The Essence of XML

Jérôme Siméon, Bell Labs, Lucent

Philip Wadler, Avaya Labs

The Evolution of Language

$2x$ (Descartes)

$\lambda x.\, 2x$ (Church)

```
(LAMBDA (X) (* 2 X))
```
(McCarthy)

```
<?xml version="1.0"?>
<LAMBDA-TERM>
  <VAR-LIST>
    <VAR>X</VAR>
  </VAR-LIST>
  <EXPR>
    <APPLICATION>
      <EXPR><CONST>*</CONST></EXPR>
      <ARGUMENT-LIST>
        <EXPR><CONST>2</CONST></EXPR>
        <EXPR><VAR>X</VAR></EXPR>
      </ARGUMENT-LIST>
    </APPLICATION>
  </EXPR>
</LAMBDA-TERM>
```

(W3C)

XML everywhere!

VoiceXML FORUM

September 3, 2001

The Business of VoiceXML

speechTECHNOLOGY MAGAZINE

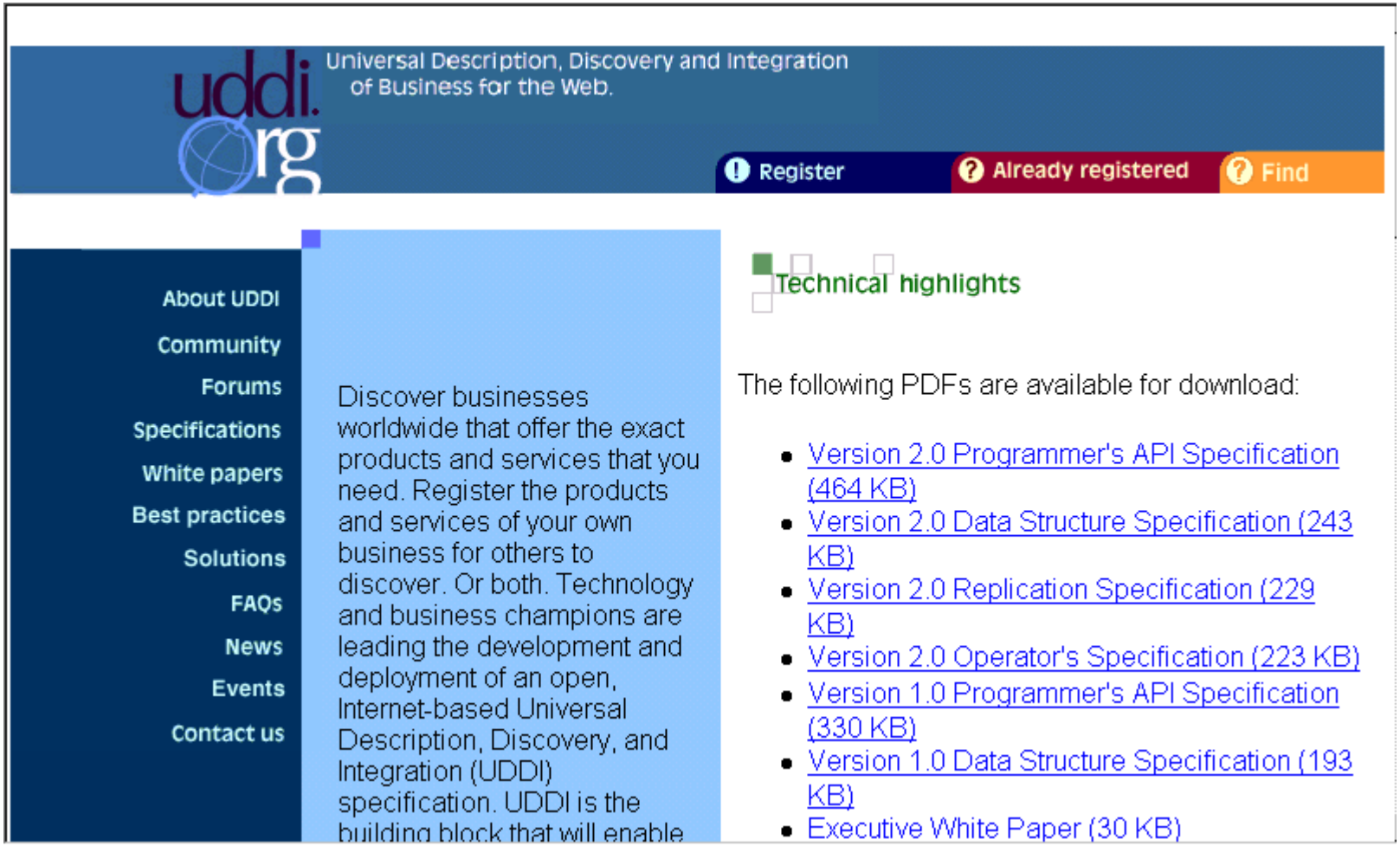
The July/August issue of *Speech Technology* magazine offers an insightful look at VoiceXML as told through interviews conducted with various VoiceXML Forum members. The article, entitled "The Business Side of VoiceXML: Faster Time to Market is Only Part of the Story", covers topics including:

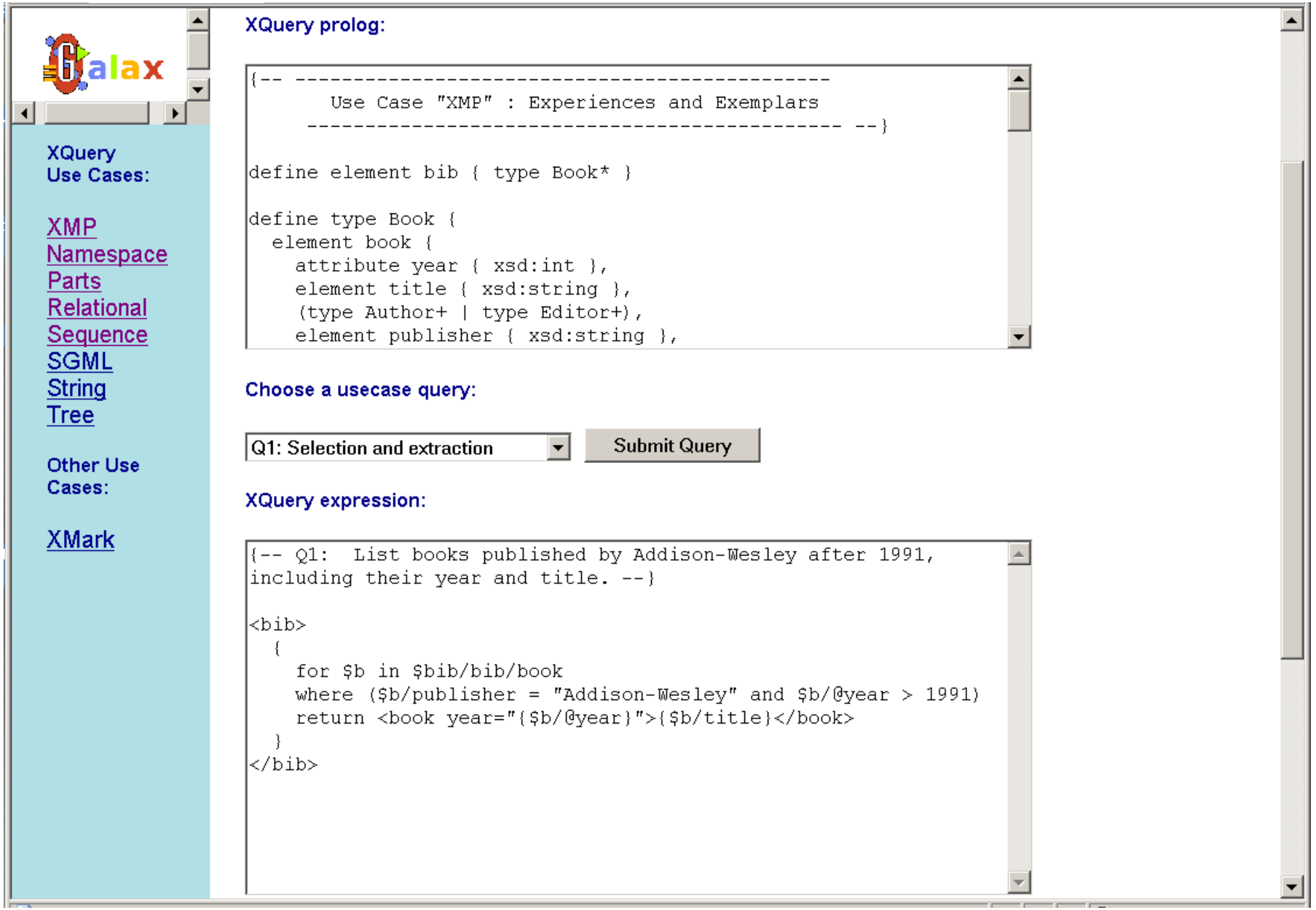
N E W S

1 August 2001

VoiceXML Forum membership grows to 537 member companies--including 4 Sponsor Members, 59 Promoter Members


ebXML
ENABLING A GLOBAL ELECTRONIC MARKET
| ABOUT | INDUSTRY SUPPORT | NEWS | SPECS | REPORTS | REFERENCE | WHITE PAPERS | TECHNICAL WORK |
ebXML
About
Industry Support
News and Articles
FAQ
Contacts
Logo
Presentations
Resources
Initiative Archive
SPECS/DOCUMENTS
Specifications
Technical Reports
Reference Materials
White Papers
TECHNICAL WORK
Overview
Business Process
Core Components
Collaboration Protocol
Messaging
Registry / Repository
Implementation
ebXML enables enterprises of any size, in any location to meet and conduct business through the exchange of XML-based messages.
ebXML NEWS
[01 August 2001] OpenTravel Alliance Endorses ebXML
[30 July 2001] UN/CEFACT Forms e-Business Transition Ad hoc Working Group
[21 June 2001] OASIS Forms ebXML Technical Committees
[22 May 2001] UN/CEFACT and OASIS
INDUSTRY SUPPORT
Open Applications Group to incorporate ebXML into 182 mature Business Object Documents
Korea Institute for Electronic Commerce (KIEC) Opens Prototype ebXML Registry & Repository
Covisint Supports ebXML Technology Findings
More ebXML Adoption News
JOINTLY SPONSORED BY
UN/CEFACT
OASIS
ebXML-DEV MAIL LIST
Join this open forum to exchange ideas on implementing ebXML.
Subscribe >>
View Archives >>
Book excerpt: ebXML: The


uddi.org
Universal Description, Discovery and Integration of Business for the Web.
Register
Already registered
Find
About UDDI
Community
Forums
Specifications
White papers
Best practices
Solutions
FAQs
News
Events
Contact us
Discover businesses worldwide that offer the exact products and services that you need. Register the products and services of your own business for others to discover. Or both. Technology and business champions are leading the development and deployment of an open, Internet-based Universal Description, Discovery, and Integration (UDDI) specification. UDDI is the building block that will enable
Technical highlights
The following PDFs are available for download:
Version 2.0 Programmer's API Specification (464 KB)
Version 2.0 Data Structure Specification (243 KB)
Version 2.0 Replication Specification (229 KB)
Version 2.0 Operator's Specification (223 KB)
Version 1.0 Programmer's API Specification (330 KB)
Version 1.0 Data Structure Specification (193 KB)
Executive White Paper (30 KB)


