An Open Interface for Hooking Solvers to Modeling Systems

Robert Fourer, Jun Ma

Industrial Engineering & Management Sciences
Northwestern University

`[4er,maj]@iems.northwestern.edu`

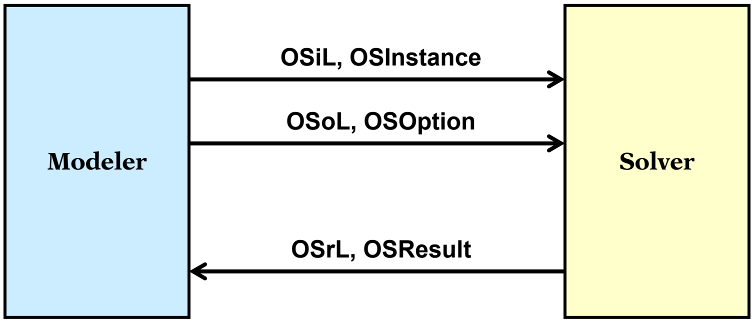
Kipp Martin

Graduate School of Business
University of Chicago

`kmartin@gsb.uchicago.edu`

INFORMS International
Session TD08.3, July 10, 2007

Quick Overview



XML text files

- ➢OSiL, OSoL, OSrL

In-memory data structures

- ➢OSInstance, OSOption, OSResult

The Modeling System Interface

Motivation

- ➢ For any standard format
- ➢ For an XML-based format

Text files

- ➢ XML schema
- ➢ OSiL example
- ➢ Compression
- ➢ Extensions

In-memory data structures

- ➢ Objects and methods
- ➢ Writing a generator
- ➢ Translating from a modeling language

Motivation

XML Means "Tagged" Text Files . . .

Example: html for a popular home page

```
<html><head><meta http-equiv="content-type" content="text/html;
charset=UTF-8"><title>Google</title><style><!--
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></p></center></body></html>
```

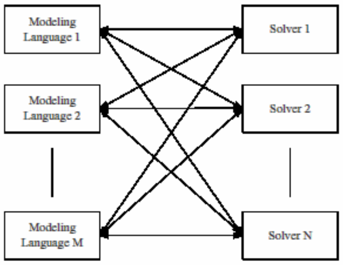
. . . a collection of XML tags is designed for a special purpose

. . . by use of a schema written itself in XML

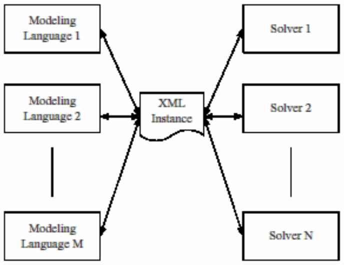
Motivation

Advantage of any standard

MN drivers
without a standard



M + N drivers
with a standard



Motivation

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***

Motivation

What about “MPS Form”?

Weaknesses

- Standard only for LP and MIP, not for
  nonlinear, network, complementarity, logical, . . .
- Standard not uniform (especially for SP extension)
- 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 systems and solvers**

Text files

Example: A Problem Instance (in AMPL)

```
ampl: display _varname, _var.lb, _var.init, _var.ub;

:          _varname        _var.lb _var.init   _var.ub     :=
1    "Trans['GARY','FRA']"     5        0      Infinity
2    "Trans['GARY','DET']"     5        0      Infinity

ampl: expand _obj;

minimize Obj:
        (1 - x[0])^2 + 100*(x[1] - x[0]^2)^2 + 9*x[1];

ampl: expand _con;

subject to Con1:
        10*x[0]^2 + 11*x[1]^2 + 3*x[0]*x[1] + x[0] <= 10;

subject to Con2:
        log(x[0]*x[1]) + 7*x[0] + 5*x[1] >= 10;
```

