Automatic Verification of Web Sites Using Partial Rewriting

M. Alpuente[1] D. Ballis[2] M. Falaschi[2]

[1] DSIC, Universidad Politécnica de Valencia, Spain.

alpuente@dsic.upv.es

[2] DIMI, Università degli Studi di Udine, Italy.

{demis,falaschi}@dimi.uniud.it

Talk plan

- Motivations
- Web site denotation and Web site specifications
- Simulation and partial rewriting
- Verification Framework
- Conclusions

Motivation

- Web sites can have a very complex structure.
- Web sites are difficult to develop and to maintain.

Motivation

- Web sites can have a very complex structure.
- Web sites are difficult to develop and to maintain.
- Suitable methodologies are needed to verify Web sites w.r.t a given scheme, which is able to express semantic as well as syntactic properties.
- Formal methods can be fruitfully employed!

Motivation

- Web sites can have a very complex structure.
- Web sites are difficult to develop and to maintain.
- Suitable methodologies are needed to verify Web sites w.r.t a given scheme, which is able to express semantic as well as syntactic properties.
- Formal methods can be fruitfully employed!
- We can verify and improve correctness of the EU-India project outcomes (Web sites)
- A workshop organized in Valencia in February 2005.
- Collaboration with Hyderabad on developments of new tools and improvements of basic framework.

Formal Verification of Web sites

We provide a rule-based specification language for

- specifying integrity conditions for a given Web site.

Formal Verification of Web sites

We provide a rule-based specification language for

- specifying integrity conditions for a given Web site.

And a verification technique for

- automatically checking whether those conditions are fulfilled.
- helping to repair faulty Web sites.

Our framework is based on a rewriting-like technique (which is called partial rewriting), more suitable for dealing with semistructured data (e.g. XML and/or HTML documents).

Web site denotation

In our framework, a Web page is a ground term. Consequently, we represent a Web site as a finite collection of ground terms of a suitable term algebra $\tau(\mathcal{T}ext \cup \mathcal{T}ag)$.

Example:

```
<members>
  <member>
    <name> Mario </name>
    <surname> Rossi </surname>
    <status> Professor </status>
  </member>
  <member>
    <name> Franca </name>
    <surname> Bianchi </surname>
    <status> Technician </status>
  </member>
</members>
```

```
members(
  member(
    name(Mario),
    surname(Rossi),
    status(Professor)
  ),
  member(
    name(Franca),
    surname(Bianchi),
    status(Technician)
  )
)
```

Specification Language

- Web Specifications are sets of rules which allow to specify conditions in order to
  - detect forbidden or incorrect information
  - detect missing or incomplete Web pages

- a Web specification is made up of
  - a set of corretNess rules $\mathtt{I}_{\mathtt{N}}$
  - a set of coMpletenes rules $\mathtt{I}_{\mathtt{M}}$
  - a set of rewrite rules (i.e. a Term Rewriting System) $\mathtt{R}$

Correctness Rules

- They allow to detect an incorrect/forbidden pattern inside a Web page.
- A *correctness rule* has the following form

$$\texttt{l} \rightharpoonup \texttt{error} \mid X_1 \ \texttt{in} \ rexp_1 \ldots X_n \ \texttt{in} \ rexp_n$$

with $X_i \in Var(\texttt{l})$ and $rexp_i$ a regular expression over $\tau(\mathcal{T}ext \cup \mathcal{T}ag)$, $i = 1, \ldots, n$.

- Interpretation: if `l` is recognized in some Web Page, then `l` is rewritten to `error` whenever the values bound to each X_i belong to the corresponding regular language $rexp_i$.

