A Temporal Logic For Network Flow Analysis

Tim Shimeall
tjs@cert.org

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

Next(*t*,*p*) – *p* is true in the instant after *t*

Global(*p*) – *p* is true independent of time

Following(*t*, *p*) – *p* is true at some instant after *t*

Until(*t*,*p*,*q*) – *p* is true at each instant after *t* until *q* is true

Forall (p) – p is true along all paths

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,…), q(f,…) – 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') → p(f) and p(f'))

Next:

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

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

X(f,p): forall(N(f,f') →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 → 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', Following(f,f',f'.stime < f.stime+*5min* and f'.dport=*email)*}|>15 and

|{f', Following(f,f', f'.stime < f.stime+*5min* and f'.dport=*email)*}| ≥ |{f', Following(f,f', f'.stime < f.stime+5*min*)}| * 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?