Expressive Logical Combinators for Free

Pierre Genevès (CNRS)

Joint work with Alan Schmitt (Inria)

IJCAI, July 30th 2015, Buenos Aires

Query Analysis

Constructs and Languages

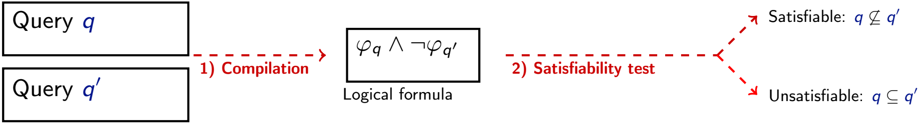
• query	q	XPath, JAQL, SPARQL
• database	d	XML, JSON, RDF
• schema	S	DTD, OWL

Problems and Applications

• Query containment	$q(d) \subseteq q'(d)$	for all d	the view update problem
• Query equivalence	$q(d) = q'(d)$	for all d	optimization of programs
• Query satisfiability	$q(d) \neq \emptyset$	for some d	dead code analysis
• Static type-checking	$q(d) \vdash S'$	for all $d \vdash S$	proving program correctness

The Logical Approach for Query Analysis

Reducing e.g. Query Containment to Logical Satisfiability:



Complexity

- Complexity of satisfiability test depends on |formula|
- Blow-ups in the logical translation increase combined complexity
- Succinctness of the logic is crucial!

  $\rightarrow$ Improving it: addressing more problems while reducing combined complexity

Example

- Set of strings over $\Sigma = \{a, b, c\}$ containing at least 2 occurrences of "a" and at least 2 occurrences of "b":

Example

- Set of strings over $\Sigma = \{a, b, c\}$ containing at least 2 occurrences of "a" and at least 2 occurrences of "b":

$$
\begin{aligned}
L_{2a2b} = \quad & (a|b|c)^\star a(a|b|c)^\star a(a|b|c)^\star b(a|b|c)^\star b(a|b|c)^\star \; | \\
& (a|b|c)^\star a(a|b|c)^\star b(a|b|c)^\star a(a|b|c)^\star b(a|b|c)^\star \; | \\
& (a|b|c)^\star a(a|b|c)^\star b(a|b|c)^\star b(a|b|c)^\star a(a|b|c)^\star \; | \\
& (a|b|c)^\star b(a|b|c)^\star b(a|b|c)^\star a(a|b|c)^\star a(a|b|c)^\star \; | \\
& (a|b|c)^\star b(a|b|c)^\star a(a|b|c)^\star b(a|b|c)^\star a(a|b|c)^\star \; | \\
& (a|b|c)^\star b(a|b|c)^\star a(a|b|c)^\star a(a|b|c)^\star b(a|b|c)^\star
\end{aligned}
$$

Example

- Set of strings over $\Sigma = \{a, b, c\}$ containing at least 2 occurrences of "a" and at least 2 occurrences of "b":

$$
\begin{aligned}
L_{2a2b} = \quad & (a|b|c)^\star a(a|b|c)^\star a(a|b|c)^\star b(a|b|c)^\star b(a|b|c)^\star \mid \\
& (a|b|c)^\star a(a|b|c)^\star b(a|b|c)^\star a(a|b|c)^\star b(a|b|c)^\star \mid \\
& (a|b|c)^\star a(a|b|c)^\star b(a|b|c)^\star b(a|b|c)^\star a(a|b|c)^\star \mid \\
& (a|b|c)^\star b(a|b|c)^\star b(a|b|c)^\star a(a|b|c)^\star a(a|b|c)^\star \mid \\
& (a|b|c)^\star b(a|b|c)^\star a(a|b|c)^\star b(a|b|c)^\star a(a|b|c)^\star \mid \\
& (a|b|c)^\star b(a|b|c)^\star a(a|b|c)^\star a(a|b|c)^\star b(a|b|c)^\star
\end{aligned}
$$

- If we add $\cap$ to the regular expression operators:

$$L_{2a2b} = ((a|b|c)^\star a(a|b|c)^\star a(a|b|c)^\star) \cap ((a|b|c)^\star b(a|b|c)^\star b(a|b|c)^\star)$$

- $\cap$ offers a dramatic reduction in expression size!
- Crucial when complexity of the decision procedure depends on |formula|
- More generally, how can we increase succinctness?

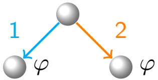
Logical Combinators

A Combinator is a Predicate that:

- takes logical formulas as input (and outputs a logical formula)
- might arbitrarily duplicate input formulas in its definition

Examples (using branching logic from [Genevès-PLDI'07])

- $\text{split}(\varphi) = \langle 1 \rangle\, \varphi \wedge \langle 2 \rangle\, \varphi$



- Order relations between tree nodes (e.g. depth-first tree traversal)

  $\text{next}(\varphi) = \underbrace{\langle 1 \rangle\, \mu z.\varphi \vee \langle 1 \rangle\, z \vee \langle 2 \rangle\, z}_{\text{descendant}(\varphi)} \quad \vee \ldots \quad \underbrace{\ldots}_{\text{following}(\varphi)}$

- Regular queries with counting e.g. "at least 3 occurrences of φ":

  $\text{threeOrMore}(\varphi) = \text{next}(\varphi \wedge \text{next}(\varphi \wedge \text{next}(\varphi)))$

Results on Combinators

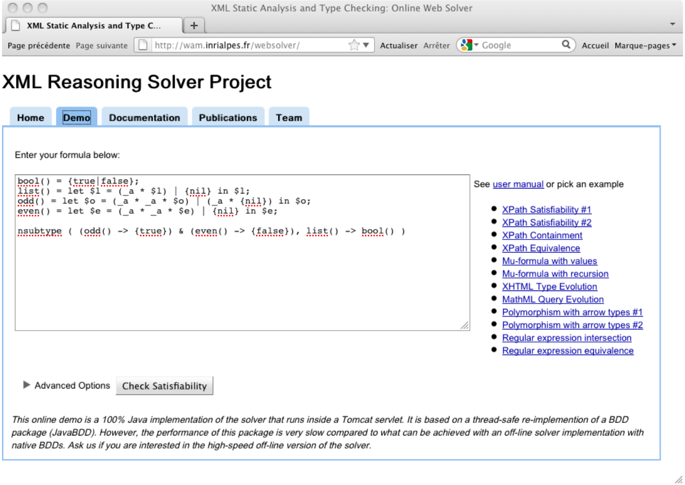
- Combinators form an expressive and succinct logical language

  (regular tree and path languages, counting...)
- Proof: combinators do not increase the complexity of decision procedures à la [Genevès-TOCL15] which stays in $2^{\mathcal{O}(|\varphi|)}$ (MSO-complete logic)

Concrete Problem	$\lvert\varphi\rvert$	Time
Simple RE intersection & equivalence	30	15 ms
Query containment $q \subseteq q'$ (XPath)	50	50 ms
Query satisfiability with constraints (e.g. SMIL 1.0)	90	350 ms
Subtyping with rich types	60	70 ms
Schema evolution (moderate: e.g. XHTML-Basic)	170	2.5 s
Schema evolution (large: e.g. MathML)	290	8 s
Analysis of CSS style sheets (IJCAI'15)	60	40 ms
Precise static type-checking for XQuery (ICFP'15)	70	35 ms

Table: Experimental Results.

Try Combinators*: http://tyrex.inria.fr/websolver



*** and build your own predicate language for addressing your problem!**