Automatic Web Application Verification

Ivaylo Gochkov

December 1, 2005

Contents

1. Introduction
2. AnWeb
3. Related projects
4. Conclusion

Introduction

Web applications

- Widely used
- The complexity grows rapidly
- New tools are needed for:
  – (partial) verification - proving that satisfies the specification
  – testing - comparing the actual value with expected one
  – analysis - checks predefine set of the specification
- Model checking - verifying it the model satisfy a formal specification (usually written as temporal logic formulas)

Web Site/Application

Web Application

A *dynamic* collection of HTML pages connected with hyperlinks.

Dynamic page

Generated by client or server side script depending on a user input.

Web Site/Application

Web Application

A *dynamic* collection of HTML pages connected with hyperlinks.

Dynamic page

Generated by client or server side script depending on a user input.

Web Site

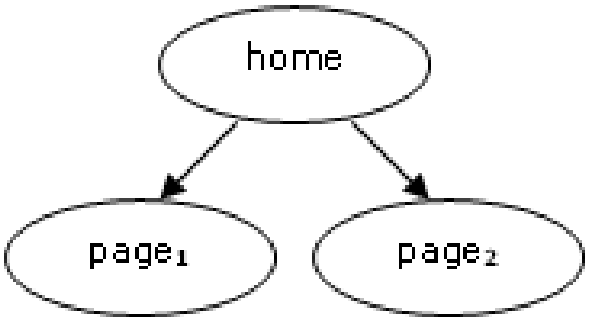
A *static* collection of HTML pages connected with hyperlinks.

Static page

The content is not dependent of a user input.

Flat Model

- Common starting point of many researches for representing a *Web Site*.
- Directed graph $G = (N,\ E)$ where
  N – set of nodes representing web pages
  E – set of edges representing the links between the pages



$$N = \{home,\ page_1,\ page_2\}$$
$$E = \{(home,\ page_1),\ (home,\ page_2)\}$$

General properties of static Web Site

- Reachability: broken links, unreachable pages, return to home page
- Non-hierarchical frame content: nested frame sets
- Longest/shortest paths: number of links, number of downloaded bytes

General properties of Web Applications

- Security: pages reachable with lower permissions
- Access control: personal data of the users, accessed by non-authorized users

General

- Tool for automatic support in the design of Web Applications
- Created in Politecnico di Bari, Italy
- 1st prototype - Java application on Windows
- 2nd prototype - PHP application on Linux
- Provides an interface to NuSMV (Symbolic Model Checker)
- Status of the research - handles static and dynamic pages

Modeling of Web Applications

Directed graph $G = (N,\ E)$ where

N – set of web pages and hyperlinks

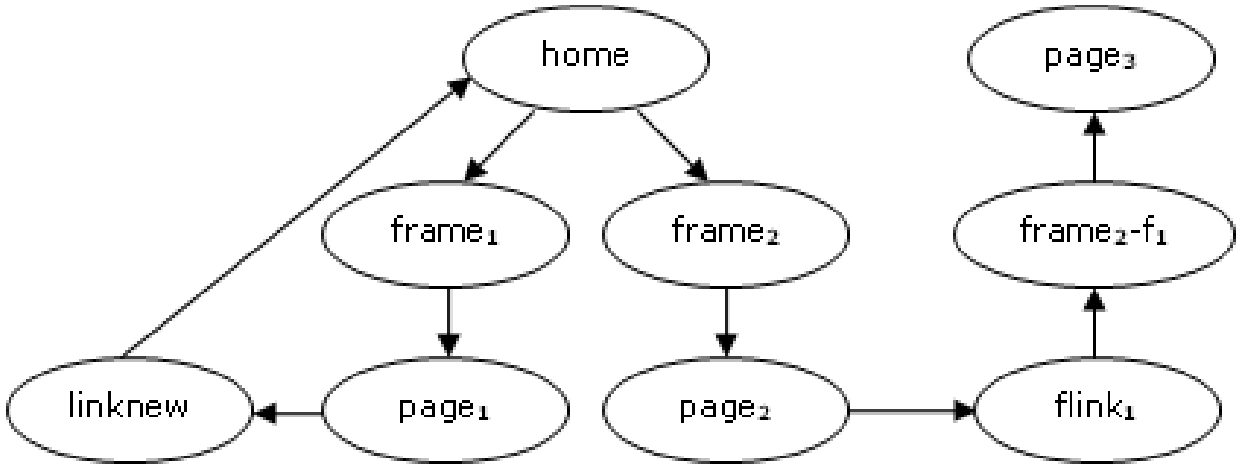
E – relations between page/link and link/page

