



A Temporal Logic For Network Flow Analysis

Tim Shimeall
tjs@cert.org



NO WARRANTY

THIS MATERIAL OF CARNEGIE MELLON UNIVERSITY AND ITS SOFTWARE ENGINEERING INSTITUTE IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this presentation is not intended in any way to infringe on the rights of the trademark holder.

This Presentation may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

Overview

Motivation

Temporal Logic

Application to Flow

Example

Implementations

Motivation

Clarify timing relationships

Formalize analysis semantics

- Clearer discussions
- Enhance automation & frameworks
- Combining analyses

Avoid over-specification of timing

Support reasoning about analysis tasks

Access temporal logic methods

Temporal Logic

Logic with explicit inclusion of time

Classically, first-order logic, could be any logic form

Temporal interpretation: Instantiating circumstances

- Linear time with rollback on contradiction
- Branching time with branch termination on contradiction
- Advantage to linear: simpler structure, no worry over paths
- Advantage to branching: can express path-related conditions

Temporal Logic Operators

$\text{Next}(t,p)$ – p is true in the instant after t

$\text{Global}(p)$ – p is true independent of time

$\text{Following}(t, p)$ – p is true at some instant after t

$\text{Until}(t,p,q)$ – p is true at each instant after t until q is true

$\text{Forall}(p)$ – p is true along all paths

$\text{Exists}(p)$ – p is true along at least one path

Adaptation to Flow

Description first, then reasoning

Iterative semantics – suitable for filter-like processing

Specific semantics:

- 5-tuple
- Ordinal time (inexact comparisons)
- Related flows

Adapted semantics

$R(f_1, f_2)$ relation – flow-flow connection

$p(f, \dots)$, $q(f, \dots)$ – logic predicates on flow records/fields

Enable reasoning using Horn clause resolution and backtracking

Temporal Operators for Flow

Globally:

$G(p)$: forall($R(f,f') \rightarrow p(f)$ and $p(f')$)

Next:

$N(f,f')$: iff $R(f,f')$ and $f'.stime > f.stime$ and
does not exist (
 $R(f,f'')$ and $f'.stime > f''.stime > f.stime$)

$N^*(f,f')$: transitive relation on N

$X(f,p)$: forall($N(f,f') \rightarrow p(f')$)

Following:

$F(f,p)$: exists($N^*(f,f')$ and $p(f')$)

Until:

$U(f,p,q)$:
exists ($N^*(f,f'')$ and $q(f'')$),
forall ($N^*(f,f')$ and $f''.stime > f'.stime \rightarrow p(f')$ and not $q(f')$)

Descriptive Temporal Example

Spam(s,f):

$R(f,f')$: $f.sip = f'.sip = s$ and s not on *whitelist*

If and only if

$|\{f', \text{Following}(f,f', f'.stime < f.stime + 5min \text{ and } f'.dport = email)\}| > 15$ and

$|\{f', \text{Following}(f,f', f'.stime < f.stime + 5min \text{ and } f'.dport = email)\}| \geq$

$|\{f', \text{Following}(f,f', f'.stime < f.stime + 5min)\}| * 0.1$

Implementation

Use temporal logic to express analysis criteria

Prolog-based (GNU-Prolog)

- Logic programming, incorporating time in resolution

- Initial prototype to refine semantics

- Construct interface to analysis tools (plugin)

Python-based (PySiLK)

- Declarative programming, incorporate limited resolution mechanism

- Secondary prototype to demonstrate applicability

Eventually construct reasoning rules for analysis relationships or proof

Conclusions

Temporal logic adaptation of flow analysis offers opportunity to encompass large literature of pre-existing methods

Formalization of time relationships offers opportunity to improve flow analysis methods

More formal reasoning on flow analysis?