

A Formal Approach for Run-Time Verification of Web Applications using Scope-Extended LTL

May Haydar

Supervisor: Prof. Houari Sahraoui

MoSART

June 19, 2008

Outline

- Motivation
- Problem Statement
- Formal Modeling of Web Applications
- Extending LTL with Propositional Scopes
- Implementation and Evaluation
- Conclusions and Future Work

Motivation

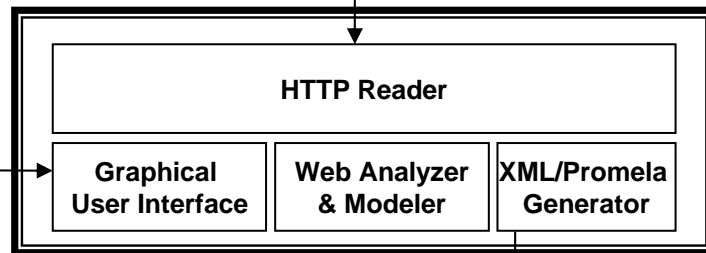
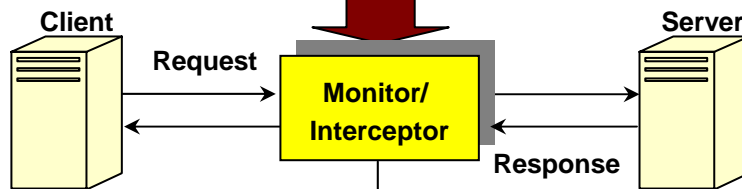
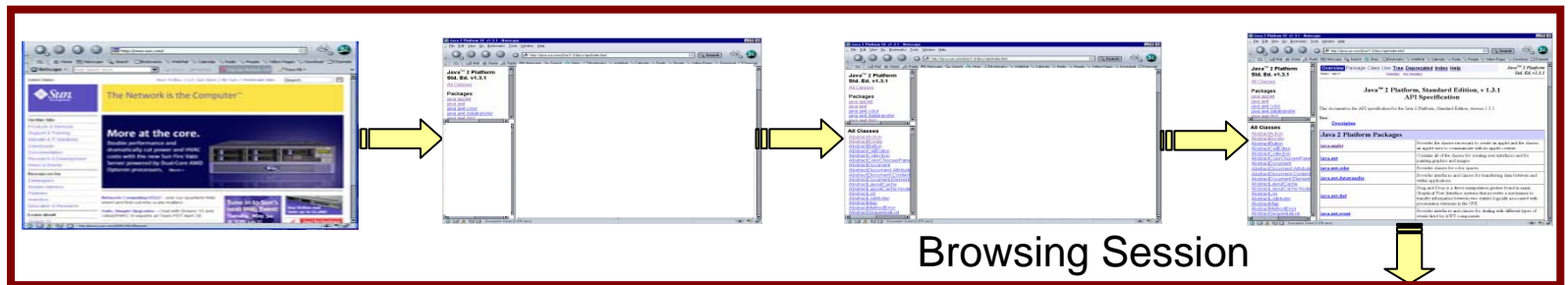
- Steep and radical growth of web applications
 - n-tiered applications, diverse types of users, increasingly complex and critical to everyday life
- Existing tools test various aspects of web applications
 - HTML syntax (HTML TIDY), Hyperlink integrity (LinkCheck)
 - Java code testing (JCover), Performance (MSACT)
- Principle limitation of existing tools is their incapability of verifying properties in executions of WAs
- Need for analysis and validation methods and tools to ensure high quality of web applications
- Formal methods are increasingly used in system verification

Problem Statement and Solution

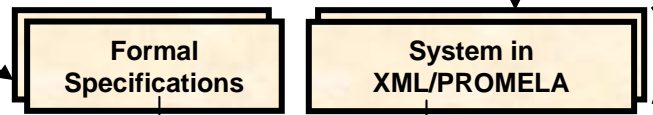
- Given a Web Application Under Test (WAUT) with no access to the code
 - Define properties related to specifications of web applications
 - Verify whether the WAUT satisfies the properties

- Solution
 - Infer automata based models from WAUT
 - Build library of user defined properties
 - alleviate the need to learn the foundations of temporal logic for property specification
 - Use of a model checker to verify if the WAUT satisfies properties

