An XML-Based Standard for Representing Linear Programming Problem Instances

Robert Fourer
Industrial Engineering & Management Sciences
Northwestern University, Evanston, IL, USA
4er@iems.northwestern.edu

Leo Lopes
Systems & Industrial Engineering Department
University of Arizona, Tucson, AZ, USA
leo@sie.arizona.edu

Kipp Martin
Graduate School of Business
University of Chicago, Chicago, IL, USA
kipp.martin@gsb.uchicago.edu

APMOD 2004
Brunel University, London — Tuesday, June 22, 2004 — TB32
XML-Based Standard Formats

Motivation
- for any standard format
- for an XML-based format

Proposals (see http://senna.iems.nwu.edu/xml/)
- OptML
- SNOML
- LPFML . . .

Aspects of LPFML
- Examples
- Schemas
- Libraries
- Compression
Standard formats

XML Means “Tags” . . .

Example: html for a popular home page

```html
<html><head><meta http-equiv="content-type" content="text/html; charset=UTF-8"><title>Google</title><style><![CDATA[
body,td,a,p,.h{font-family:arial,sans-serif;}
.h{font-size: 20px;}
.q{text-decoration:none; color:#0000cc;}
//-->
</style></head><body bgcolor=#ffffff text=#000000 link=#0000cc vlink=#551a8b alink=#ff0000 onLoad=sf()><center><table border=0 cellspacing=0 cellpadding=0><tr><td><img src="/images/logo.gif" width=276 height=110 alt="Google"></td></tr></table><br>........<font size=-2>&copy;2003 Google - Searching 3,307,998,701 web pages</font></center></body></html>
```

. . . a collection of XML tags is designed for a special purpose
. . . by use of a schema written itself in XML
Advantage of any standard

**MN drivers**
*without a standard*

**M + N drivers**
*with a standard*
Advantages of an XML Standard

Specifying it

- Unambiguous definition via a schema
- Provision for keys and data typing
- Well-defined expansion to new name spaces

Working with it

- Parsing and validation via standard utilities
- Amenability to compression and encryption
- Transformation and display via XSLT style sheets
- Compatibility with web services
Standard formats

What about “MPS Format”?

Weaknesses

- Standard only for LP and MIP, not for nonlinear, network, complementarity, logical, . . .
- Standard not uniform (especially for extensions)
- Verbose ASCII form, with much repetition of names
- Limited precision for some numerical values

Used for

- Collections of (mostly anonymous) test problems
- Bug reports to solver vendors

Not used for

- Communication between modeling languages & solvers
Example: AMPL Model and Data

```AMPL
set RES;  # resources
set PRD;  # products

param hrs {RES};      # hrs of resource i available
param prf {PRD};      # profit per unit of product j
param act {RES,PRD};  # res i consumed by 1 unit if product j

var Make {PRD} >= 0;  # units of product j to be made

maximize TotPrf:
    sum {j in PRD} prf[j] * Make[j];

subject to HrsAvl {i in RES}:
    sum {j in PRD} act[i,j] * Make[j] <= hrs[i];

param: RES: hrs := cutdye 630 sew 600 finish 708 pack 135 ;
param: PRD: prf := RC 10 LFA 9 ;
param act (tr): cutdye sew finish pack :=
    RC  0.7  0.5  1.0  0.1
    LFA  1.0  0.0  0.6667 0.25;
```
Standard formats

Example: MPS Format

<table>
<thead>
<tr>
<th>NAME</th>
<th>PRODMIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROWS</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>TOTPROF</td>
</tr>
<tr>
<td>L</td>
<td>HRAV1</td>
</tr>
<tr>
<td>L</td>
<td>HRAV2</td>
</tr>
<tr>
<td>L</td>
<td>HRAV3</td>
</tr>
<tr>
<td>L</td>
<td>HRAV4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COLUMNS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MAKE1</td>
<td>TOTPROF</td>
</tr>
<tr>
<td>MAKE1</td>
<td>HRAV1</td>
</tr>
<tr>
<td>MAKE1</td>
<td>HRAV3</td>
</tr>
<tr>
<td>MAKE2</td>
<td>TOTPROF</td>
</tr>
<tr>
<td>MAKE2</td>
<td>HRAV1</td>
</tr>
<tr>
<td>MAKE2</td>
<td>HRAV3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RHS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RHS1</td>
<td>HRAV1</td>
</tr>
<tr>
<td>RHS1</td>
<td>HRAV2</td>
</tr>
<tr>
<td>RHS1</td>
<td>HRAV3</td>
</tr>
<tr>
<td>RHS1</td>
<td>HRAV4</td>
</tr>
</tbody>
</table>

ENDATA
Standard formats

Example: AMPL’s “nl” Format

```
g3 2 1 0
 2 3 1 0 0
 0 0
 0 0
 0 0 0
 0 0 0 1
 0 0 0 0 0
 6 2
 0 0
 0 0 0 0 0
C0 n0
C1 n0
C2 n0
O0 1
n0

----->
```

```
r
 1 630
 1 708
 1 135
b
 0 0 1200
 2 0
k1
 3
J0 2
 0 0.7
 1 1
J1 2
 0 1
 1 0.6667
J2 2
 0 0.1
 1 0.25
G0 2
 0 10
 1 9
```
Standard formats

Example: LPFML

First we’ll show . . .
- Diagrams of parts of the LPFML schema
- Corresponding XML for the example

