{p \lor q, q\} \rightarrow \emptyset$$

$$\{p \lor q, p\} \not\rightarrow \{p\}$$

$$\{\Box q\} \not\rightarrow \{p\}$$
3. Informativeness

Let $\bar{\tau}$ be a finite state sequence. $\bar{\tau}$ is informative for $\varphi$ iff there exists a finite sequence $\bar{IS} \in (2^{LTL})^*$ of sets of formulas, say of length $n + 1 \leq |\bar{\tau}| + 1$, such that

- $\varphi \in \bar{IS}(0)$;

- $\bar{IS}(n) = \emptyset$;

- for all $0 \leq i < n$ and $\psi \in \bar{IS}(i)$,
  - if $\psi = p$, then $p \in \bar{\tau}(i)$; if $\psi = \neg p$, then $p \notin \bar{\tau}(i)$;
  - $\bar{IS}(i)$ is locally informative;
  - $\bar{IS}(i) \rightarrow \bar{IS}(i + 1)$. 

/ department of electrical engineering
3. Informativeness

An informative sequence $\bar{IS}$ can be regarded as a proof that $\varphi$ holds for all traces with prefix $\bar{\tau}$

That the state sequence $\{q\}\{q\}\{q, p\}$ is informative for the formula $pVq$ is demonstrated by the sequence

$$\{q, pVq\}, \{q, pVq\}, \{p, q, pVq\}, \emptyset$$
3. Informative Normal Form

For monitoring prefixes we employ a tableau based construction of an automaton that represents ‘all alternative proofs’ of $\varphi$

The construction is based on an ‘Informative Normal Form’ which yields a (sufficiently large) set of sets of formulas that are all locally informative

<table>
<thead>
<tr>
<th>Case</th>
<th>$\Theta \cup {(New \cup {\psi}, Old)}$ REDUCES TO:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\psi = false$</td>
</tr>
<tr>
<td>2</td>
<td>$\psi = true$</td>
</tr>
<tr>
<td>3</td>
<td>$\psi = p$</td>
</tr>
<tr>
<td>4</td>
<td>$\psi = \neg p$</td>
</tr>
<tr>
<td>5</td>
<td>$\psi = \psi_1 \lor \psi_2$</td>
</tr>
<tr>
<td>6</td>
<td>$\psi = \psi_1 \land \psi_2$</td>
</tr>
<tr>
<td>7</td>
<td>$\psi = \circ\psi'$</td>
</tr>
<tr>
<td>8</td>
<td>$\psi = \psi_1 \cup \psi_2$</td>
</tr>
<tr>
<td>9</td>
<td>$\psi = \psi_1 \lor \psi_2$</td>
</tr>
</tbody>
</table>
3. Example

To prove the formula \((p \lor q) \lor r\) \ldots
3. Example

\[(p \lor q) \lor r^*, \quad p \lor q, \quad (p \lor q) \lor r^*, \quad r,\]

\[\ldots \text{one could try to prove } p \lor q \text{ or try to prove } r \ldots\]
3. Example

...(p \lor q) \lor r^*,
(p \lor q)^*, p
(p \lor q)^*, q
(p \lor q)^*, r^*

... and for \( p \lor q \) one could prove \( q \) or \( p \)
3. Example (Temporal Informativeness)

In case we prove $p$ for the current prefix, we still need to prove $p \cup q$ for the remainder of the prefix.

In the other cases, we are done.
3. Informative Prefixes
4. The Tableau Algorithm

\[
\begin{align*}
Q_0 & := NF(\varphi), \ Q := \emptyset, \ New := Q_0, \ \delta := \emptyset \\
\text{while } New \neq \emptyset \text{ do} \\
& \quad \text{Let } \Phi \in New \\
& \quad \text{New} \ := \text{New} \setminus \{\Phi\} \\
& \quad Q := Q \cup \{\Phi\} \\
& \quad \text{for every } \Phi' \in NF(\text{Next}(\Phi)) \text{ do} \\
& \quad & \quad \delta := \delta \cup \{(\Phi, \Phi')\} \\
& \quad & \quad \text{if } \Phi' \notin Q \text{ then } New := \text{New} \cup \{\Phi'\} \\
& \quad \text{od} \\
& \text{od}
\end{align*}
\]
4. Example

Tableau of the formula $p \lor q$

That the state sequence $\{q\}\{q\}\{q, p\}$ is informative for the formula $p \lor q$ is demonstrated by the sequence

$\{q, p \lor q\}, \{q, p \lor q\}, \{p, q, p \lor q\}, \emptyset$

a run on the transition system

/department of electrical engineering
4. Informative Tableaux

In general, the labelled transition systems produced by the informative tableau algorithm have the following properties:

- A prefix is informatively good iff it matches a run ending in the location $\emptyset$.

- A prefix is informatively bad iff it has no run on the transition system.
5. Constructing Monitors for Runtime Verification

- Deterministic and incremental
- On-the-fly determinisation
- Localise updates (stutter-closed formulas)
- Instrumenting programs and models
- Specification language for properties and atomic propositions
6. Conclusions and Future Work

- Tableau based construction of monitors for linear temporal logic properties
- For the incremental analysis of prefixes
- Prefix is counterexample and path through automaton the corresponding proof
- A single monitor for good and bad prefixes
- Checking for all (also non-informative) good (bad) prefixes
- Implementation and timed temporal logic monitoring