Galax
XQuery Use Cases:
XMP
Namespace
Parts
Relational
Sequence
SGML
String
Tree
Other Use Cases:
XMark
XQuery prolog:
{-- --
Use Case "XMP" : Experiences and Exemplars
-- --}
define element bib { type Book* }
define type Book {
element book {
attribute year { xsd:int },
element title { xsd:string },
(type Author+ | type Editor+),
element publisher { xsd:string },
Choose a usecase query:
Q1: Selection and extraction
Submit Query
XQuery expression:
{-- Q1: List books published by Addison-Wesley after 1991,
including their year and title. --}
<bib>
{
for $b in $bib/bib/book
where ($b/publisher = "Addison-Wesley" and $b/@year > 1991)
return <book year="{$b/@year}">{$b/title}</book>
}
</bib>

XML Query Language Demo

Microsoft .net

New XQuery Prototype released!

A new version of Microsoft's XQuery Prototype was released on December 9, 2002. This version is based on the August 15th draft of the W3C XQuery specification.

Documentation

- What's New
- Readme
- Known Issues

Specifications

- XQuery Syntax Draft
- XQuery Functions & Operators
- XQuery Use Cases

W3C Use Cases

- XMP Cases
- TREE Cases
- SEQ Cases
- SGML Cases
- NS Cases

Download

XQuery Demo (Dec 20, 2001 Draft)

Feedback

Click here to send us feedback.

Introduction

Welcome to Microsoft's XQuery Demo. This demo was designed with the August 15th, 2002 version of the XQuery working draft.

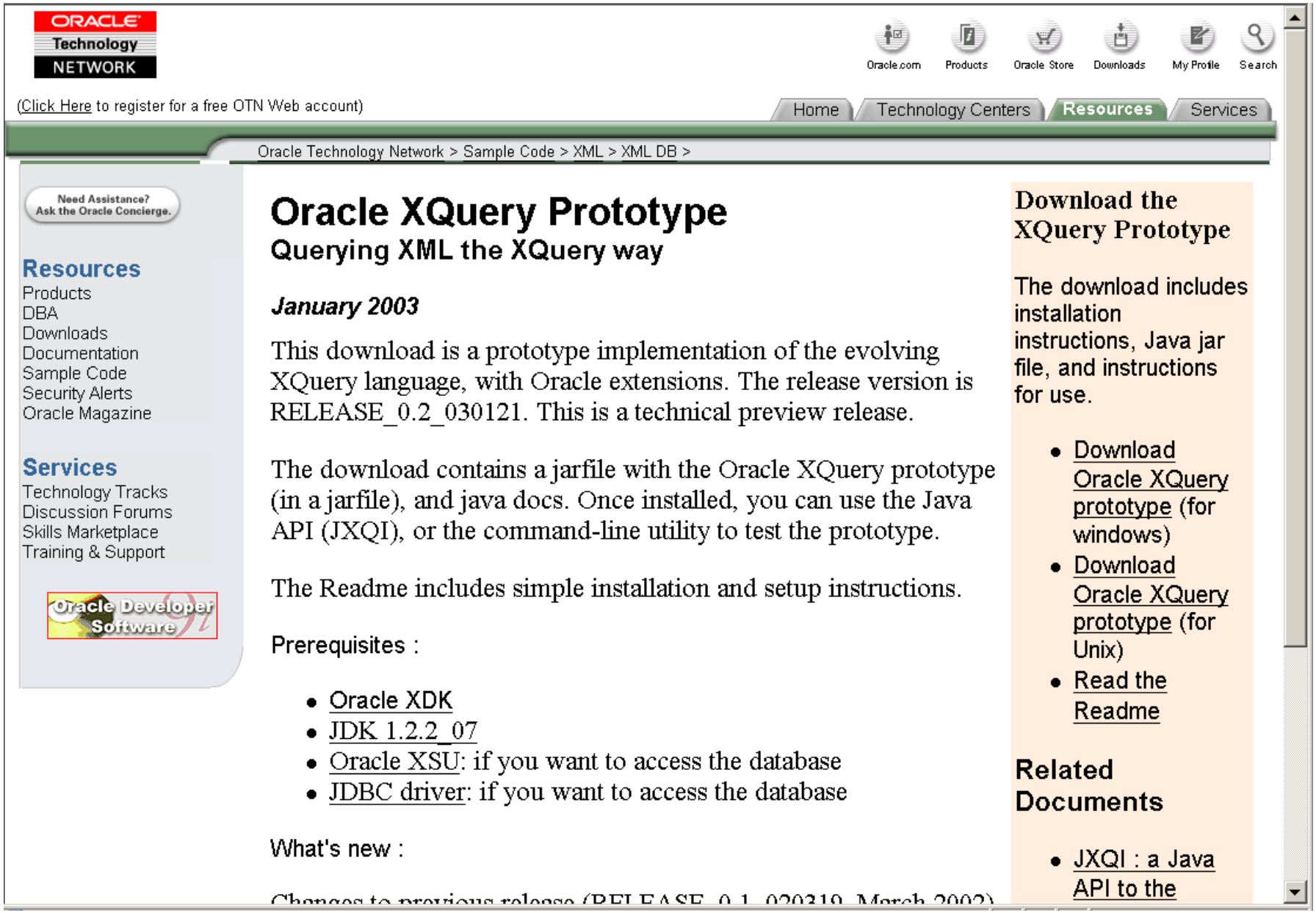
Instructions:

1. Ether select one of the example "W3C Use Cases" in the pane to the left or type your own query in the text box below.
2. Click the "Execute Query" Button to generate results. (Note: Results will be displayed in a new window)

Please refer to the Readme for more information.

