Resource Trees Models and Semi-structured Data

Nicolas Biri and Didier Galmiche
LORIA - Université Henri Poincaré
Nancy, France

3rd Nancy-Saarbruecken Workshop on Logic, Proofs and Programs, Nancy, October 2005.
Resources and distribution (1)

- **Resource**: basic and central notion in computer science.

- Location, access to, ownership, consumption of resources are central concerns in the design of programs and systems.

- Focus on **location** and **distribution** of resources
  - new resource models,
  - logical axiomatizations of these models
    * separation and spatial logics
    * reasoning on semi-structured data
Resources and distribution (2)

Separation logics (O’Hearn et al. 01, Reynolds 02)

- Built from BI or IL logics (with $\ast$ as separation connector).
- Resources from a preordered monoid, $\mathcal{R} = (R, \cdot, e, \sqsubseteq)$.

- Sharing and separation of resources:
  - $m \models \psi \ast \phi$ iff $\exists n, n'$ such that $n \cdot n' \sqsubseteq m$, $n \models \phi$ and $n' \models \psi$ (separation)
  - $m \models \psi \land \phi$ iff $m \models \phi$ and $m \models \psi$ (sharing).

Example: $m \models p \ast (q \land q')$ iff the resource $m$ can be decomposed in two resources, one satisfying $p$, the other satisfying $q$ and $q'$. 
Resources and distribution (3)

Spatial logics (Cardelli et al. 01-02)

- Ambients logic, logic for trees (with $|$ as spatial connector).
- **Locations** instead of resources:
  - $P \models A|B$ iff $\exists Q, Q'$ such that $Q|Q' \equiv P$, $Q \models A$ and $Q' \models B$.
  - $P \models A \wedge B$ iff $P \models A$ and $P \models B$.
  - $P \models l[A]$ iff $P \equiv l[P']$ and $P' \models A$.
  - Propositions restricted to units 0 (empty), $T$ (true) and $F$ (false).

Example: $P \models l_1[0]|l_2[0]$ iff $P$ can be divided into two locations, one empty location $l_1$ and another empty location $l_2$. 
Resources and distribution (4)

A new separation logic (Biri-Galmiche 03)

- $[l] \phi$ means that $\phi$ holds at location $l$.
- It expresses both spatial and resource separation.
- A new model based on resource trees.

Example: $[l_1] p \ast [l_2] (q \land q')$ is satisfied if the space can be divided in two locations: $l_1$ in which a resource verifies $p$ and $l_2$ in which a resource verifies both $q$ and $q'$. 
Semi-structured data and logics (1)

- Logical models to reason about semi-structured data (for instance XML document transformations)
  - Models to represent data.
  - Commands to manipulate models.
  - A logic for specification of documents.
  - An assertion language to verify transformations.
Semi-structured data and logics (2)

- Tree logic (Calcagno, Cardelli et al. 03):
  - structure: labelled trees, logic: static ambient logic;
  - issues: limited representation (elements, id and idref attributes), problems with tree update.

- Context logic (Calcagno, Gardner et al. 05):
  - structure: trees and contexts, logic: tree logic and context operators;
  - issues: limited representation (elements, id and idref attributes), two levels of reasoning (tree, context).
Our approach

• Resource tree model (Biri-Galmiche 03):
  – Application to XML data:
    - Resource trees to represent XML data;
    - Separation logic for XML document specification.

• New results:
  – Complete representation of XML document, tree update, one level of reasoning.
  – Commands and an assertion language to reason about transformations.
Resource Trees (1)

\[ T_{M,\mathcal{L}} ::= M \times [\mathcal{L} \xrightarrow{\text{fin}} T_{M,\mathcal{L}}] \]

- Labelled tree with resources inside nodes:
  - Labels in \( \mathcal{L} \) and unique: a label only used at one given place.
  - Resources in \( M \), a partial monoid of resources.
  - Other representation as a usual tree grammar:

\[
P ::= m \mid P|P \mid [l](P)
\]
Resource Trees (2)

\[
[l_1](m_1 \mid [l_2]e) \mid [l_3][l_4]m_2
\]
Resource Trees (3)

- Tree composition and tree update:

\[
[l_1]m[l_2]e \quad \mid \quad [l_1]m'[l_3]e \quad \equiv \quad [l_1](m\cdot m')[l_2]e[l_3]e
\]
An example: semi-structured data (1)

A XML data and its corresponding resource tree.

```xml
<message>
  <from>Alice</from>
  <to>Bob</to>
</message>

<message id='43'>
  <from>Bob</from>
  <to>Alice</to>
</message>
```

\[ l_1 \rightarrow l_2 \rightarrow l_3 \quad l_4 \rightarrow l_5 \quad l_6 \rightarrow l_7 \rightarrow l_8 \rightarrow l_9 \rightarrow l_{10} \]
An example: semi-structured data (2)

To alter a given structure (by adding data inside a node):

(P) \hspace{1cm} (P') \hspace{1cm} (P'')
An example: semi-structured data (3)

Using partiality to avoid invalid XML trees (two ids in a node).

\[(P)\] \[(P')\]

Here \textit{id42} $\bullet$ \textit{id43} is undefined.
BI-Loc and resource tree model (1)

- BI-Loc is a logic in which cohabit:
  - multiplicative $(*, \ast)$ to handle separation;
  - additive ($\land, \lor, \rightarrow$) to handle sharing;
  - location modality $([l])$ to handle spatial distribution;
  - quantifications ($\forall, \exists$) on location, path and resources.