Correctness Rules - Examples

- A rule to forbid sexual contents from being published in the home pages.

$$\texttt{hpage}(\texttt{X}) \rightharpoonup \texttt{error} \mid \texttt{X in } [\!: \texttt{TextTag} :\!] * \texttt{ sex } [\!: \texttt{TextTag} :\!]*$$

- A rule to detect blinking text (aim: improving accessibility for people with disabilities)

$$\texttt{blink}(\texttt{X}) \rightharpoonup \texttt{error}$$

Completeness Rules

- They allow to discover missing information in a Web site.
- A completeness rule is a rule of the following form

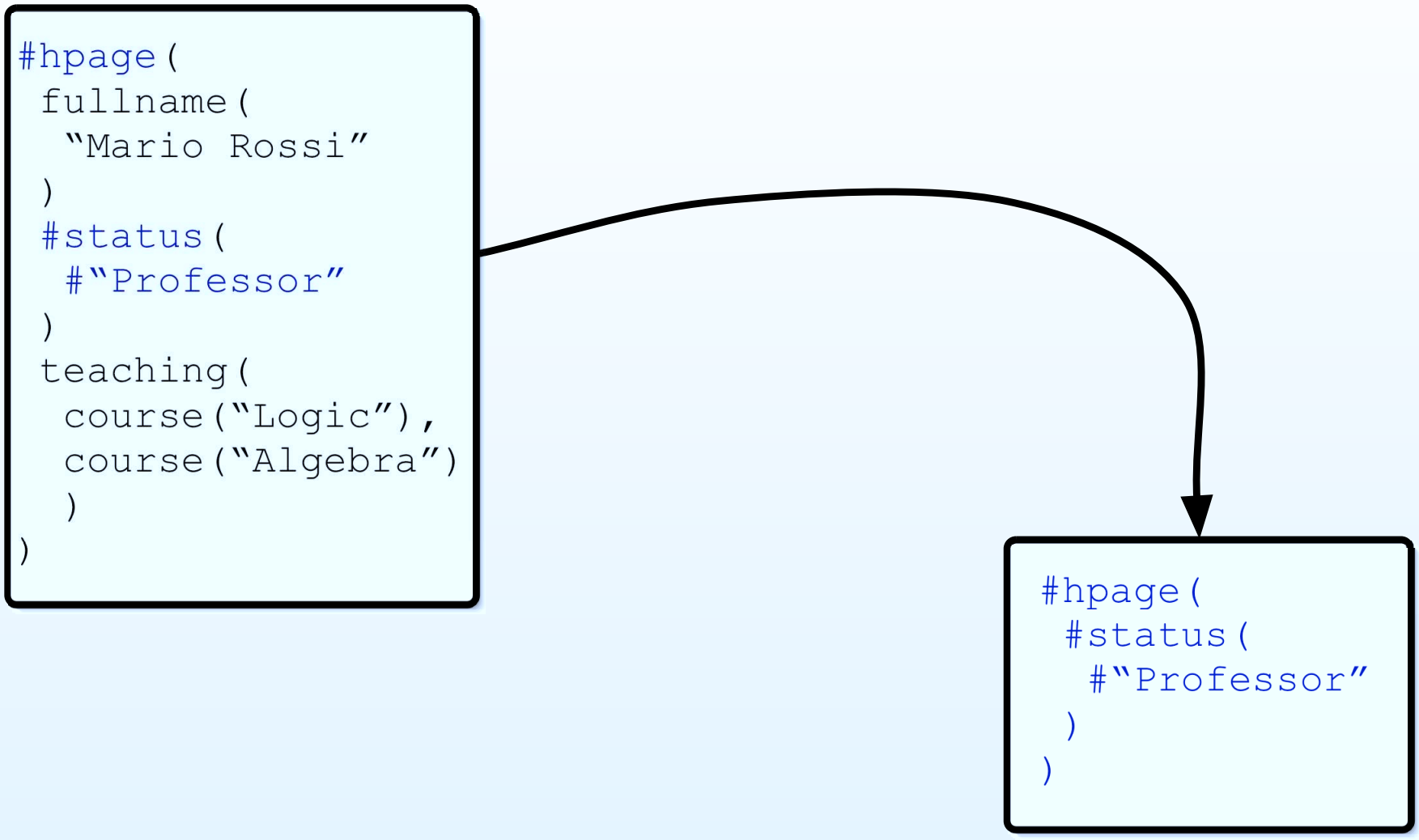
$$\mathtt{l} \rightharpoonup \mu(\mathtt{r})\ \langle \mathtt{q} \rangle$$

  where $\mathtt{l}$,$\mathtt{r}$ are terms, μ is a marking function for marking some symbols of $\mathtt{r}$ by means of a $\#$.
- $\mathtt{q} \in \{\mathtt{A}, \mathtt{E}\}$, q can be either an universal or an existential quantifier.
- Marks are used to select the Web pages on which we want to check a given condition.