Formal Framework

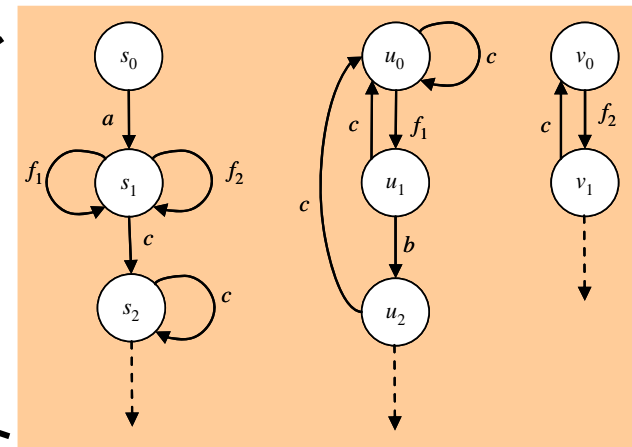


Properties to Check



Property Results:
1. Satisfied
2. Violated

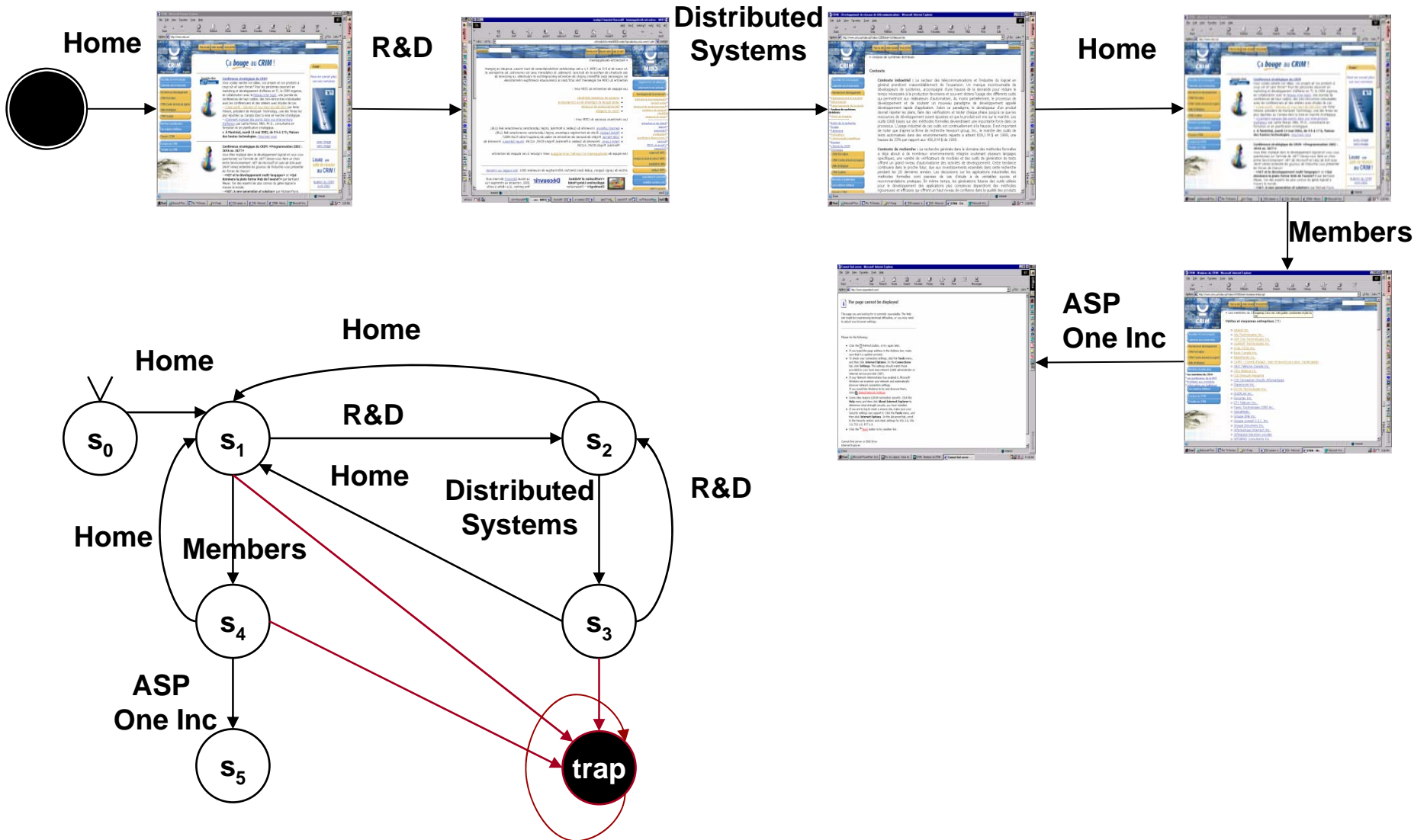
Communicating Automata Model



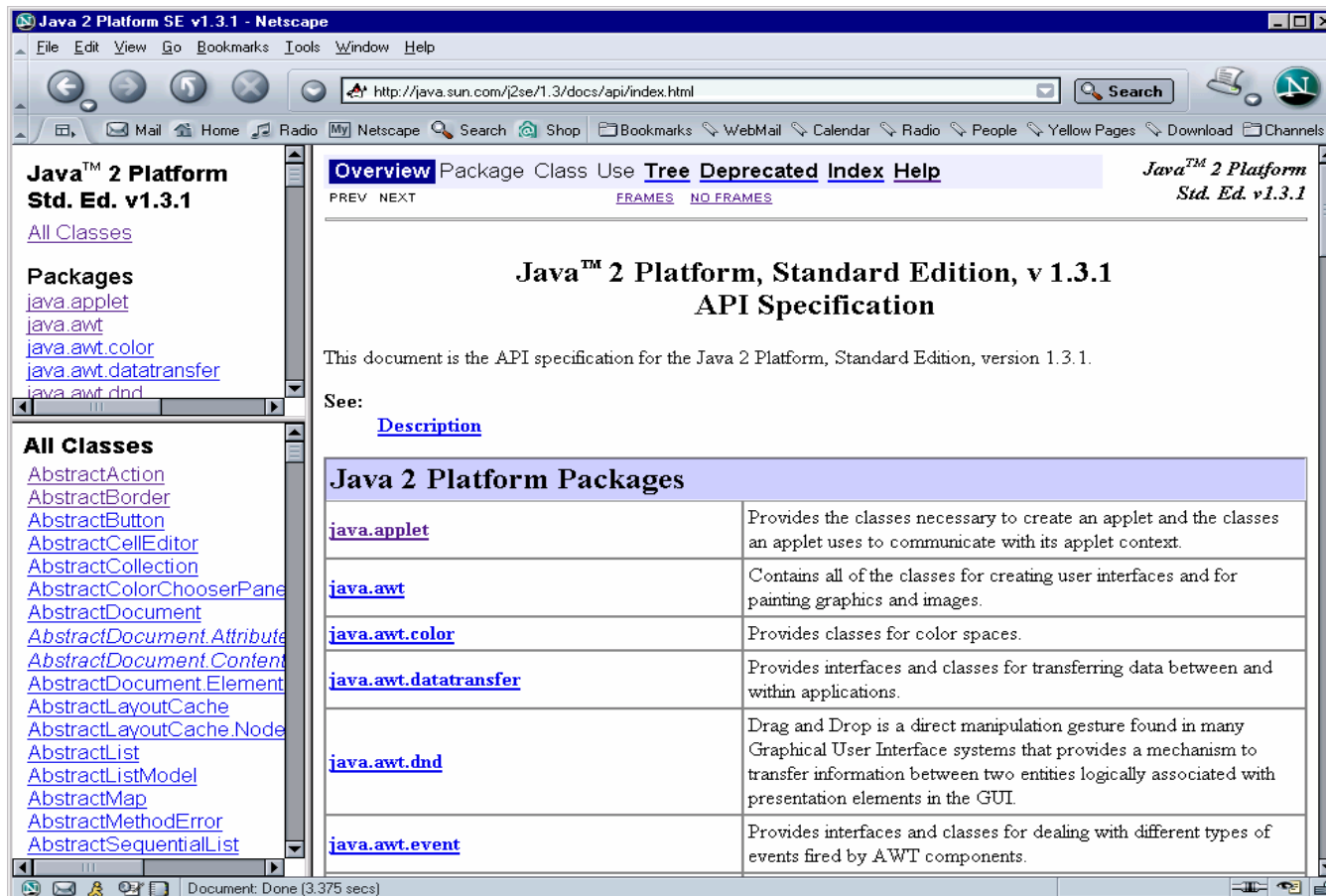
Run-time Observation of WA

- Log a trace of HTTP requests and responses - Request Response Sequence (RRS)
 - Requests for links clicked and forms submitted
 - Responses are HTML content of static/dynamic pages
- Links and forms could be repeated on different pages
- Model a RRS by an automaton