Text files

Text from the OSiL Schema

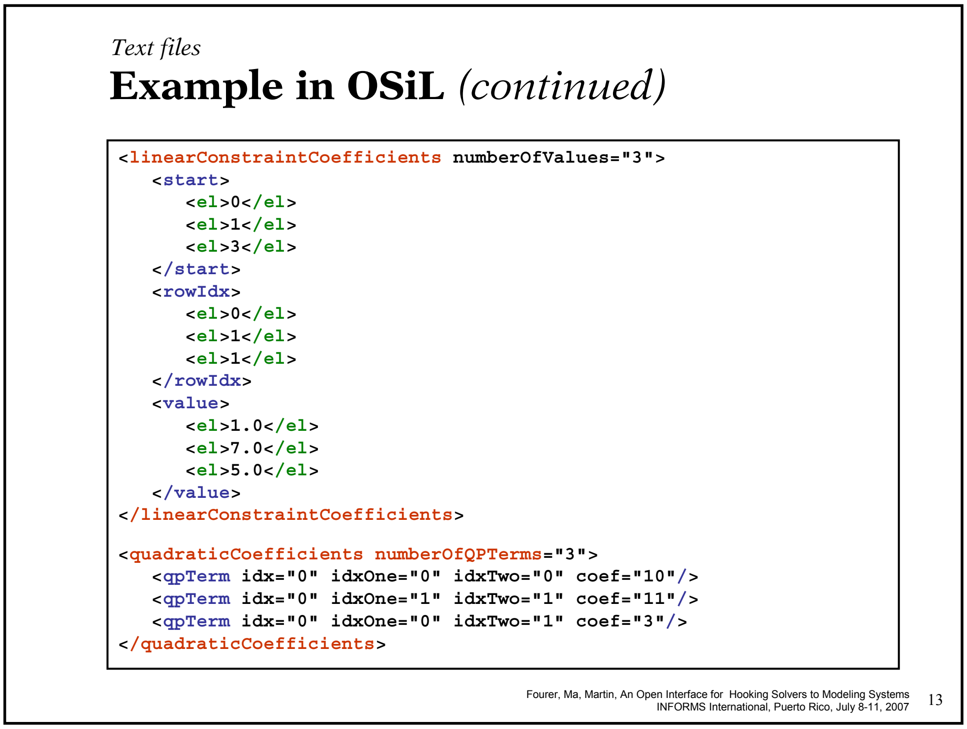
```
<xs:complexType name="Variables">
  <xs:sequence>
    <xs:element name="var" type="Variable" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="number" type="xs:positiveInteger" use="required"/>
</xs:complexType>
```

```
<xs:complexType name="Variable">
  <xs:attribute name="name" type="xs:string" use="optional"/>
  <xs:attribute name="init" type="xs:string" use="optional"/>
  <xs:attribute name="type" use="optional" default="C">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="C"/>
      <xs:enumeration value="B"/>
      <xs:enumeration value="I"/>
      <xs:enumeration value="S"/>
    </xs:restriction>
  </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="lb" type="xs:double" use="optional" default="0"/>
  <xs:attribute name="ub" type="xs:double" use="optional" default="INF"/>
</xs:complexType>
```