Explicit distinguishing between pages and links - more detailed properties in the temporal logic

Hyperlinks: links with targets, frames, anchors

Labeling the nodes:
page, *link*, *frame*, *anchor*
linknew, *flink* - special case of hyperlinks

Example of modeling static Web Site



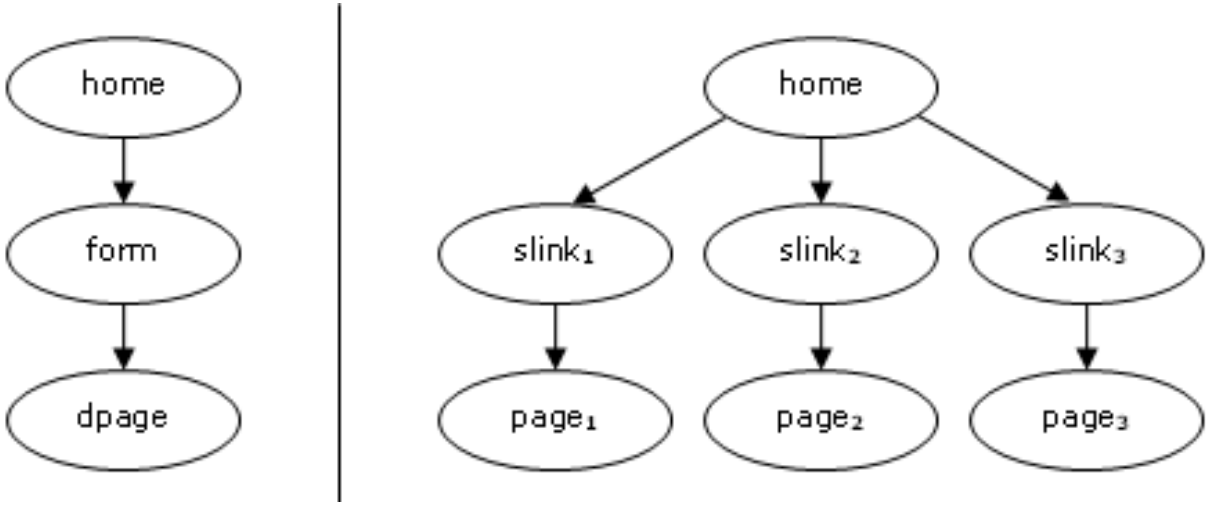
Modeling scripts

Client side scripts - as static pages

Server side scripts - as dynamic pages

Two types of dynamic pages:
– redirection to pages depending on user input
– generated, getting data from database depending on user input

Example of modeling dynamic redirection



NuSMV

Open source tool for symbolic model checking developed in University of Trento

Extends SMV which is based on BDD (Binary Decision Diagrams - directed, acyclic graphs)

Created for verification of industrial designs

Widely used also in the software domain

Own language for finite-state concurrent systems

Syntax

Computation Tree Logic (CTL)

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \rightarrow \psi \mid \phi \leftrightarrow \psi$$
$$\mid EF\phi \mid EX\phi \mid EG\phi \mid E(\phi U \psi) \mid AG\phi \mid AF\phi \mid AX\phi \mid A(\phi U \psi)$$

- p is propositional atom
- ϕ and ψ are CTL formulas
- E (Exists) and A(Always) - path quantifiers
- X (neXt time), U (Until), G (Globally) and F (Future) - temporal operators

Semantics

Computation Tree Logic (CTL)

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \rightarrow \psi \mid \phi \leftrightarrow \psi$$
$$\mid EF\phi \mid EX\phi \mid EG\phi \mid E(\phi U \psi) \mid AG\phi \mid AF\phi \mid AX\phi \mid A(\phi U \psi)$$

- $EF\phi$: a path exists such that ϕ holds in some future state

Semantics

Computation Tree Logic (CTL)