 $A_{RRS} = \langle S \cup \{trap\}, s_0, \Sigma, T \rangle$ called session automaton

Example



Modeling Multi Display WAs

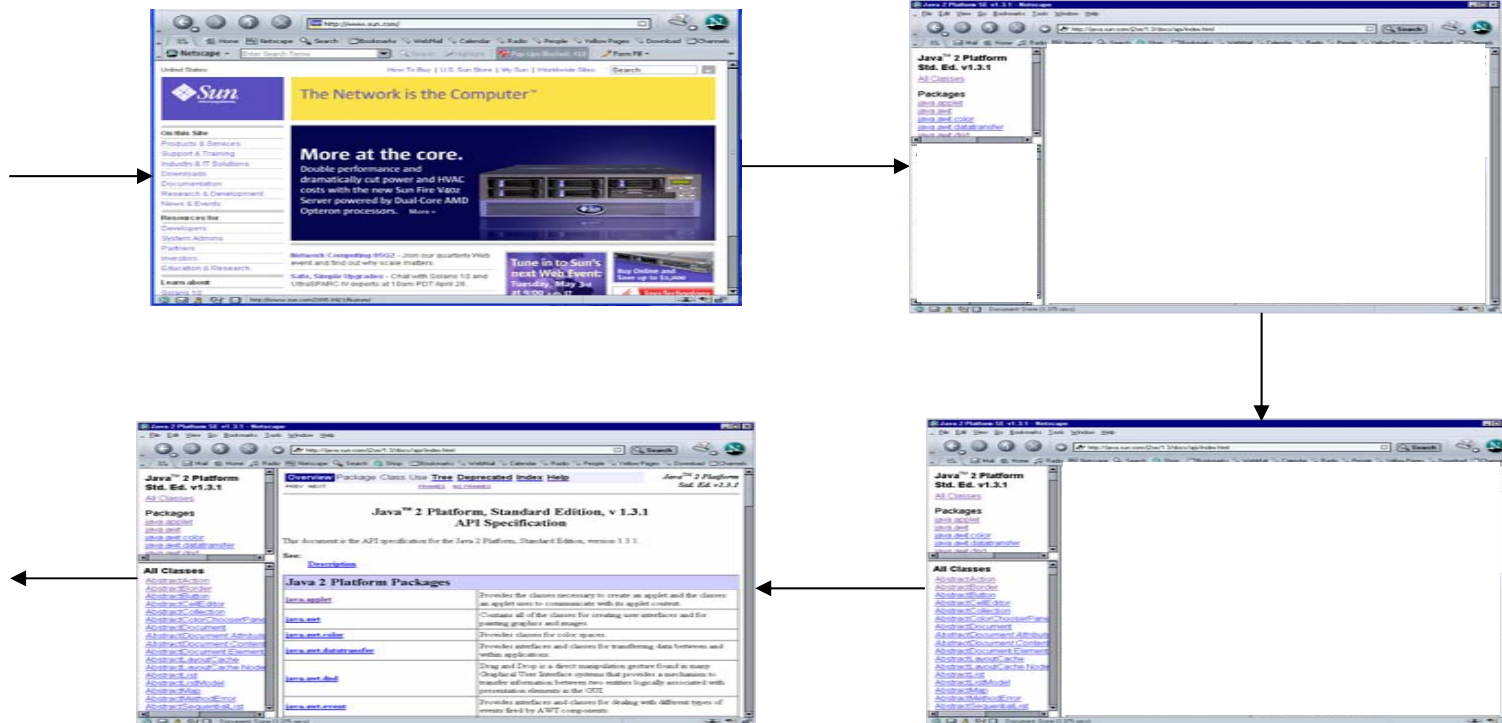


- From browsing session
 - 1 main window, 3 frames
- 4 local sessions

Local Session Communicating Automata

- An automaton represents a window or a frame
- The set of events is extended by URIs of frame source pages
- Rendezvous communication on common events
 - Links and filled forms with targets are common events in automaton of the source and target entities
 - Common events are enabled in all the states of the automata of the target entity