Query Expression

```
<bib>
 {
  for $b in document("http://www.bn.com/bib.xml")/bib/book
  where $b/publisher = "Addison-Wesley" and $b/@year > 1991
  return
    <book year="{ $b/@year }">
     { $b/title }
    </book>
 }
</bib>
```



ORACLE Technology NETWORK

Oracle.com | Products | Oracle Store | Downloads | My Profile | Search

(Click Here to register for a free OTN Web account)

Home | Technology Centers | Resources | Services

Oracle Technology Network > Sample Code > XML > XML DB >

Need Assistance? Ask the Oracle Concierge.

Resources
Products
DBA
Downloads
Documentation
Sample Code
Security Alerts
Oracle Magazine

Services
Technology Tracks
Discussion Forums
Skills Marketplace
Training & Support

Oracle Developer Software 9i

Oracle XQuery Prototype

Querying XML the XQuery way

January 2003

This download is a prototype implementation of the evolving XQuery language, with Oracle extensions. The release version is RELEASE_0.2_030121. This is a technical preview release.

The download contains a jarfile with the Oracle XQuery prototype (in a jarfile), and java docs. Once installed, you can use the Java API (JXQI), or the command-line utility to test the prototype.

The Readme includes simple installation and setup instructions.

Prerequisites :

- Oracle XDK
- JDK 1.2.2_07
- Oracle XSU: if you want to access the database
- JDBC driver: if you want to access the database

What's new :

Changes to previous release (RELEASE_0.1_020319, March 2002)

Download the XQuery Prototype

The download includes installation instructions, Java jar file, and instructions for use.

- Download Oracle XQuery prototype (for windows)
- Download Oracle XQuery prototype (for Unix)
- Read the Readme

Related Documents

- JXQI : a Java API to the

The Essence of XML

XML vs. S-expressions

<foo>1 2 3</foo>

(foo "1 2 3")
(foo 1 2 3)

<bar>1 two 3</bar>

(bar 1 "two" 3)
(bar 1 "two" "3")

XML Schema and Validation

<foo>1 2 3</foo>

⇓

element foo **of type** integer-list { 1, 2, 3 }

⇓

<foo>1 2 3</foo>

```
<xs:simpleType name="integer-list">
  <xs:list itemType="xs:integer" />
</xs:simpleType>
<xs:element name="foo"  type="integer-list" />
```

Mixing it up

<bar>1 two 3</bar>

element bar **of type** mixed-list { 1, ”two”, 3 }

<bar>1 two 3</bar>

```
<xs:simpleType name=”mixed-list”>
  <xs:list>
    <xs:union memberTypes=”xs:integer xs:string” />
  </xs:list>
</xs:simpleType>
<xs:element name=”bar”  type=”mixed-list” />
```

Really mixing it up

element bar **of type** mixed-list { 1, "two", "3" }

⇓

<bar>1 two 3</bar>

⇓

element bar **of type** mixed-list { 1, "two", 3 }

```
<xs:simpleType name="mixed-list">
  <xs:list>
    <xs:union memberTypes="xs:integer xs:string" />
  </xs:list>
</xs:simpleType>
<xs:element name="bar"  type="mixed-list" />
```

The Essence of XML

- The problem it solves is not hard.
- It doesn't solve it very well.

The Essence of XML

- The problem it solves is not hard.
- It doesn't solve it very well.
- (Not entirely fair:
  XML is based on SGML, which was aimed at documents, not data)
- (NB. "Essence" is used in the same sense as
  Reynolds "The Essence of Algol"
  Harper and Mitchell "The Essence of ML"
  Wadler "The Essence of Functional Programming")

Our contribution

- XML and Schema are in widespread use,
  so worth some effort to model.
- We give a foundational theory.
- Validation differs from matching.
- We characterize validation with a theorem.
- Simple version in paper,
  less simple in XQuery formal semantics.