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \rightarrow \psi \mid \phi \leftrightarrow \psi$$
$$\mid EF\phi \mid EX\phi \mid EG\phi \mid E(\phi U \psi) \mid AG\phi \mid AF\phi \mid AX\phi \mid A(\phi U \psi)$$

- $EF\phi$: a path exists such that ϕ holds in some future state
- $EX\phi$: ϕ holds in some next state

Semantics

Computation Tree Logic (CTL)

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \rightarrow \psi \mid \phi \leftrightarrow \psi$$
$$\mid EF\phi \mid EX\phi \mid EG\phi \mid E(\phi U \psi) \mid AG\phi \mid AF\phi \mid AX\phi \mid A(\phi U \psi)$$

- $EF\phi$: a path exists such that ϕ holds in some future state
- $EX\phi$: ϕ holds in some next state
- $EG\phi$: a path exists such that ϕ holds globally along the path

Semantics

Computation Tree Logic (CTL)

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \rightarrow \psi \mid \phi \leftrightarrow \psi$$
$$\mid EF\phi \mid EX\phi \mid EG\phi \mid E(\phi U \psi) \mid AG\phi \mid AF\phi \mid AX\phi \mid A(\phi U \psi)$$

- $EF\phi$: a path exists such that ϕ holds in some future state
- $EX\phi$: ϕ holds in some next state
- $EG\phi$: a path exists such that ϕ holds globally along the path
- $E(\phi U \psi)$: a path exists such that ϕ holds until ψ holds

Semantics

Computation Tree Logic (CTL)

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \rightarrow \psi \mid \phi \leftrightarrow \psi$$
$$\mid EF\phi \mid EX\phi \mid EG\phi \mid E(\phi U \psi) \mid AG\phi \mid AF\phi \mid AX\phi \mid A(\phi U \psi)$$

- $EF\phi$: a path exists such that ϕ holds in some future state
- $EX\phi$: ϕ holds in some next state
- $EG\phi$: a path exists such that ϕ holds globally along the path
- $E(\phi U \psi)$: a path exists such that ϕ holds until ψ holds
- A - analogically: for all paths

Example verification properties written as CTL formulas:

Broken links

$\phi_1 \;=\; AG(link \;\rightarrow\; EXpage)$

Unreachable pages

$\phi_2 \;=\; AG(home \;\rightarrow\; EFpage_1) \;\wedge\; \ldots \;\wedge\; AG(home \;\rightarrow\; EFpage_n)$
where $page_1 \ldots page_n$ the pages in the *Web Application*.

Return to the home page

$\phi_3 \;=\; AG(page_1 \;\rightarrow\; EFhome) \;\wedge\; \ldots \;\wedge\; AG(page_n \;\rightarrow\; EFhome)$
where $page_1 \ldots page_n$ the pages in the *Web Application*.

ReWeb

Tool for web site analysis

Reverse engineering on web sites

Download, analyze simple properties of static pages

Interface for searching and navigation in the results

Evolution of web site

Components

- Splitter - responsible for downloading and saving version of a web site (Java)
- Analyzer - apply different analyzes on the downloaded version (Java)
- Viewer - displays the results in GUI (Dotty - customizable graph editor)

Web Site modeling

$G = (N, E)$

$N = N_1 \cup N_2 \cup F$

where:

N_1 – all web pages

N_2 – set of frames for one web page

F – all frames

$E = E_1 \cup E_2 \cup E_3$

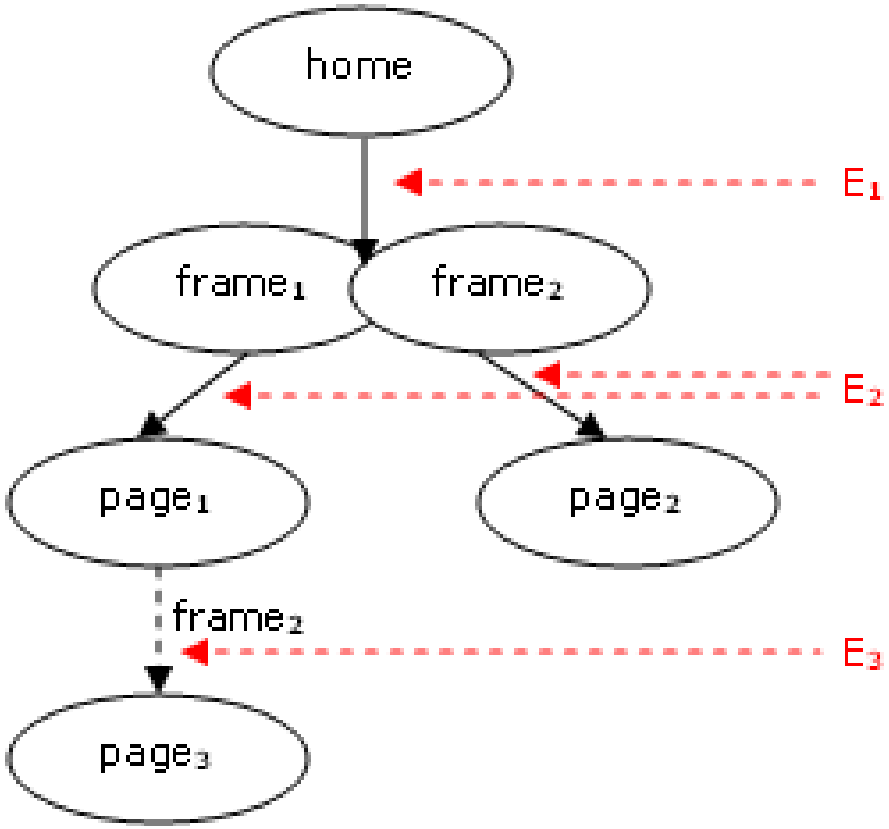
where:

E_1 – hyperlinks between pages or relations from page to frames

E_2 – relations from frames to pages

E_3 – labeled relations from page to particular frame

Example modeling



Analyses

Guess

The analysis are coded in Java. No language like CTL for writing verification properties was described in the paper.

- reaching frames - set of frames in with the page may appear
- dominators - the pages in the path to a page (index.html is dominator to all other pages)
- shortest path - minimum number of pages to target page
- pattern matching - matching a library with patterns against given web site. The patterns are graphs themselves. Example 'full connectivity': each node connected to all other nodes.

MCWEB - Model Checking the Web

Developed in University of California at Barkeley

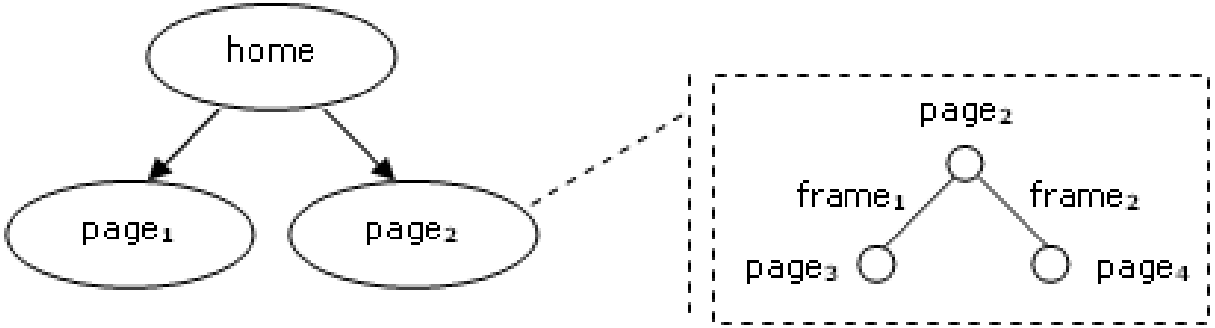
Tool for web site verification

Constructive μ-calculus (CμC) for writing the verification properties

Checks connectivity, frame and cost properties

Modeling

- Extension of the nodes to *WebNodes* (a tree representing a frame structure)
- The nodes in the tree represent pages
- The branches a labeled with the frame name



Conclusion

AnWeb, MCWEB

- Step in the right direction - model checkers
- Language for writing verification properties

AnWeb – models WA based on parsing sources
MCWEB – models WA based on downloaded version

ReWeb

- Simple predefined analysis
- No language like CTL
- Handles only static Web Sites

Questions