Stable – Transient Displays

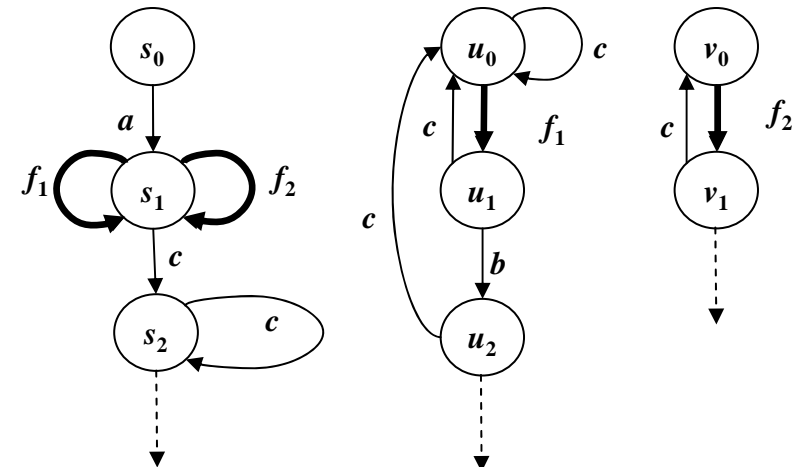
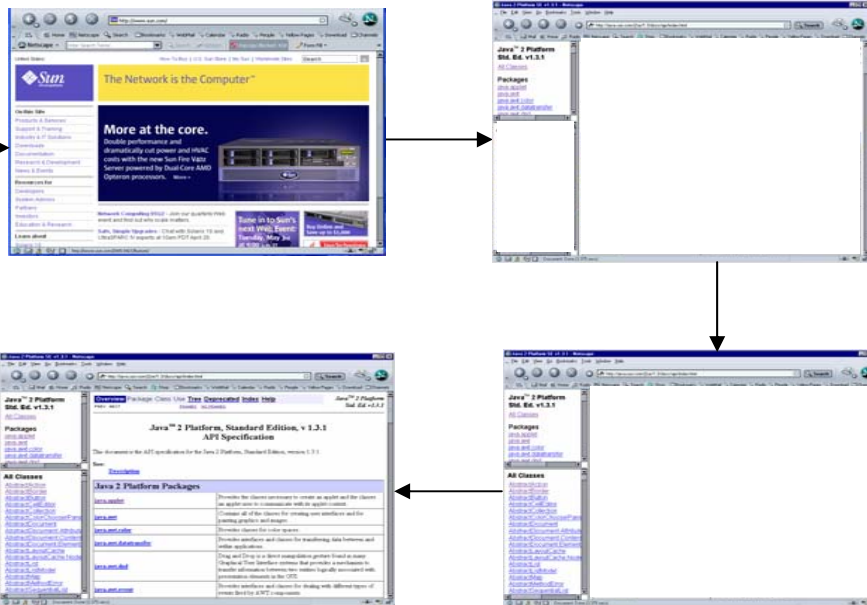


Some properties need to be checked only in stable displays

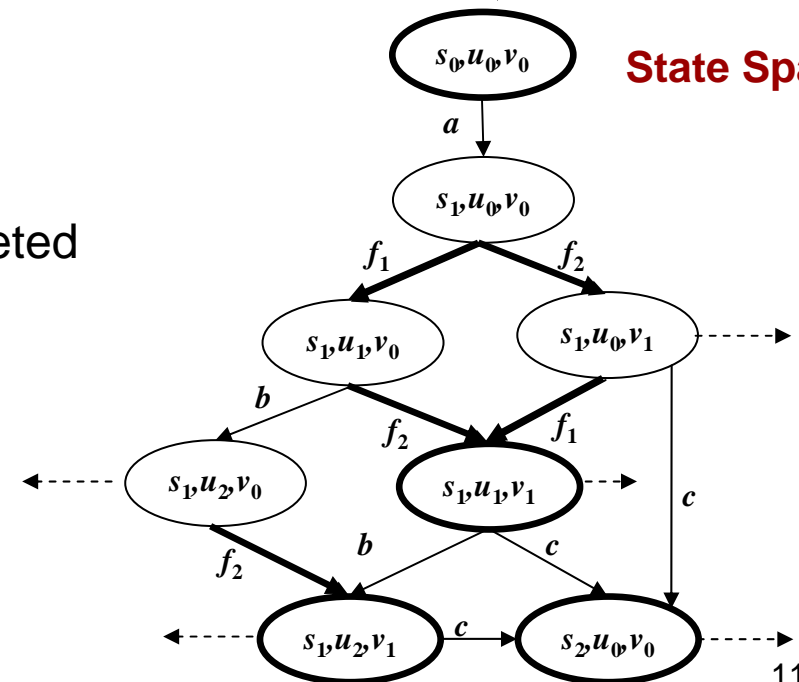
- *An important advertisement should be present in every display shown to the user of a WA with (at least) two frames*
- *The number of links shown to the user should always be greater than a predefined threshold*

