# Mechanized metatheory: ready for prime time?

James Cheney
University of Edinburgh
jcheney@inf.ed.ac.uk

The POPLMark challenge has inspired, and workshops such as WMM and LFMTP have documented, great progress in mechanizing proofs of properties of programming languages. There is growing evidence that several current techniques are adequate for ensuring the correctness of syntactic proofs of results in papers in top conferences and journals, or verifying compilers for existing languages. These are excellent goals for mechanized metatheory research. Nevertheless, we should aim higher.

Is mechanized metatheory ready for prime time? Can it be used to inform the design of new languages, rather than for postmortem analysis of existing ones? I think the answer is still no. I suggest that the community of researchers interested in mechanized metatheory consider targeting some new languages that are interesting to people outside this community, and trying to use mechanized metatheory to inform and improve these languages. Doing so will give us a clearer idea of what (engineering and research) challenges remain to realize the full potential of mechanization. Doing so may also help differentiate techniques in terms of scalability, expressiveness and ease of use, in ways that (relatively) small benchmarks such as the POPLMark problems or formalization tours de force have not. Here is one suggestion.

The World Wide Web consortium (W3C) is designing new languages and language extensions at a breakneck pace: ten W3C Recommendations have been published so far in 2010 as of this writing (though only one or two of these are programming languages, as such). With a few exceptions, these languages are not formally specified, let alone mechanically checked. If mechanized metatheory is to have impact beyond the programming language research community, then we need to figure out how to mechanize the metatheory of real, evolving language designs, fast and effectively enough to inform overworked standards committees.

There are many obstacles that prevent this from being easy or rewarding in the short term. Standards organizations are subject to many pressures, and careful formal thinking about language design and semantics is not necessarily recognized (and sometimes actively discouraged). Standards committees include both academic and industry representatives with a wide variety of backgrounds, meaning that involvement from programming languages researchers will likely require patience and compromise. However, some recent W3C standards efforts have been receptive to the idea of formal semantics; in the case of XQuery [2], a partial formal semantics was adopted as part of the standard [4]. We should build upon these encouraging signs.

XQuery is a language for querying XML databases. It is a W3C Recommendation (that is, a Web standard). Although intended primarily for database applications, XQuery has many features of general-purpose languages, and it includes functions, recursion, and arithmetic, making it Turing-complete. XQuery is becoming an important language, and one for which subtle reasoning is needed to ensure that optimization and compilation techniques are correct. Moreover, because of its comparatively simple and clean design, it appears within reach to completely formalize the semantics of XQuery expressions, and verify standard optimization and compilation techniques. I think this can be done within, say, two years using at least some current tools.

Many papers on implementing, analyzing or optimizing XQuery have been published (including [6, 5, 1], among others). Yet these results are generally based on a simplified fragment of XQuery whose semantics is relatively clean, which for convenience I will call *mini-XQuery*. Results about mini-XQuery may not transfer directly to the full language. Many equational optimization rules or transformations hold on-the-nose for mini-XQuery but can be invalid in full XQuery. For ex-

ample, the identity:

$$\texttt{let } \$x = e_1 \texttt{ return } e_2 \equiv e_2[e_1/\$x]$$

does not hold if, for example, $e_1$ is an expression such as $\langle foo\rangle bar\langle/foo\rangle$ that constructs a new XML element node (with a fresh identity) and $e_2$ performs an operation that is sensitive to the node identity of the value of $\$x$, such as $\$x$ `is` $\$x$ or `count`($\$x$ `union` $\$x$).

XQuery is a more appealing target for formalization than other related languages. It is small compared to the relational database query language SQL, and its specification is freely and publicly available. In comparison with other recent W3C standards, XQuery already has a relatively clean (although partial) formal semantics that can be used as a starting point for mechanization. There are also open-source implementations of XQuery that pass most of the W3C's informal benchmarks, and these implementations can be used to determine how implementers have addressed ambiguities in the standard.

XQuery is also appealing because it is still a work in progress: version 1.0 leaves out many features that are considered important, such as updates. Well-targeted work on formalizing the metatheory of XQuery or other W3C standards could have an impact on future versions as they are developed, particularly as advanced features are added.

I have begun working on a formalization of mini-XQuery in Nominal Isabelle [8], and plan to develop this into a complete formalization that handles all interesting aspects of the language. In this talk I will present the background described above, discuss what has been formalized so far and enumerate the challenges remaining. Many of the challenges do not immediately relate to name-binding:

1. XML trees are modeled using explicit identifiers that can be accessed through the language by features such as `is` (which tests whether two nodes have the same identity). Other features are sensitive to node identity, including `union`, which removes duplicate nodes. Node identifiers can be generated at run time when new data is constructed.
2. XQuery builds on the XPath language which permits navigating a tree data structure in any direction. For example, given a node we can always find its parent, ancestors, and siblings. The most obvious way to model this semantics is to use node identities, but formalizing this seems likely to become complicated quickly.
3. XQuery also includes features such as unordered processing modes that make the semantics nondeterministic and may further complicate equational reasoning about optimization rules.
4. There are also several optional or high-level features, such as modules, namespaces, and schema importing that might be (at least bureaucratically) challenging to formalize. These are a lower priority, since optimization is usually performed only at the level of expressions.
5. Several extensions to XQuery are being developed, including updates [3], and (for XQuery 1.1) higher-order functions, exceptions and more advanced querying features [7]. These extensions are not defined by a formal semantics.

## References


[1] Michael Benedikt and James Cheney. Schema-based independence analysis for XML updates. *PVLDB*, 2(1):61–72, 2009.

[2] S. Boag, D. Chamberlin, M. Fernández, D. Florescu, J. Robie, and J. Siméon. XQuery 1.0: An XML query language. W3C Recommendation, January 2007.

[3] D. Chamberlin, M. Dyck, D. Florescu, J. Melton, J. Robie, and J. Siméon. XQuery update facility 1.0. W3C Candidate Recommendation, August 2008.

[4] D. Draper, P. Fankhauser, M. Fernández, A. Malhotra, K. Rose, M. Rys, J. Siméon, and P. Wadler. XQuery 1.0 and XPath 2.0 formal semantics. W3C Recommendation, January 2007.

[5] Giorgio Ghelli, Nicola Onose, Kristoffer Høgsbro Rose, and Jérôme Siméon. XML query optimization in the presence of side effects. In *SIGMOD*, pages 339–352, 2008.

[6] Christopher Ré, Jérôme Siméon, and Mary F. Fernández. A complete and efficient algebraic compiler for XQuery. In *ICDE*, page 14, 2006.

[7] Jonathan Robie, Don Chamberlin, Michael Dyck, and John Snelson. XQuery 1.1: An XML query language. W3C Working Draft, December 2009.

[8] Christian Urban. Nominal techniques in Isabelle/HOL. *J. Autom. Reasoning*, 40(4):327–356, 2008.