XML Schema

**Motivation**
- Enforcing type invariants brings safety
- There are many useful type systems
- Type systems are typically closed
- How to type new abstractions?
- How to “debug” new type systems?

**Applications**
- Physical units (kg, m/s, N etc.)
- Nullability, sign, oddity etc. of values
- Security vulnerability of format strings
- Usage of user pointers in kernel space
- Deadlocks and dataraces
- Regular expression types
- Ownership types

**Typing XML in C++**
- Existing language tools available
- No composibility problem
- No need for a dedicated preprocessor
- Existing language tools available

**XTL Abstractions**
- is subtype<T,S>
- subtype_cast<T,S>

```
// Example 1: Array subtyping (assumes that user defined int <: double)
extern double sum(double a[4]);
int a[7] = {0,1,2,3,4,5,6};
double r = sum(subtype_cast<double>[4]>(a)); // r == 21.0
```

```
// Example 2: Function subtyping (class D is derived from B)
typedef B (D_to_B)(B);
typedef D (B_to_D)(B);
extern B f(D);
extern D g(B);
D_to_B* pf = &g; // function f is of type D_to_B
D_to_B* pf = &g; // function g is of type B_to_D
D_to_B* p = subtype_cast<D_to_B*>(&g); // OK with the help of XTL
```

**Type Qualifiers**
- DECLARE_NEGATIVE_QUALIFIER(pos);
- DECLARE_POSITIVE_QUALIFIER(tainted);

```
// Define your type qualifiers and their relation to the type they qualify
DECLARE_NEGATIVE_QUALIFIER(pos); // When Q<T> <: T, Q is a negative qualifier
DECLARE_POSITIVE_QUALIFIER(tainted); // When T <: Q<T>, Q is a positive qualifier
```

```
// Define how operations transfer qualifiers: adds/drops/changes/carry etc.
template struct type { typedef  qualifiers; template struct type { typedef qualifiers; template struct type { typedef qualifiers; template struct type { typedef qualifiers;
```

```
You guys are both my witnesses... No inquisition that a type system cannot be a library!
```

**Solution: Type System as a Library**
- No need for a dedicated preprocessor
- No composibility problem
- Existing language tools available

**Cartoon credit:** SangleyFormalMethods

http://shemesh.larc.nasa.gov/fm/

**XTL: an eXtensible Typing Library**

Yuriy Solodkyy, Jaakko Järvi, Esam Mlaih

Programming Techniques, Languages, and Tools Research Group

When I use a “type cast” it means just what I chose it to mean - neither more nor less.
- Humpty Dumpty

```
<schema .../>
<element name="name" type="string"/>
<element name="tel" type="decimal"/>
<element name="email" type="string"/>
<element name="contact">
  <complexType>
    <sequence>
      <element ref="name"/>
      <element ref="tel" maxOccurs="unbounded"/>
      <element ref="email" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</element>
</schema>
```