Stable – Transient Global States

System Model



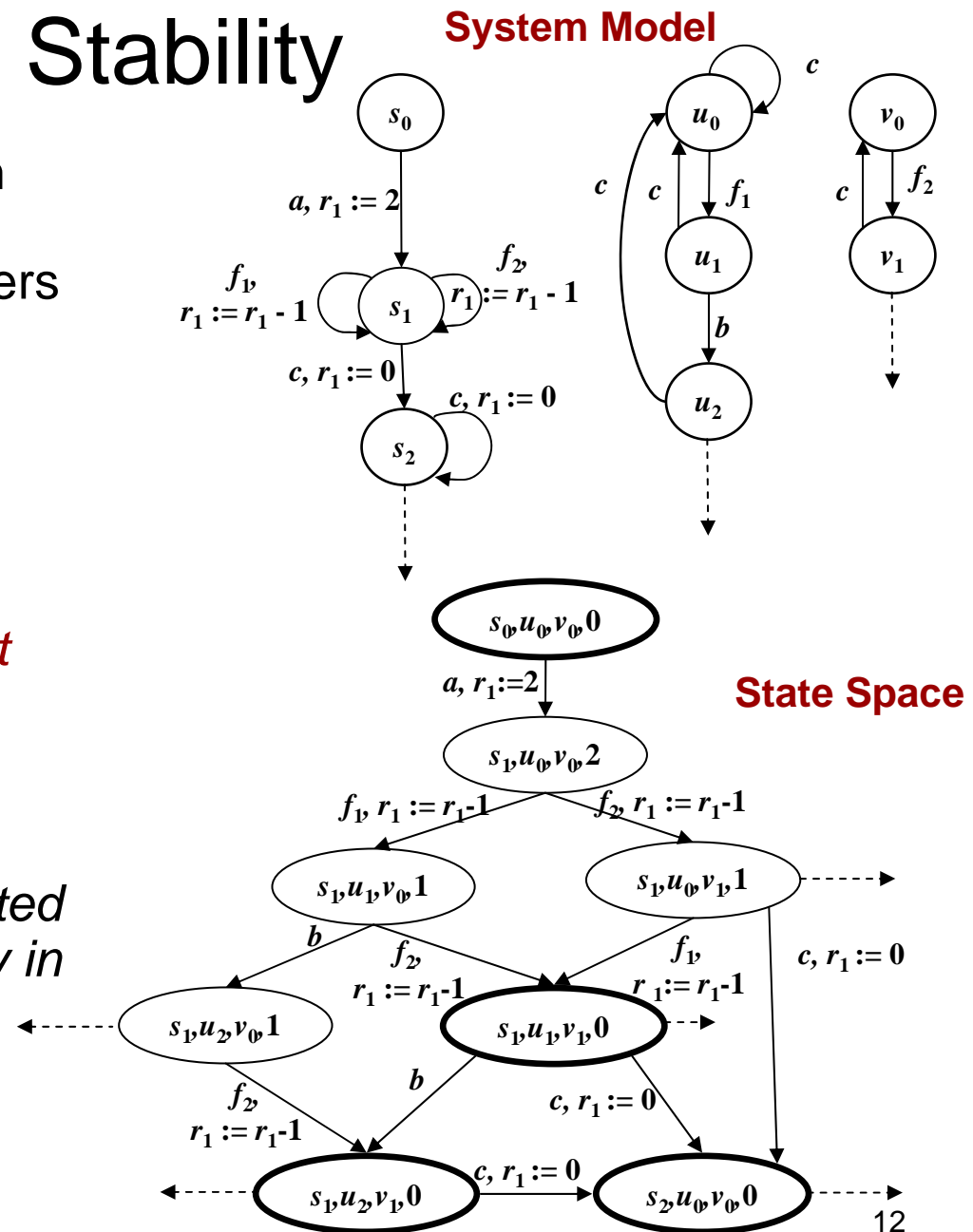
State Space



- In multi display web applications
 - Stable state corresponds to completed display
 - Transient state corresponds to incomplete display
- User actions: links, forms
- Browser actions: URLs of frame default pages

Determining State Stability

- Need for means to distinguish stable and transient states recognizable by model checkers
- Devise generic algorithm to extend the communicating automata with local variables and updates
- Each automaton has an *event tracking* variable to count *designated events* enabled at each state
- *Designated events are executed once* \Rightarrow System does not stay in transient mode



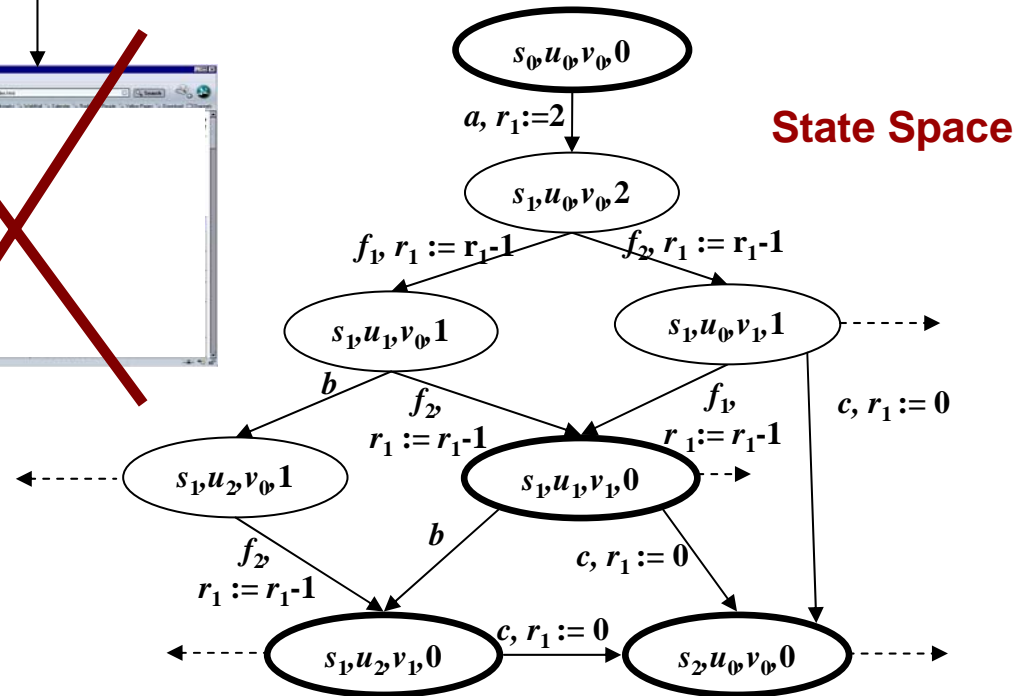
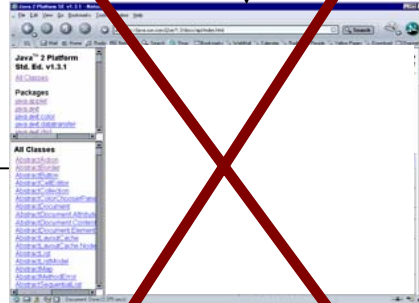
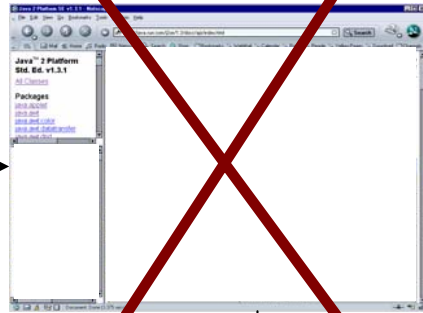
LTL is not Enough

- Specifying properties specification is difficult for novices and professionals
 - Lack of complete specifications
 - Limitations of the language
- Web application model may include states uninteresting to certain properties
- How to specify LTL properties checkable in an arbitrary subset of the state space