Text files

Example in OSiL

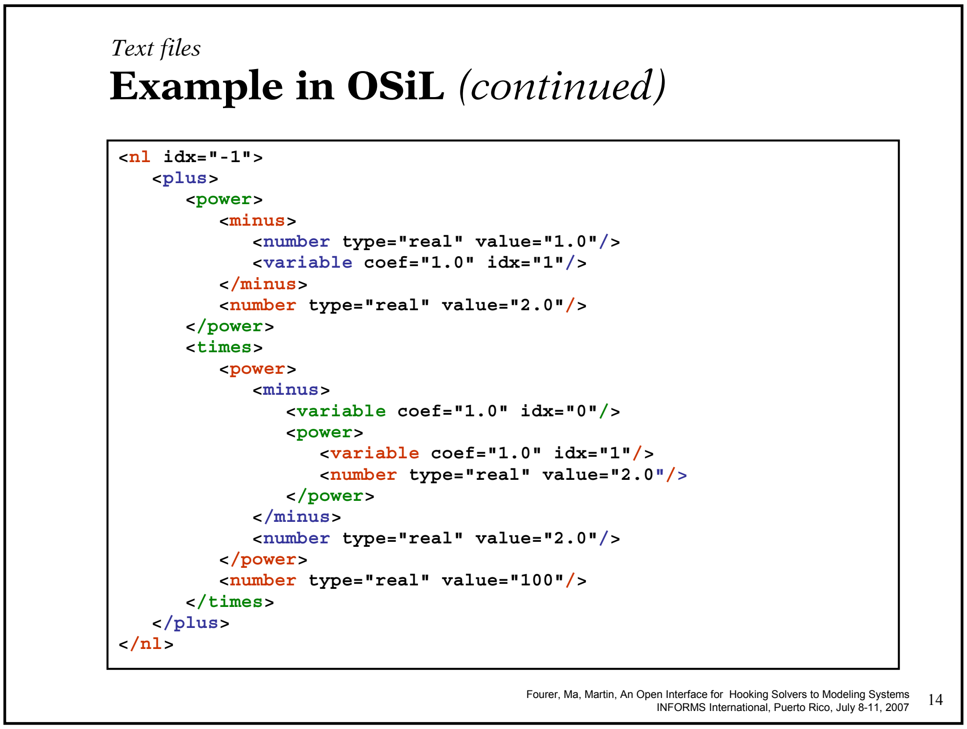
```
<instanceHeader>
   <name>Modified Rosenbrock</name>
   <source>Computing Journal3:175-184, 1960</source>
   <description>Rosenbrock problem with constraints</description>
</instanceHeader>

<variables number="2">
   <var lb="5" name="x0" type="C"/>
   <var lb="5" name="x1" type="C"/>
</variables>

<objectives number="1">
   <obj maxOrMin="min" name="RError" numberOfObjCoef="1">
      <coef idx="1">9</coef>
   </obj>
</objectives>

<constraints number="2">
   <con ub="10.0" name="Quad"/>
   <con lb="10.0" name="LogQuad"/>
</constraints>
```



Text files

Example in OSiL *(continued)*

```
<linearConstraintCoefficients numberOfValues="3">
   <start>
      <el>0</el>
      <el>1</el>
      <el>3</el>
   </start>
   <rowIdx>
      <el>0</el>
      <el>1</el>
      <el>1</el>
   </rowIdx>
   <value>
      <el>1.0</el>
      <el>7.0</el>
      <el>5.0</el>
   </value>
</linearConstraintCoefficients>

<quadraticCoefficients numberOfQPTerms="3">
   <qpTerm idx="0" idxOne="0" idxTwo="0" coef="10"/>
   <qpTerm idx="0" idxOne="1" idxTwo="1" coef="11"/>
   <qpTerm idx="0" idxOne="0" idxTwo="1" coef="3"/>
</quadraticCoefficients>
```



Text files

Example in OSiL *(continued)*

```
<nl idx="-1">
   <plus>
      <power>
         <minus>
            <number type="real" value="1.0"/>
            <variable coef="1.0" idx="1"/>
         </minus>
         <number type="real" value="2.0"/>
      </power>
      <times>
         <power>
            <minus>
               <variable coef="1.0" idx="0"/>
               <power>
                  <variable coef="1.0" idx="1"/>
                  <number type="real" value="2.0"/>
               </power>
            </minus>
            <number type="real" value="2.0"/>
         </power>
         <number type="real" value="100"/>
      </times>
   </plus>
</nl>
```

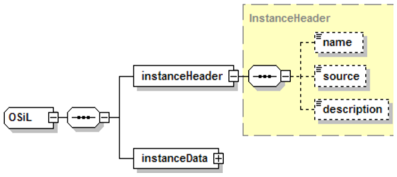
Text files

Example in OSiL *(continued)*

```
<nl idx="1">
   <ln>
      <times>
         <variable idx="0"/>
         <variable idx="1"/>
      </times>
   </ln>
</nl>
```