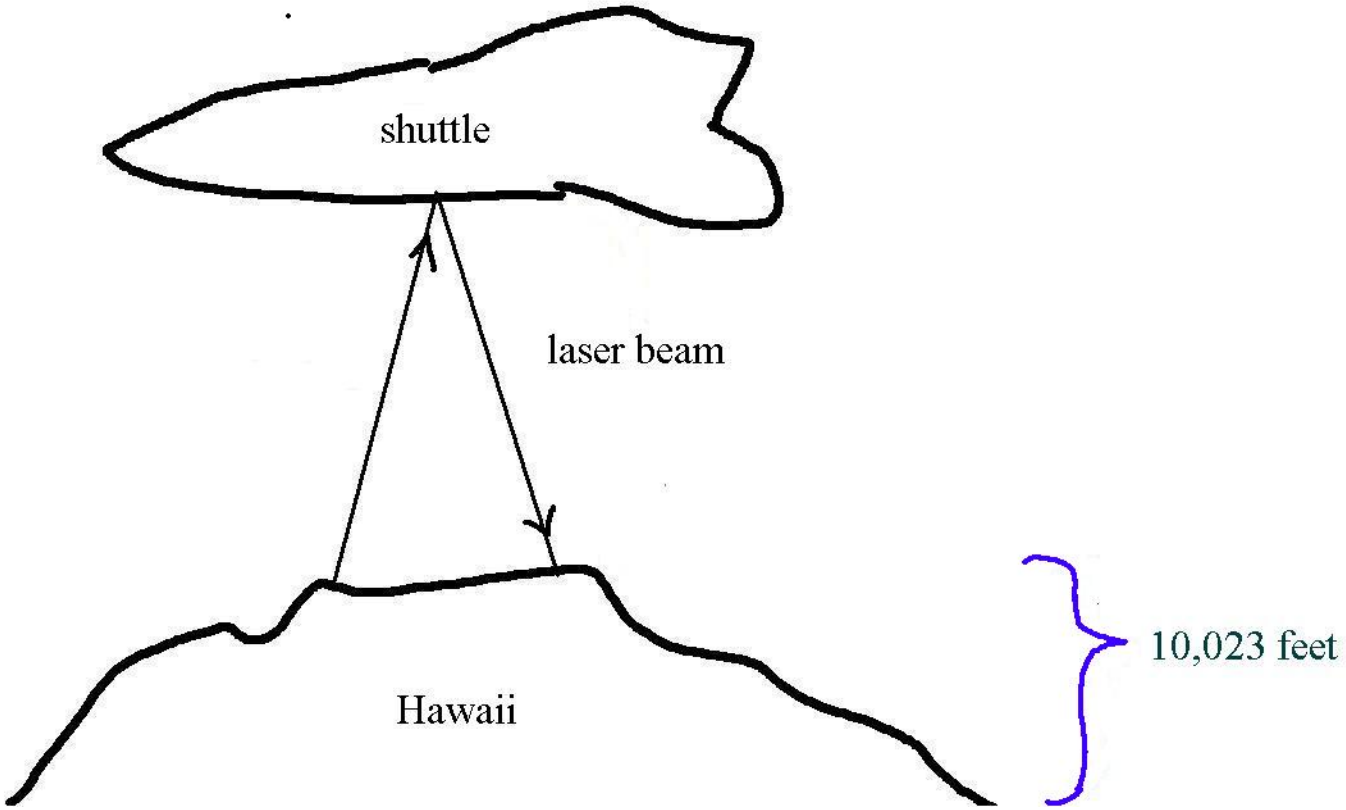
What's in a name?

Structural types vs. Named types

type Feet = Integer
type Miles = Integer

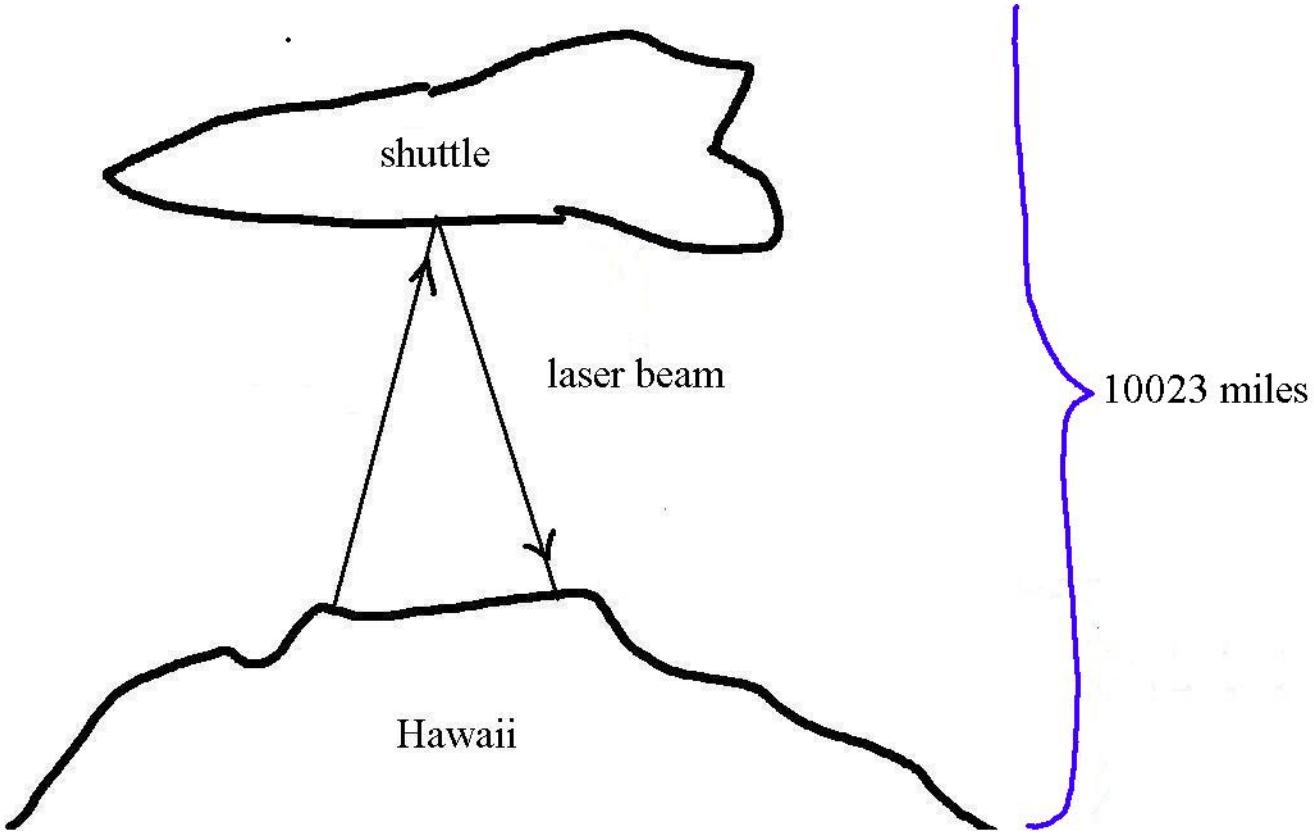
- Structural: two names for the same thing
- Named: two distinct types