- A model based on resource trees
  (partial monoid semantics of BI (Galmiche et al. 02) but with trees instead of resources).
BI-Loc and resource tree model (2)

- Resource tree model:
  - $t \models [l] \phi$ iff there exists $t'$ s.t. $[l]t' \preceq t$ and $t' \models \phi$.
  - $t \models \phi \ast \psi$ iff there exist $t', t''$ s.t. $t'|t''$ is defined, $t'|t'' \preceq t$, $t' \models \phi$ and $t'' \models \psi$.
  - $t \models \phi \land \psi$ iff $t \models \phi$ and $t \models \psi$.
  - $t \models \phi \ast \ast \psi$ iff for all $t'$ such that $t' \models \phi$ and $t|t'$ is defined, $t|t' \models \psi$.
  - $t \models \phi \rightarrow \psi$ iff $t \models \phi$ implies that $t \models \psi$.
  - .......

and for all $p$, for all $t, t'$ if $t \models p$ and $t' \preceq t$ then $t' \models p$. 
BI-Loc and resource tree model (3)

• Some results (Biri-Galmiche 03, 04)

  – Decidability (by model-checking) for
    - BI logic (boundable resource models);
    - BI-Loc logic (resources trees)

  – Decidability (by theorem-proving)
    - a tableau method based on labels, constraints, dependency graphs;
    - build counter-models in case of non-provability.
Example: resource tree properties

- A subtree at path \( L \) satisfies \( P \): \( \text{contains}(L, P) = \top * [L]P \).

- Path \( L \) exists in the tree: \( \text{exists}(L) = \text{contains}(L, \top) \).

- Path \( L \) does not exist in the tree: \( \text{no}(L) = \text{exists}(L) \rightarrow \bot \).

- Add information to a fresh path: \( \text{no}(L) \land ([L]P \ast Q) \).

- Tree update: \( [L]P \ast ([L]P' \ast Q) \).
XML and Resource Trees (1)

- Principles:

  - to adapt the resource monoid for XML document representation;

  - to use the resource logic for specification;

  - to define a command language for XML document manipulation;

  - to verify document transformations with an assertion language.
XML and Resource Trees (2)

- XML data as resource trees:
  
  - partial composition ensure that documents are well-formed.

```xml
<state id='s2'>
  <scode>NE</scode>
  <sname>Nevada</sname>
  <capital idref='c3'/>
</state>
<city id='c3'>
  <ccode>CCN</ccode>
  <cname>Carson City</cname>
  <state_of idref='s2'/>
</city>
```
BI-Loc to specify XML documents

- *state* elements have 3 children, *scode*, *sname*, and *capital*:

\[
\forall_{path} x. (contains(x, elem(state)) \rightarrow \\
[x] \exists_{loc} y_1, y_2, y_3. [y_1](elem(scode) \ast \top) * \\
[y_2](elem(sname) \ast \top) * [y_3](elem(capital) \ast \top) * \\
\forall_{loc} z. no(x : z))
\]

- Each *state_of* element make reference to an existing *id* of a *state*:

\[
\forall_{path} x. \forall_{res} v. (contains(x, elem(state_of) \ast (attr(idref) \land \\
value(v))) \rightarrow \exists_{path} y. ((([y]id \land (elem(state) \ast value(v))) \ast \top)))
\]
A language for XML data manipulation

Commands to:

- add informations (elements, attributes, data, subtrees);
- delete informations (elements, attributes, data, subtrees);
- retrieve information (elements, attributes, data, subtrees);
- find a specific location in the tree.
Commands of the language (sample)

- Create a new element \( E_2 \) below path \( E_1 \):
  \[
  C := x := \text{new}_{\text{elem}}E_2@E_1.
  \]

- Delete the attribute \( E_2 \) from location \( E_1 \):
  \[
  C := x := \text{delete}E_2@E_1.
  \]

- Put in \( x \) the element present at location \( E \) (element lookup):
  \[
  C := x := \text{elem}@E.
  \]

- Put in \( x \) the location name present below \( E_3 \) and which contains an attribute \( E_1 \) which has a value \( E_2 \) (find a location):
  \[
  C := x := \text{getloc}_{\text{attr}}(E_1, E_2)@E_3.
  \]
Example of transformation (1)

Initial data and program:

\[
x := \text{getLoc}_\text{elem}(\text{state})@\text{nil}
\]
\[
x := \text{getLoc}_\text{elem}(\text{capital})@x
\]
\[
y := \text{getAttr}(\text{idref})@x
\]
\[
z := \text{getLoc}_{\text{attr}}(\text{idref}, y)@\text{nil}
\]
\[
\text{addAttr}(\text{capital}, \text{yes})@z
\]
\[
z := \text{getLoc}_\text{elem}(\text{cname})@z
\]
\[
z = \text{getLoc}_{\text{cont}}@z
\]
\[
y = \text{getContent}@z
\]
\[
y = \text{new}_{\text{cont}}(y)@x
\]
Example of transformation (2)

Result:
Assertions and transformations (1)

- Hoare’s triples \(\{P\}C\{Q\}\):
  If a document \(d\) checks property \(P\) and if command \(C\) is applied on \(d\), the resulting document \(d'\) satisfies property \(Q\).