Marked part of a term

```
#hpage(
 fullname(
  “Mario Rossi”
 )
 #status(
  #“Professor”
 )
 teaching(
  course(“Logic”),
  course(“Algebra”)
  )
)
```

Marked part of a term



Completeness Rules - Interpretation

Given a Web site $\mathtt{W}$

- An existential completeness rule $\mathtt{l} \rightharpoonup \mu(\mathtt{r})\langle \mathtt{E} \rangle$ is interpreted as follows:

  if $\mathtt{l}$ is recognized in some Web page of $\mathtt{W}$, then (the irreducible form of) $\mathtt{r}$ must be recognized in some Web page of $\mathtt{W}$ which contain the marked part of $\mathtt{r}$.

- An universal completeness rule $\mathtt{l} \rightharpoonup \mu(\mathtt{r})\langle \mathtt{A} \rangle$ is interpreted as follows:

  if $\mathtt{l}$ is recognized in some Web page of $\mathtt{W}$, then (the irreducible form of) $\mathtt{r}$ must be recognized in every Web page of $\mathtt{W}$ which contain the marked part of $\mathtt{r}$.

Completeness Rules - Examples

- A rule to verify whether the home pages of professors contain some teaching information inside.

$$\texttt{hpage}(\texttt{status}(\texttt{professor})) \rightharpoonup \#\texttt{hpage}(\#\texttt{status}(\#\texttt{professor}), \texttt{teaching}))\ \langle \texttt{A} \rangle$$

- A rule to verify whether there exists a home page for each member of a group.

$$\texttt{member}(\texttt{name}(\texttt{X}), \texttt{surname}(\texttt{Y})) \rightharpoonup \#\texttt{hpage}(\texttt{fullname}(\texttt{append}(\texttt{X}, \texttt{Y})), \texttt{status})\ \langle \texttt{E} \rangle$$

- A rule to verify whether each author of a publication is also a member of the group.

$$\texttt{pubs}(\texttt{pub}(\texttt{name}(\texttt{X}), \texttt{surname}(\texttt{Y}))) \rightharpoonup \#\texttt{member}(\texttt{name}(\texttt{X}), \texttt{surname}(\texttt{Y}))\ \langle \texttt{E} \rangle$$

A Web specification - Example

Consider a Web site containing some information about a research group (e.g., member group affiliation, personal data, publications,. . .).

- Correctness Rules $\mathtt{I}_N$

$$\mathtt{hpage(X)} \rightharpoonup \mathtt{error} \mid \mathtt{X\ in\ [:\ TextTag\ :]} * \mathtt{\ sex\ [:\ TextTag\ :]}*$$
$$\mathtt{blink(X)} \rightharpoonup \mathtt{error}$$

- Completeness Rules $\mathtt{I}_M$

$$\mathtt{hpage(status(professor))} \rightharpoonup \#\mathtt{hpage(\#status(\#professor), teaching))}\ \langle \mathtt{A} \rangle$$
$$\mathtt{member(name(X), surname(Y))} \rightharpoonup \#\mathtt{hpage(fullname(append(X, Y)), status)}\ \langle \mathtt{E} \rangle$$
$$\mathtt{pubs(pub(name(X), surname(Y)))} \rightharpoonup \#\mathtt{member(name(X), surname(Y))}\ \langle \mathtt{E} \rangle$$

- Rewrite Rules $\mathtt{R}$ = Definition of function append

Tree Simulation

Simulation allows us to recognize the structure and the labels of a Web page (template) into another. It provides a powerful pattern-matching mechanism:

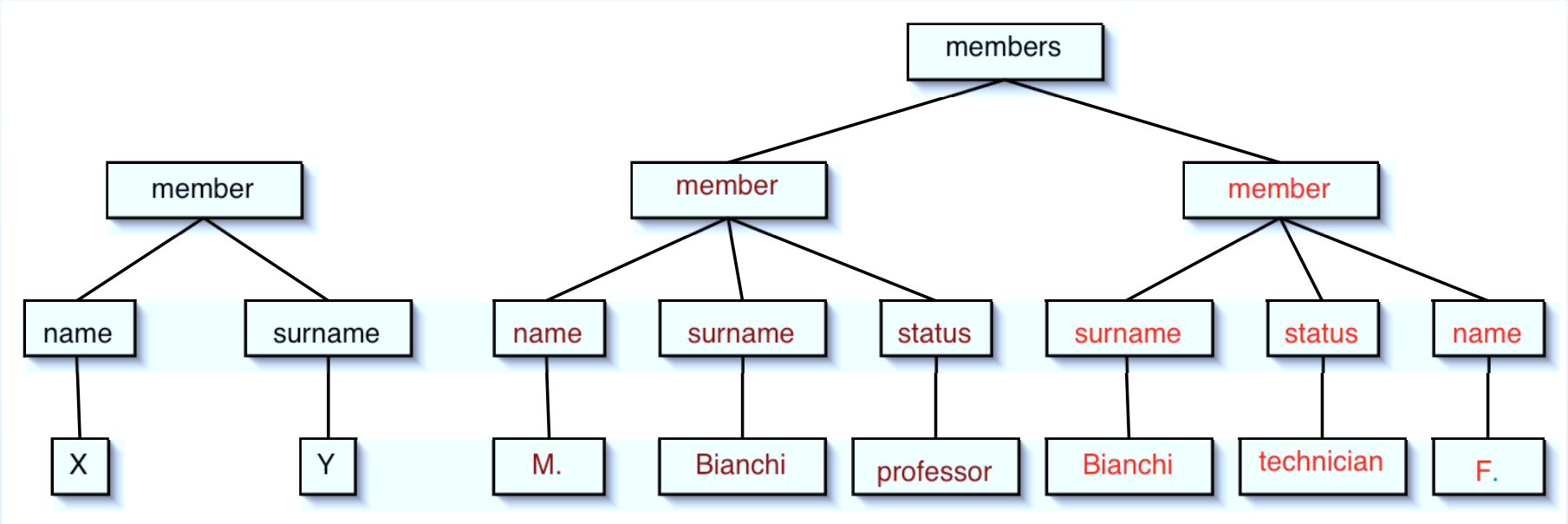
- suitable for dealing with HTML/XML data (partial matching, unordered trees)
- fast (efficient algorithms do exist)

Tree Simulation

Simulation allows us to recognize the structure and the labels of a Web page (template) into another. It provides a powerful pattern-matching mechanism:

- suitable for dealing with HTML/XML data (partial matching, unordered trees)
- fast (efficient algorithms do exist)

Example (minimal, injective simulation)