Named typing and strategic defense



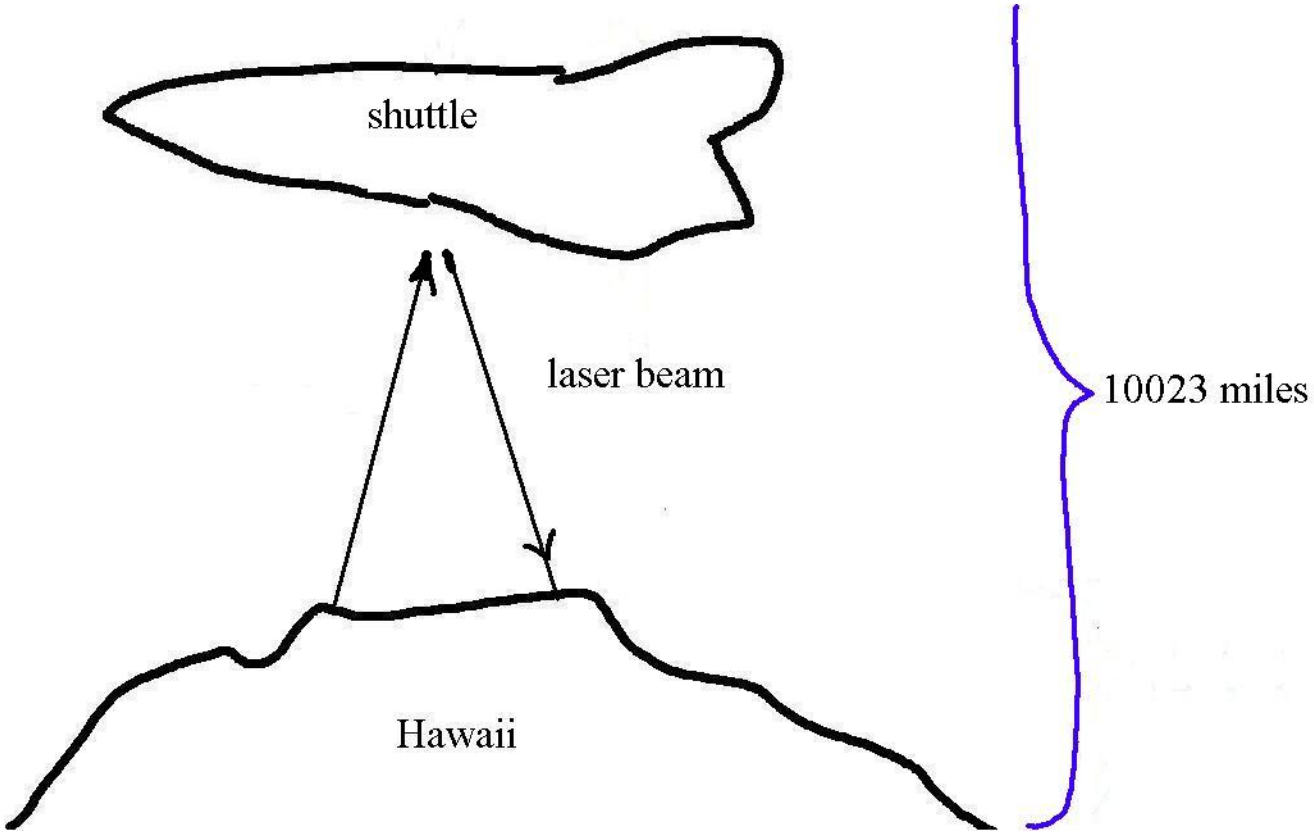

```
enter height?  10023
```

Named typing and strategic defense




```
enter height?  10023
```

Named typing and strategic defense




```
enter height?  10023
```

Schema and XQuery

XML Schema

```
<xs:simpleType name="integer-list">
  <xs:list itemType="xs:integer" />
</xs:simpleType>
<xs:element name="foo"  type="integer-list" />

<xs:simpleType name="mixed-list">
  <xs:list>
    <xs:union memberTypes="xs:integer xs:string" />
  </xs:list>
</xs:simpleType>
<xs:element name="bar"  type="integer-list" />
```

XQuery

```
define type integer-list { xs:integer* }
define element foo of type integer-list

define type mixed-list { (xs:integer|xs:string)* }
define element bar of type mixed-list
```

Schema

```
<xs:simpleType name="feet">
  <xs:restriction base="xs:integer" />
</xs:simpleType>
<xs:simpleType name="miles">
  <xs:restriction base="xs:integer" />
</xs:simpleType>
<xs:element name="configuration">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="shuttle"  type="miles" />
      <xs:element name="laser"  type="feet" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

XQuery

```
define type feet restricts xs:integer
define type miles restricts xs:integer
define element configuration of type configuration.type
define type configuration.type {
   element shuttle of type feet,
   element laser of type miles
}
```

Validation, Matching, and Erasure

Data model

```
<configuration>
  <shuttle>120</shuttle>
  <laser>10023</laser>
</configuration>
```

=

```
element configuration {
  element shuttle { "120" },
  element laser { "10023" }
}
```

Validation

validate as *Type* { *UntypedValue* } ⇒ *Value*