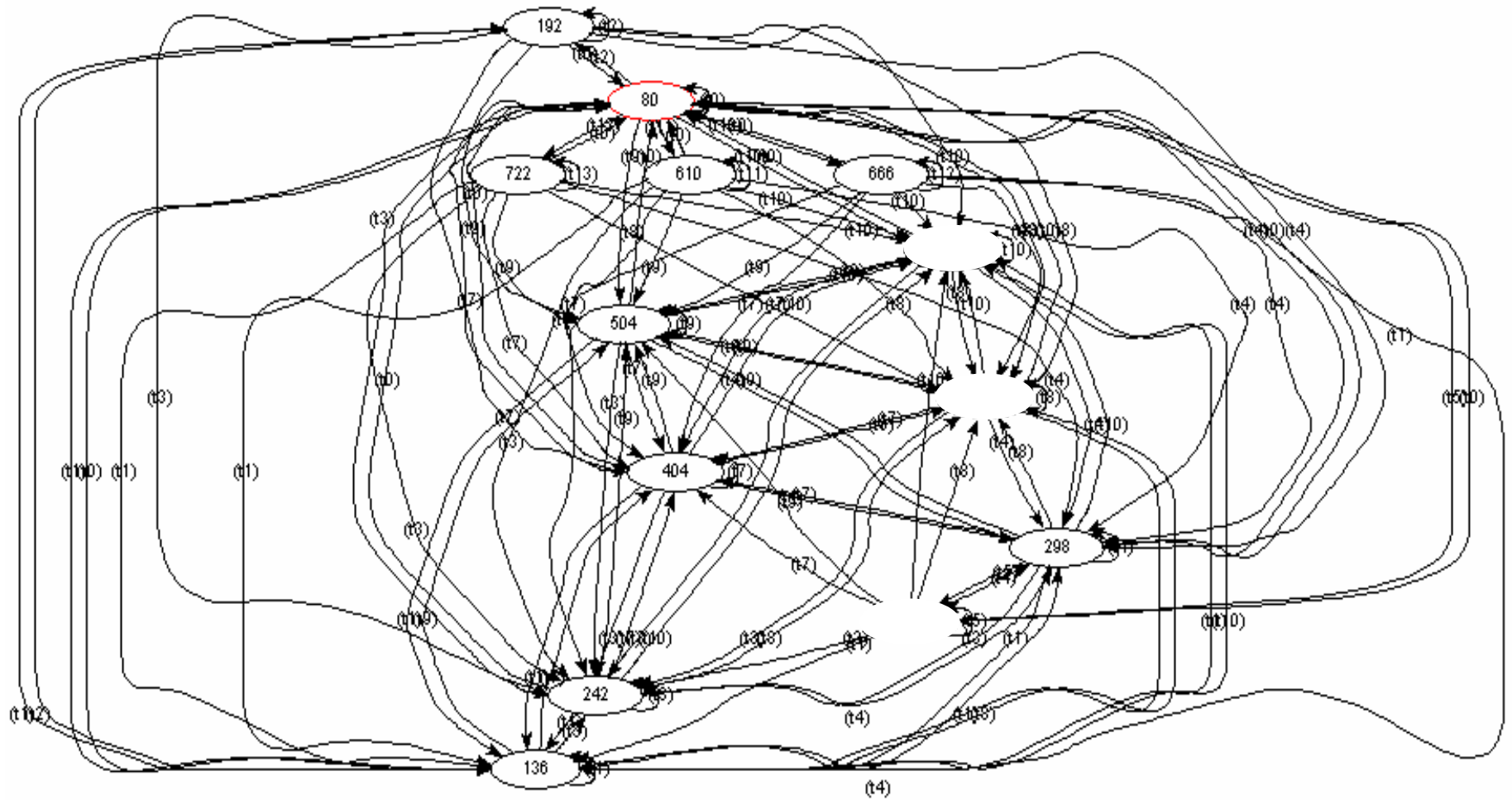
Intuitive solution

remove the uninteresting states from the model

Example



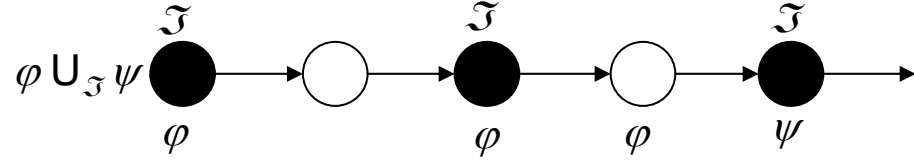
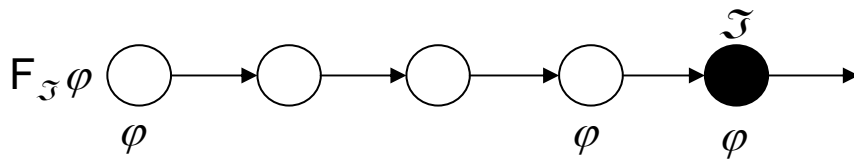
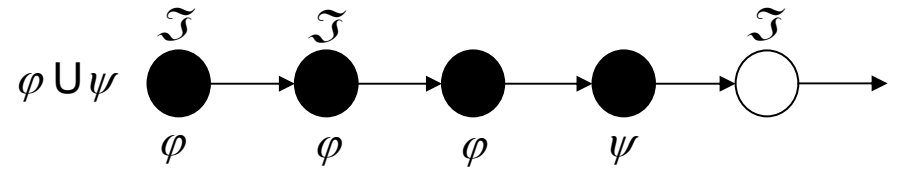
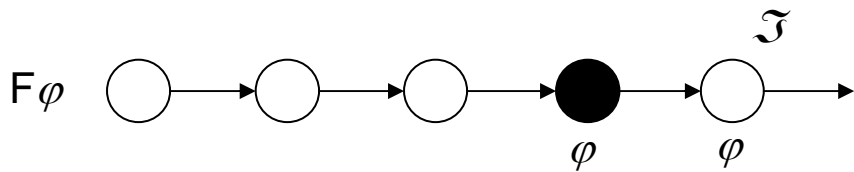
Example