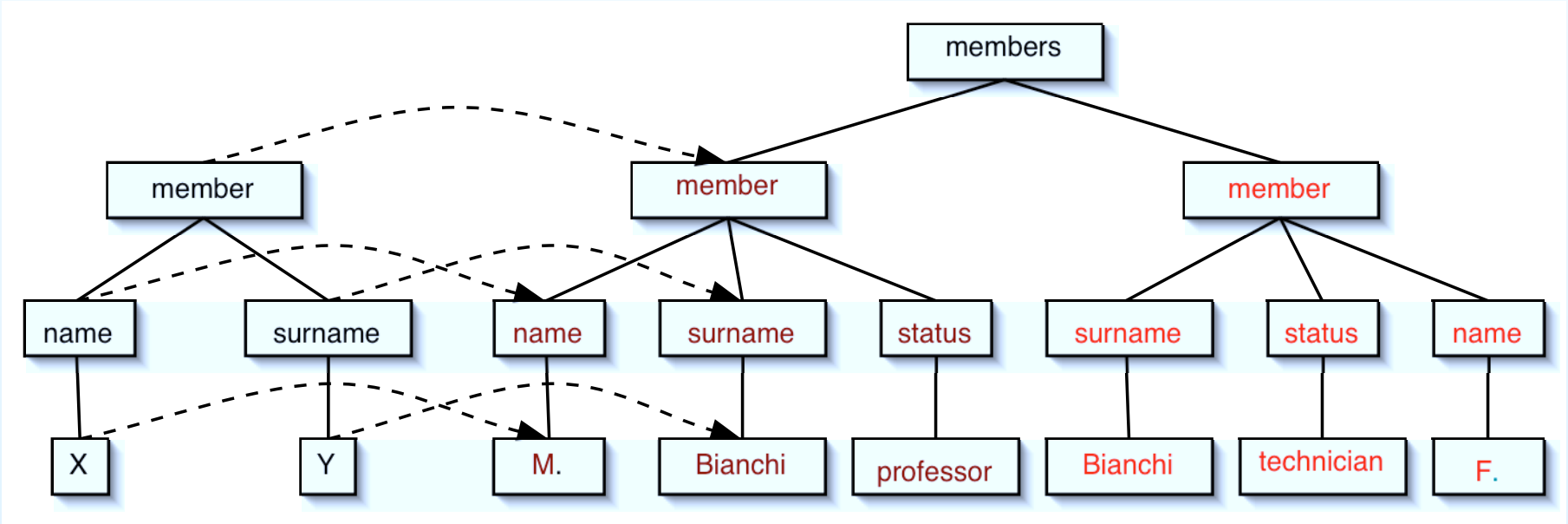


Tree Simulation

Simulation allows us to recognize the structure and the labels of a Web page (template) into another. It provides a powerful pattern-matching mechanism:

- suitable for dealing with HTML/XML data (partial matching, unordered trees)
- fast (efficient algorithms do exist)

Example (minimal, injective simulation)

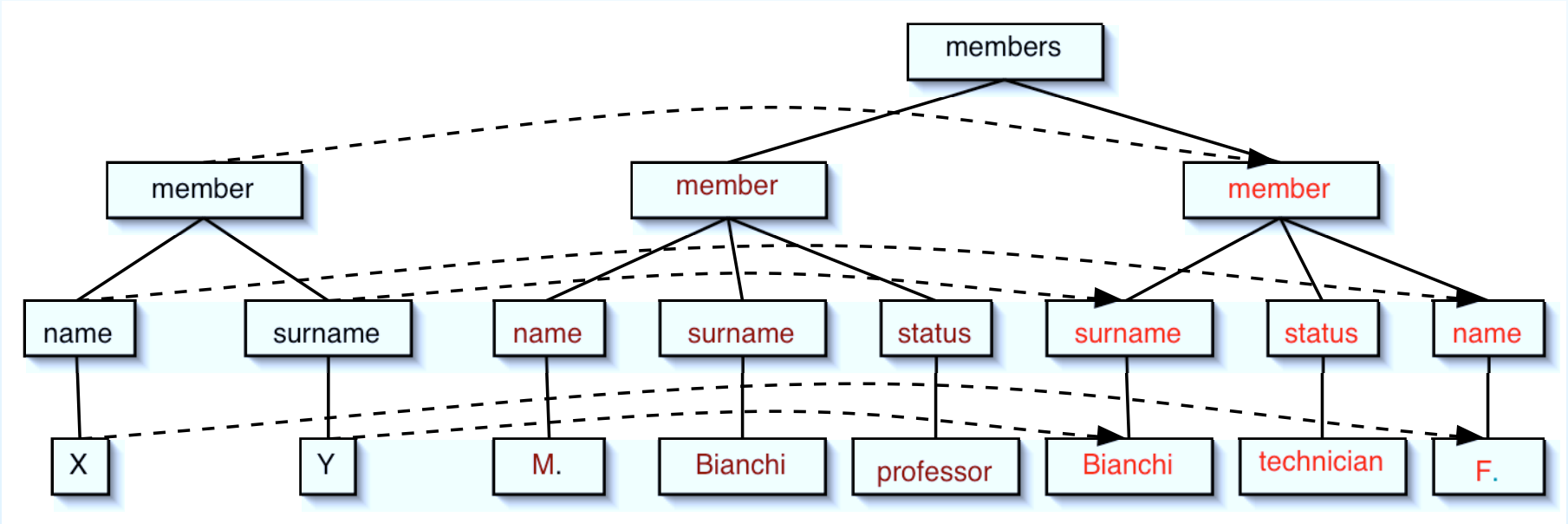


Tree Simulation

Simulation allows us to recognize the structure and the labels of a Web page (template) into another. It provides a powerful pattern-matching mechanism:

- suitable for dealing with HTML/XML data (partial matching, unordered trees)
- fast (efficient algorithms do exist)

Example (minimal, injective simulation)



Partial Rewriting $\rightharpoonup$

Partial Rewriting: a rewriting relation in which

- the traditional *pattern matching* mechanism is replaced by tree simulation
- the context of selected reducible expressions is disregarded
- we deal with marking information

$$\mu(\mathtt{t}) \rightharpoonup_r^{\sigma} \mu'(\mathtt{t}')$$

Partial rewriting steps

```
members(member(name(Mario),surname(Rossi),
               status(Professor)),
        member(name(Franca),surname(Bianchi),
               status(technician))
        )
```

is partially rewritten to

$\underline{\texttt{hpage}}\texttt{(fullname(append(Mario,Rossi), status)}$
$\rightarrow_{\mathrm{R}} \underline{\texttt{hpage}}\texttt{(fullname(MarioRossi), status)}$

and

$\underline{\texttt{hpage}}\texttt{(fullname(append(Franca,Bianchi), status)}$
$\rightarrow_{\mathrm{R}} \underline{\texttt{hpage}}\texttt{(fullname(FrancaBianchi), status)}$

by rule $\texttt{member(name(X), surname(Y))} \rightharpoonup \underline{\texttt{hpage}}\texttt{(fullname(append(X, Y)), status)}$

Verification of a Web site

We provide a verification methodology which is able to

1. detect erroneous patterns of a Web sites `W` w.r.t a Web specification `I`;
2. detect missing/incomplete Web pages of a Web sites `W` w.r.t a Web specification `I`;
3. collect information which is needed to repair `W`.

Erroneous pattern detection

Methodology

1. Check whether some Web page of `W` can be partially rewritten to `error` by using some correctness rule r_{corr}.
2. Check whether the instantiated condition of r_{corr} holds.

If Points (1) and (2) are satisfied, an incorrectness error is raised!

Missing/incomplete Web pages

The main idea is the following.

1. Generate a set of requirements $Req_{\mathtt{I},\mathtt{W}}$ by applying $\mathtt{I}$ to $\mathtt{W}$.
2. Check whether requirements in $Req_{\mathtt{I},\mathtt{W}}$ are fulfilled in $\mathtt{W}$.

Terms, which are derived via partial rewriting from the original Web site by means of the completeness rules, can be considered as requirements to be fulfilled by $\mathtt{W}$.
We can compute the set of requirements via a fixpoint computation.

$$Req_{\mathtt{I},\mathtt{W}} = lfp(R_{\mathtt{I}})$$

where $\mathtt{R_I}$ is a continuos operator similar to the immediate consequence operator which is used in Logic Programming.

Missing/incomplete Web pages (cont'd)

A requirement is not fulfilled if it cannot be simulated in a suitable subset of the Web site which has been generated by exploiting marking information.

Existential and universal checks!

Repairing Web sites

During the verification process, we can collect useful information in order to semi-automatically fix the correctness/completeness errors which are discovered. That is,

- detected incorrect patterns allow us to locate the information we need to change in order to get a correct Web site.
- unsatisfied requirements provide us the missing information which is necessary to complete the Web site.

Conclusions

- We presented a rewriting-based language for the specification and the verification of integrity conditions on Web sites.
  - partial rewriting
  - erroneous patterns and missing/incomplete Web pages
- We have also implemented the prototype VERDI (VErification and Rewriting for Debugging Internet sites), which is publicly available at

  `http://www.dimi.uniud.it/~demis/#software`
- We tested the system on real Web sites, e.g.

  `http://www.dimi.uniud.it/clg`

END