- Examples:
  - Element lookup:
    \[
    \{\exists_{path\ y}. (exists(y : E) \land \exists_{res\ c}. (\exists_{[\ y][\ E]} elem(c) \land P_{c/x}) ))\}
    x := elem@E \quad \{P\}
    \]
  - Add element:
    \[
    \{\exists_{path\ y}. (exists(y : E_2) \land \forall_{path\ y'} . \forall_{loc\ z} . (\no(y' : z) \rightarrow
    \exists_{[\ y][\ E_2][\ z]} elem(E_1) \rightarrow P_{z/x}) ))\}
    x := new elem \ E_1 \ @ \ E_2 \quad \{P\}
    \]
Assertions and transformations (2)

- Examples:
  - Delete Attribute:
    \[
    \{\exists_{path}.(P \land [y][E_1]attr(E_2))\} \ deleteE_2@E_1 \ \{P\}
    \]
  - Find path:
    \[
    \{\exists_{path}.\exists_{loc}z.([y][E_3][z](attr(E_1) \land value(E_2)) \land \forall_{loc}.((\top \land
    [y][E_3][t](attr(E_1) \land value(E_2)) \rightarrow \bot)) \land P\{z/x\})
    \]
    \[
    x := \text{getLocattr} (E_1, E_2)@E_3 \ \{P\}
    \]

- Weakest preconditions:
  - Result: the axioms are sound and complete according to the semantics.
An example (1)

Back to the transformation example:

- Initial property: $P_{init}$.

- Final property:

$$P_{final} = \top \ast [l_7](\text{attr}(\text{capital}) \land \text{value}(\text{yes})) \ast$$

$$\exists_{\text{loc}x} \exists_{\text{res}c}.([l_7][l_10][l_11]\text{dat}(c) \ast [l_1][l_6][x]\text{dat}(c)).$$

- Backward reasoning:

\[
\begin{align*}
\{P_0\} & \quad x := \text{getLoc}_\text{elem}(\text{state})@\text{nil} \quad \{P_1\} \\
\{P_1\} & \quad x := \text{getLoc}_\text{elem}(\text{capital})@x \quad \{P_2\} \\
\{P_2\} & \quad y := \text{getAttr}(\text{idref})@x \quad \{P_3\}
\end{align*}
\]
An example (2)

- Backward reasoning:

\[
\begin{align*}
\{P_3\} & \quad z := \text{getLoc}_{\text{attr}}(id, y)@\text{nil} & \{P_4\} \\
\{P_4\} & \quad \text{addAttr}(\text{capital}, \text{yes})@z & \{P_5\} \\
\{P_5\} & \quad z := \text{getLoc}_{\text{elem}}(\text{cname})@z & \{P_6\} \\
\{P_6\} & \quad z = \text{getLoc}_{\text{cont}}@z & \{P_7\} \\
\{P_7\} & \quad y = \text{getContent}@z & \{P_8\} \\
\{P_8\} & \quad y = \text{new}_{\text{cont}}(y)@x & \{P_{\text{final}}\}
\end{align*}
\]

- Is \( P_0 \) a logical consequence of \( P_{\text{init}} \)?

\[\Rightarrow\] Proof-search tools must be used to \textit{glue} formulae
Resource trees and other models (1)

**Pointer models** (O’Hearn et al. 01)

- Resource trees and BI’s pointer models
  - Correspondence between **heaps** and **resource trees**
  - Assertion language for pointer manipulation

- Resource trees and permission models (Bornat et al. 05).
  - Correspondence between **permissions** and **resource trees**
  - Solution to an open problem: how to represent trees in this model?
Resource trees and other models (2)

Hierarchical storage
(hierarchical representation of memory as a store)

Store: partial map from path into value: \( s \in Path \rightarrow Val \)

<table>
<thead>
<tr>
<th>Hierarchical Storage</th>
<th>Resource Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path</td>
<td>Path</td>
</tr>
<tr>
<td>Value</td>
<td>Resource</td>
</tr>
</tbody>
</table>

Store: a resource tree with a totally undefined monoid of resources.

- Contrary to Hierarchical storage our resource trees are finite.

+ We can extend our results to the propositional logic for hierarchical storage (Ahmed et al 03).
Further works

• What about proof theory for specific models?

• Are there decidable fragments including $\rightarrow^*$ and quantifications?

• What about relationships between based-on resource trees representation of XML data and W3C languages as DTD or XSLT?