Extending LTL with Scopes

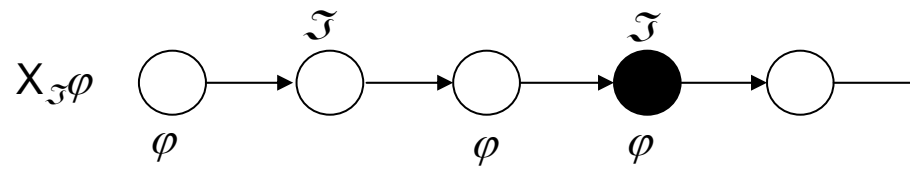
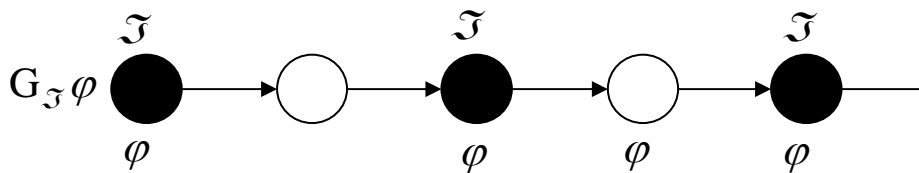
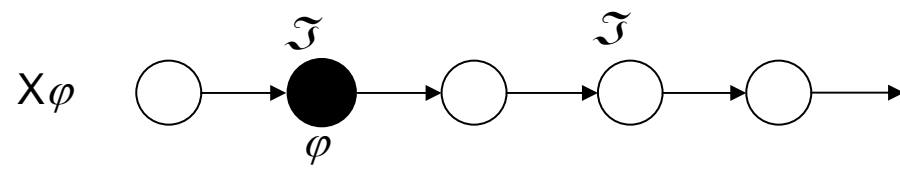
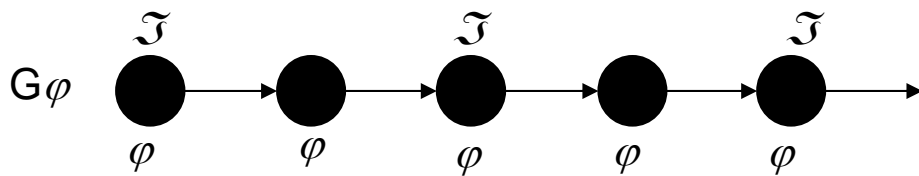
- Define a means to specify LTL properties over arbitrary subsets of the state space
 - Do not add to the complexity of specification
 - Do not modify the model
 - Model checking algorithms still can be used
- Solution
 - Introduce scopes to define subsets of the search state space
 - Add new operators to LTL to check properties within defined scopes
- Result in succinct formulae

LTL with Propositional Scopes



$$F_{\mathcal{I}}\varphi = F(\varphi \wedge \mathcal{I})$$

$$\varphi U_{\mathcal{I}}\psi = (\mathcal{I} \rightarrow \varphi) U (\psi \wedge \mathcal{I})$$



$$G_{\mathcal{I}}\varphi = G(\mathcal{I} \rightarrow \varphi)$$

$$X_{\mathcal{I}}\varphi = \neg\mathcal{I} U [\mathcal{I} \wedge X(\neg\mathcal{I} U (\mathcal{I} \wedge \varphi))]$$

Scope Operator **In**

- Recursively applies propositional scope on any formula
- Makes property specification intuitive and succinct

$$p \text{ In } \mathcal{S} = \neg \mathcal{S} \cup (p \wedge \mathcal{S})$$

$$(\neg \varphi) \text{ In } \mathcal{S} = \neg_{\mathcal{S}}(\varphi \text{ In } \mathcal{S})$$

$$(\varphi \wedge \psi) \text{ In } \mathcal{S} = (\varphi \text{ In } \mathcal{S}) \wedge_{\mathcal{S}}(\psi \text{ In } \mathcal{S})$$

$$(\varphi \cup \psi) \text{ In } \mathcal{S} = (\varphi \text{ In } \mathcal{S}) \text{ In }_{\mathcal{S}}(\psi \text{ In } \mathcal{S})$$

$$(G \varphi) \text{ In } \mathcal{S} = G_{\mathcal{S}}(\varphi \text{ In } \mathcal{S})$$

$$(F \varphi) \text{ In } \mathcal{S} = F_{\mathcal{S}}(\varphi \text{ In } \mathcal{S})$$

$$(X \varphi) \text{ In } \mathcal{S} = X_{\mathcal{S}}(\varphi \text{ In } \mathcal{S})$$

Example

*The user accesses a **Secure page** only twice and each time after a visit to **Authentication page***

- Standard LTL

$$(\neg \text{Authentication} \wedge \neg \text{Secure}) \text{ U } (\text{Authentication} \wedge \neg \text{Secure} \wedge \text{X} (\neg \text{Secure} \text{ U } (\text{Secure} \wedge \neg \text{Authentication} \wedge \text{X} (\neg \text{Authentication} \text{ U } (\text{Authentication} \wedge \neg \text{Secure} \wedge \text{X} (\neg \text{Secure} \text{ U } (\text{Secure} \wedge \neg \text{Authentication} \wedge \text{X} (\neg \text{Authentication} \text{ U } (\text{Authentication} \wedge \neg \text{Secure} \wedge \text{X} (\text{G} (\neg \text{Secure})))))))))))))$$

- Using **In** operator

- Scope is $(\text{Authentication} \oplus \text{Secure})$ where \oplus is the exclusive or operator
- $\text{Authentication} \wedge \text{X}(\text{Secure} \wedge \text{X}(\text{Authentication} \wedge \text{X}(\text{Secure} \wedge \text{X}(\text{Authentication} \wedge \text{G}(\neg \text{Secure})))))) \text{ In } (\text{Authentication} + \text{Secure})$

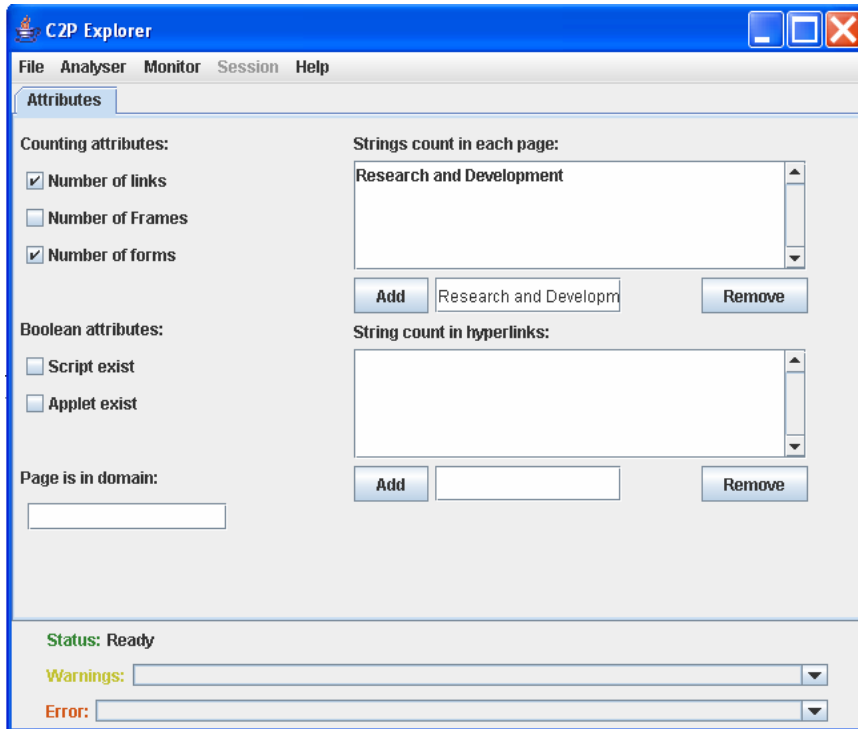
WeSPaS - Web Specification Pattern System