```
validate as element configuration {
  element configuration {
    element shuttle { "120" },
    element laser { "10023" }
  }
} ⇒
  element configuration of type configuration.type {
    element shuttle of type miles { 120 },
    element laser of type feet { 10023 }
  }
```

Matching

Value matches *Type*

element configuration **of type** configuration.type {
  element shuttle **of type** miles { 120 },
  element laser **of type** feet { 10023 }
}
matches
element configuration **of type** configuration.type

Matching depends on type names

Value matches *Type*

element configuration **of type** configuration.type {
  element shuttle **of type** miles { 120 },
  element laser **of type** miles { 10023 }
}

matches

element configuration **of type** configuration.type

(not!)

Unvalidated data does not match

```
element configuration {
  element shuttle { "120" },
  element laser { "10023" }
}
```

matches

```
element configuration of type configuration.type
```

(not!)

Erasure

Value erases to *UntypedValue*

element configuration **of type** configuration.type {
  element shuttle **of type** miles { 120 },
  element laser **of type** feet { 10023 }
}

erases to

element configuration {
  element shuttle { "120" },
  element laser { "10023" }
}

Erasure is a relation

validate as xs:integer ("7") ⇒ 7
validate as xs:integer ("007") ⇒ 7

7 erases to "7"
7 erases to "007"

Inference rules

Matching: Sequence and choice

$$\frac{}{() \text{ matches } ()}$$

$$\frac{\mathit{Value}_1 \text{ matches } \mathit{Type}_1 \quad \mathit{Value}_2 \text{ matches } \mathit{Type}_2}{\mathit{Value}_1 \text{ , } \mathit{Value}_2 \text{ matches } \mathit{Type}_1 \text{ , } \mathit{Type}_2}$$

$$\frac{\mathit{Value} \text{ matches } \mathit{Type}_1}{\mathit{Value} \text{ matches } \mathit{Type}_1 \text{ | } \mathit{Type}_2}$$

$$\frac{\mathit{Value} \text{ matches } \mathit{Type}_2}{\mathit{Value} \text{ matches } \mathit{Type}_1 \text{ | } \mathit{Type}_2}$$

Matching: Occurrence and base types

$$\frac{\mathit{Value}\ \text{matches}\ () \mid \mathit{Type}}{\mathit{Value}\ \text{matches}\ \mathit{Type}?}$$

$$\frac{\mathit{Value}\ \text{matches}\ \mathit{Type}\ ,\ \mathit{Type}*}{\mathit{Value}\ \text{matches}\ \mathit{Type}+}$$

$$\frac{\mathit{Value}\ \text{matches}\ \mathit{Type}+?}{\mathit{Value}\ \text{matches}\ \mathit{Type}*}$$

$$\frac{\mathit{AtomicTypeName}\ \text{derives from}\ \text{xs:string}}{\mathit{String}\ \text{matches}\ \mathit{AtomicTypeName}}$$

$$\frac{\mathit{AtomicTypeName}\ \text{derives from}\ \text{xs:integer}}{\mathit{Integer}\ \text{matches}\ \mathit{AtomicTypeName}}$$

Matching: Element

ElementType
yields *BaseElementName* **of type** *BaseTypeName*

BaseTypeName resolves to *Type*

ElementName substitutes for *BaseElementName*

TypeName derives from *BaseTypeName*

Value matches *Type*

---

element *ElementName* **of type** *TypeName* { *Value* }
matches *ElementType*

Validation: Element

ElementType
yields *BaseElementName* **of type** *BaseTypeName*

BaseTypeName resolves to *Type*

ElementName substitutes for *BaseElementName*

validate as *Type* { *UntypedValue* } ⇒ *Value*

---

validate as *ElementType* {
element *ElementName* { *UntypedValue* }
} ⇒ **element** *ElementName* **of type** *TypeName* { *Value* }

The validation theorem

The validation theorem

Theorem We have that

$$\text{validate as } \mathit{Type} \{ \mathit{UntypedValue} \} \Rightarrow \mathit{Value}$$

if and only if

$$\mathit{Value} \text{ matches } \mathit{Type}$$
$$\mathit{Value} \text{ erases to } \mathit{UntypedValue}.$$

- Obvious in retrospect, not so obvious in prospect.
- Trick is to make validation and erasure into relations.

Ambiguity and Roundtripping

Definition The type *Type* is *unambiguous for validation* if for every *UntypedValue* there is at most one *Value* such that

$$\text{validate as } \mathit{Type} \{ \mathit{UntypedValue} \} \Rightarrow \mathit{Value}.$$

Corollary (Roundtripping) If

$$\begin{gathered} \mathit{Value} \text{ matches } \mathit{Type} \\ \mathit{Value} \text{ erases to } \mathit{UntypedValue} \\ \text{validate as } \mathit{Type} \{ \mathit{UntypedValue} \} \Rightarrow \mathit{Value}' \\ \mathit{Type} \text{ is unambiguous for validation} \end{gathered}$$

then

$$\mathit{Value} = \mathit{Value}'.$$

Example: An unambiguous type

element foo **of type** integer-list { 1, 2, 3 }

erases to

<foo>1 2 3</foo>

validate as **element** foo {

<foo>1 2 3</foo>

} ⇒

element foo **of type** integer-list { 1, 2, 3 }

Example: An ambiguous type

element bar **of type** mixed-list { "1", "two", "3" }

erases to

<bar>1 two 3</bar>

validate as **element** bar {

<bar>1 two 3</bar>

} ⇒

element bar **of type** mixed-list { 1, "two", 3 }

Conclusions

XQuery 1.0: An XML Query Language

W3C Working Draft 16 August 2002

This version:
http://www.w3.org/TR/2002/WD-xquery-20020816/
Latest version:
http://www.w3.org/TR/xquery/
Previous versions:
http://www.w3.org/TR/2002/WD-xquery-20020430/
http://www.w3.org/TR/2001/WD-xquery-20011220/
http://www.w3.org/TR/2001/WD-xquery-20010607/
Editors:
Scott Boag (XSL WG), IBM Research <scott_boag@us.ibm.com>
Don Chamberlin (XML Query WG), IBM Almaden Research Center <chamberlin@almaden.ibm.com>
Mary F. Fernandez (XML Query WG), AT&T Labs <mff@research.att.com>
Daniela Florescu (XML Query WG), XQRL <dana@xqrl.com>
Jonathan Robie (XML Query WG), DataDirect Technologies <jonathan.robie@datadirect-technologies.com>
Jérôme Siméon (XML Query WG), Bell Labs, Lucent Technologies <simeon@research.bell-labs.com>

XQuery 1.0 and XPath 2.0 Formal Semantics

W3C Working Draft 16 August 2002

This version:
http://www.w3.org/TR/2002/WD-query-semantics-20020816/
Latest version:
http://www.w3.org/TR/query-semantics/
Previous versions:
http://www.w3.org/TR/2002/WD-query-semantics-20020326/
http://www.w3.org/TR/2001/WD-query-semantics-20010607/
http://www.w3.org/TR/2001/WD-query-algebra-20010215/
http://www.w3.org/TR/2000/WD-query-algebra-20001204/
Editors:
Denise Draper (XML Query WG), Nimble Technology <ddraper@nimble.com>
Peter Fankhauser (XML Query WG), Infonyte GmbH <fankhaus@infonyte.com>
Mary Fernández (XML Query WG), AT&T Labs - Research <mff@research.att.com>
Ashok Malhotra (XML Query and XSL WGs), Microsoft <ashokma@microsoft.com>
Kristoffer Rose (XSL WG), IBM Research <krisrose@us.ibm.com>
Michael Rys (XML Query WG), Microsoft <mrys@microsoft.com>
Jérôme Siméon (XML Query WG), Lucent Technologies <simeon@research.bell-labs.com>
Philip Wadler (XML Query WG), Avaya <wadler@avaya.com>

3.3.2 Matches

Notation

The judgment

$$\mathit{Value}\ \textbf{matches}\ \mathit{Type}$$

holds when the given value matches the given type.

Semantics

This judgment is specified by the following rules.

The empty sequence matches the empty sequence type.

$$\frac{}{\text{statEnv} \vdash () \textbf{ matches } ()}$$

If two values match two types, then their sequence matches the corresponding sequence type.

$$\frac{\begin{array}{c}\text{statEnv} \vdash \mathit{Value}_1\ \textbf{matches}\ \mathit{Type}_1 \\ \text{statEnv} \vdash \mathit{Value}_2\ \textbf{matches}\ \mathit{Type}_2\end{array}}{\text{statEnv} \vdash \mathit{Value}_1, \mathit{Value}_2\ \textbf{matches}\ \mathit{Type}_1, \mathit{Type}_2}$$

The Essence of XML

- Validation

$$\text{validate as } \mathit{Type} \{ \mathit{UntypedValue} \} \Rightarrow \mathit{Value}$$

- Matching

$$\mathit{Value} \text{ matches } \mathit{Type}$$

- Erasure

$$\mathit{Value} \text{ erases to } \mathit{UntypedValue}$$

- Validation Theorem

  Theorem We have that

$$\text{validate as } \mathit{Type} \{ \mathit{UntypedValue} \} \Rightarrow \mathit{Value}$$

  if and only if

$$\mathit{Value} \text{ matches } \mathit{Type}$$
$$\mathit{Value} \text{ erases to } \mathit{UntypedValue}.$$

XQuery formal semantics (not in paper)

- Dynamic Semantics

$$DynEnv \vdash Expr \Rightarrow Value$$

- Static Semantics

$$StatEnv \vdash Expr : Type$$

- Type Soundness

  Theorem If

$$DynEnv \vdash Expr \Rightarrow Value$$
$$StatEnv \vdash Expr : Type$$

  then

$$Value \text{ matches } Type.$$

Success stories

- XQuery has two specifications, one in prose and one using formal methods — one of the first uses of formal methods in an industrial standard.

- Formalization of *named typing* raised ten issues not resolved in the prose specification.

- XQuery face-to-face, Chapel Hill, NC, 17–18 October 2002: After presentation of formal semantics of *pure named typing*, it was accepted *without dissent*. In the two-day meeting, this was the *only* decision adopted without dissent.

- Our techniques also adopted by James Clark and Makoto Murata to formalize Relax NG, another industrial standard.

RELAX NG Specification

Committee Specification 11 August 2001

This version:

Committee Specification: 11 August 2001

Editors:

James Clark <jjc@jclark.com>, MURATA Makoto <mura034@attglobal.net>

6.2. Patterns

The axioms and inference rules for patterns use the following notation:

p
: ranges over patterns (elements matching the pattern production)

$cx \mid\!\!- a; m =\!\sim p$
: asserts that with respect to context cx, the attributes a and the sequence of elements and strings m matches the pattern p

6.2.1. `choice` pattern

The semantics of the `choice` pattern are as follows:

$$\text{(choice 1)}\quad \frac{cx \mid\!\!- a; m =\!\sim p_1}{cx \mid\!\!- a; m =\!\sim \texttt{<choice>}\, p_1\, p_2\, \texttt{</choice>}}$$

$$\text{(choice 2)}\quad \frac{cx \mid\!\!- a; m =\!\sim p_2}{cx \mid\!\!- a; m =\!\sim \texttt{<choice>}\, p_1\, p_2\, \texttt{</choice>}}$$

Action items

- Paper in POPL proceedings misprinted; get it from the web.

- Review XQuery and send us your comments!