

Fancy Types for XML:

Benjamin C. Pierce
University of Pennsylvania

<Links> meeting, Edinburgh, April 2005

Fancy Types for XML: Friend or Foe?

Benjamin C. Pierce
University of Pennsylvania

<Links> meeting, Edinburgh, April 2005

XML and the Web

- Any candidate web-programming language *must* deal seriously with XML
- At least, there must be good support for XML concrete syntax... but this is a pretty trivial matter.
- The real question: How XML gets into the type system?


Xtatic

- Xtatic is a lightweight extension of C# with
 - regular types (a la XDuce)
 - regular pattern matching

See my web page
for lots of papers

The Xtatic Experience

What was learned:

- XML processing with "native" static types is indeed very pleasant
- rich type structure of many XML documents  more traction for type system

The Xtatic Experience

What was learned:

- XPath-style ("vertical") and regular-pattern-style ("horizontal") pattern matching are both very useful, in different situations
- it appears [cf. Benzaken et al 2005, Gapeyev&Pierce 2004, etc.] that they can be placed on a common foundation
- one nice use-case for the horizontal style is statically typed string regexps
- another is that horizontal patterns also generalize ML-style algebraic pattern matching

See our PLANX '05 paper for more details

The Less-Than-Xtatic Experience

Some tricky issues...

- Standards compliance (W3C Schema, XPath, etc., etc.)
 - "Best effort" approach (but, e.g., no notion of "Schema-validated run-time values")
 - deciding subtyping efficiently
 - algorithms are known that seem to be "efficient enough in practice," but these are not trivial to implement [Hosoya/Vouillon/Pierce ICFP 2000]
 - compiling regular patterns efficiently [but cf. Michael Levin's forthcoming dissertation]
 - Precise type inference for pattern variables [but not clear it is absolutely necessary]
 - Finding the "right" type system for attributes is still an open problem [but see proposal by Hosoya&Murata]

Some Critical Design Points

- **Structural** (types are descriptions of structure of values) vs. **nominal** (each value is tagged at run time with a single atomic "type name" that it belongs to) treatment of types
- Interaction between **subtyping** of XML types and the full language's subtype relation. (Your language *has* subtyping, right??)
- **Single-type tree grammars** (W3C Schema, XQuery) vs. full **regular tree grammars** (XDuce, Xtatic, CDuce, RelaxNG, etc.)
- **Power** of the language of XML types (unions and recursion for sure; but what about intersections? differences? interleaving?)