Then we’ll see . . .
- Actual schema files
Standard formats

**linearProgramDescription**

```xml
<linearProgramDescription>
  <source>Par Inc. Problem from Anderson, Sweeny, and Williams</source>
  <maxOrMin>max</maxOrMin>
  <numberRows>4</numberRows>
  <numberVars>2</numberVars>
</linearProgramDescription>
```
Standard formats

**linearProgramData**

```
<rows>
  <row rowName="HrsAvl[cutdye]" rowUB="630"/>
  <row rowName="HrsAvl[sew]" rowUB="600"/>
  <row rowName="HrsAvl[finish]" rowUB="708"/>
  <row rowName="HrsAvl[pack]" rowUB="135"/>
</rows>

<columns>
  <col objVal="10" colName="Make[RC]" colType="C" colLB="0.0"/>
  <col objVal="9" colName="Make[LFA]" colType="C" colLB="0.0"/>
</columns>
```
Standard formats

amatrx

... optional base-64 encoding of vectors
Standard formats

linearProgramSolution

```
<linearProgramSolution>
  <primalSolution>
    <sol idx="1" name="Make1" val="540"/>
    <sol idx="2" name="Make2" val="252"/>
  </primalSolution>
  <dualSolution>
    <sol idx="1" name="cutdye" val="4.37457"/>
    <sol idx="3" name="finish" val="6.9378"/>
  </dualSolution>
  <optimalValue>7667.94</optimalValue>
  <status statusId="optimalSolutionFound">maximum
    primal infeas 1.3e-7 dual infeas 2.7e-6</status>
  <solverMessage>XPLEX 14.7
dual simplex optimizer with superpivot</solverMessage>
</linearProgramSolution>
```
Standard formats

Schema for <rows> Element

```xml
<xs:element name="rows">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="row" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:attribute name="rowName" type="xs:string" use="optional"/>
          <xs:attribute name="rowUB" type="xs:double" use="optional"/>
          <xs:attribute name="rowLB" type="xs:double" use="optional"/>
          <xs:attribute name="mult" type="xs:int" use="optional"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```
Standard formats

Schema for <intVector> Type

```xml
<xs:complexType name="intVector">
  <xs:choice>
    <xs:element name="base64BinaryData" type="base64BinaryData"/>
    <xs:element name="el" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:simpleContent>
          <xs:extension base="xs:int">
            <xs:attribute name="mult" type="xs:int" use="optional"/>
            <xs:attribute name="incr" type="xs:int" use="optional"/>
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
  </xs:choice>
</xs:complexType>
```
Schema for `<colType>` Simple Type

```xml
<xs:simpleType name="colType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="C"/>
    <xs:enumeration value="B"/>
    <xs:enumeration value="I"/>
  </xs:restriction>
</xs:simpleType>
```

*Standard formats*
Libraries

*Read and write LP instances in LPFML format*

- Allow format to be used immediately
- Hide all parsing code
- Allow for future changes and extensions without rewriting code

... *major contribution of this work*
Parsing Library

FMLHandler class

- Aggregate data from LPFML into rows, columns, and other LP components
- Methods startElement, endElement, etc.

FMLParser class

- Virtual methods for setting up LP components
- Derived class for each solver
- For each LP component, derived methods implement LPFML input to individual solvers
- Event driven: Derived method only called after component has been parsed

... also derived methods for starting solver & writing solution
Parsing Library (cont’d)

Example

- Class FMLLINDOParser derived from FMLParser
- Virtual methods such as `onObjectiveSense`, `onConstraints`, etc.
  replaced by LINDO-specific routines

Advantages of event-driven approach

- Avoid searching the LPFML file
- Reduce number of copies of data that must exist at one time
Current Parsing Library

Classes inheriting from FMLParser

- FMLCOINParser
  * Creates CoinPackedMatrix data structure
- FMLOSIParser
  * Connects to any solver that has an Open Solver Interface implementation
- FMLLINDOParser
  * Supports data structures of the LINDO API

Utilities

- nl2fml
- FMLCOINMPSToXML
- FMLLINDOToXML

... implement interface between AMPL and any solver that supports the Open Solver Interface
Communicating Instances

*Tightly coupled environments*
- Modeling system & solver communicating directly on the same machine
- *Parsing time* is the primary concern

*Loosely coupled environments*
- Modeling system & solver reside on different machines and networks
- *File size* is the primary concern

... *tests on 15 largest netlib problems*
Compression

*LPFML-specific space-saving features*
- Collapse sequences of row/column numbers
- Collapse repeated element values
- Base-64 representation of arrays

*Comparisons without compression*
- MPS > LPFML > base-64 LPFML >> AMPL nl

*Comparisons with compression*
- gzipped MPS $\approx 2 \times$ gzipped LPFML
- gzipped LPFML $\approx 1.5 \times$ bzipped LPFML
  * bzip2 reorders file before searching for patterns
- gzipped LPFML $\approx 1.65 \times$ xmilled LPFML
  * xmill uses XML-specific compression techniques
Parsing Time

*File-based using base-64 encoding*
- Specialized LPFML \( \approx \) COIN MPS
- Generic Xerces LPFML \( \approx 3-4 \times \) COIN MPS

*In-memory using base-64 encoding*
- Generic Xerces LPFML \( \approx \) COIN MPS
Extensions to Come

*Quadratic*

- Matrix of coefficients for each quadratic objective or constraint

*Stochastic*

*Nonlinear*

- Algebraic expressions
- Logical expressions
Distribution

*Open source*
- Source code available without additional charge
- License does not require that modifications or redistributions be open source

*Availability*
- Download from [gsbkip.uchicago.edu/fml/fml.html](gsbkip.uchicago.edu/fml/fml.html)
- Available for Windows and Linux