- Common web quality attributes include
 - Reliability and functionality, Usability, Security and privacy
- Survey of various resources: IBM, Opquast (Temesis), literature, Ph.D. work within GEODES (Ghazwa Malak)
- Identified *Custom* attribute related to stakeholders needs
 - Incorrect login is allowed only three times in a row, and then login is forbidden*
- Categorization (118 patterns)
 - Non-functional: Navigation and Links – Presentation and Content
 - Functional
 - Reachability
 - Security/Authentication and Trust
 - E-commerce (Customer Service –Product Info and Navigation – Purchase Transaction)

Which Properties are Preserved

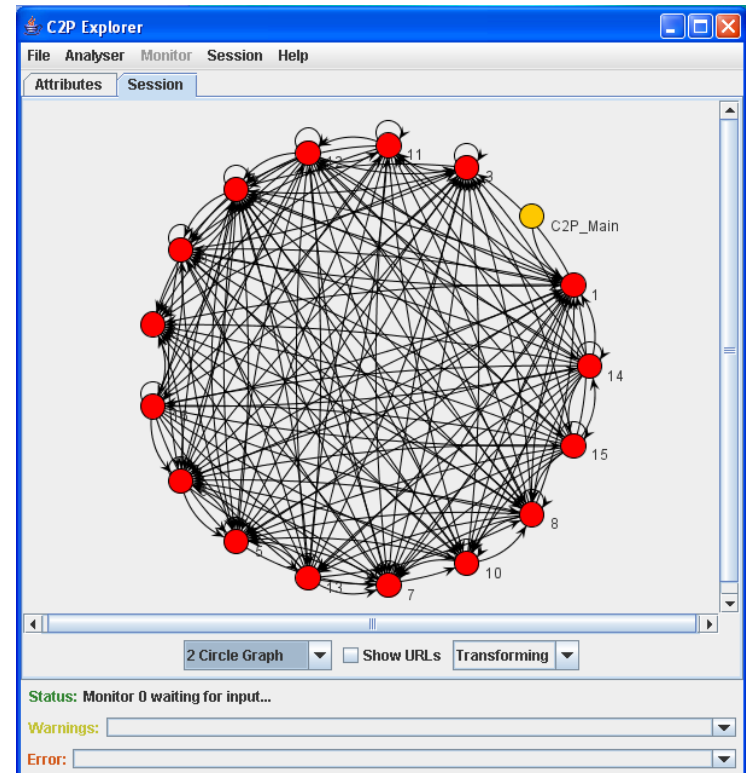
- If a property valid/invalid in session automaton can the same verdict be said about original WAUT ?
- prove that session automaton models fragment of the behavior of the WAUT
- Inverse preservation (violation) of
 - Safety properties
 - Properties whose counter examples are infinite
- Reachability checked on a single path (Limitation of LTL)
 - Ex: *user can reach a page with e-shop return policy*
 - Negate property: *on all paths e-shop return policy always absent*
⇒ *safety property*
 - If negation is invalid then original property is valid

Prototype Tool



- Analysis of execution traces and model generation
- Automata model visualization and statistical data
 - Visual
 - Textual

- Property based attribute selection
- Execution interception and monitoring



Evaluation – Tool Performance

URL	Automata Model	Time
www.crim.ca	1 automaton, 294 states, 3879 transitions	2 min, 32 sec.
www.umontreal.ca	1 automaton, 267 states, 1657 transitions	4 min, 51 sec.
www.computer.org	2 automata, 36 global states, 213 transitions	1 min.
www.concordia.ca	4 automata, 2049 global states, 28204 transitions	1 min.
www.eclipse.org	9 automata, 847 global states, 9652 transitions	1 min, 20 sec.
www.berkeley.edu	14 automata, 19802 global states, 72801 transitions	2 min.

- Automatic browsing using crawler
- Time to produce the models is relative to the following parameters:
 - server response time
 - content of web pages visited
 - number of attributes evaluated in web pages

Evaluation - Property Verification

■ Non-Functional

- *Number of links in each display (single or multi) should not exceed a certain threshold (depends on size of application).*
- *Number of links in each display (single or multi) is balanced.*
- *Combinations of certain words/objects are absent.*

■ Functional

- *Home page is reachable from every other page without going through a certain page X.*
- *Secure pages are not reachable without authentication process.*
- *Privacy policy page is reachable from every page.*

	Property										
	1.1	1.2	1.3	1.4	1.5	2.1	2.2	2.3	2.4	2.5	2.6
Single window WA	2	2	1	2	0	0	3	0	1	1	0
Multi-display WA	1	5	5	5	3	0	4	1	n/a	n/a	n/a

Conclusion and Future Work

■ Conclusions

- Formal framework for run-time verification of WA
- Extension of Linear Temporal Logic with scopes
- System of web property patterns

■ Future Work

- Reconstruction of counter examples and play back
- Partial automation of form population
- Dynamic means using XPath to specify attributes
- Extend LTL with temporal scopes
- Implementation of specification patterns
- Address emerging technologies
 - Ajax (Asynchronous Java + XML)
 - Web services composition



Questions?