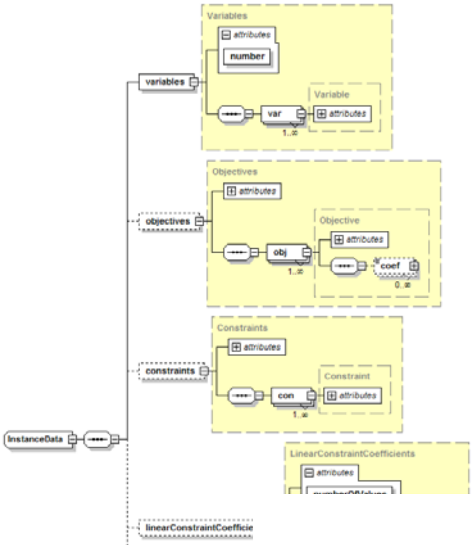
Text files

Diagram of the OSiL Schema



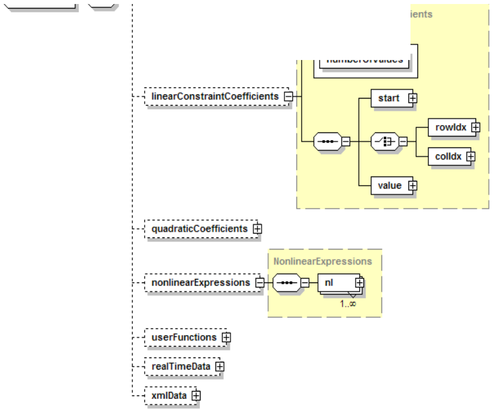
Text files

Details of OSiL's instanceData Element

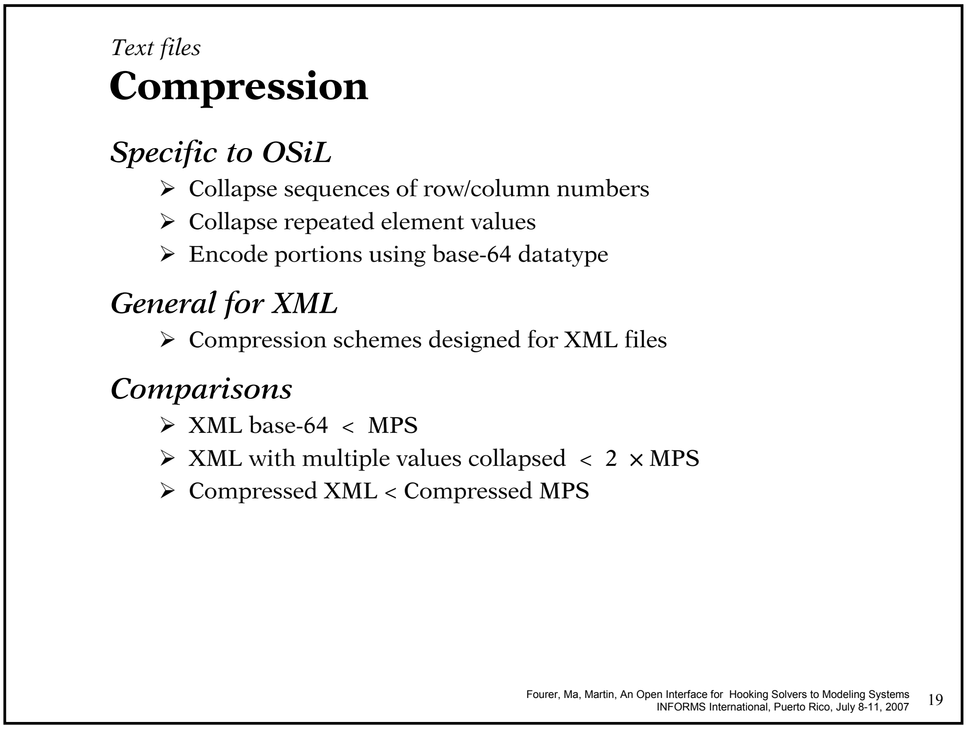


Text files

Details of OSiL's instanceData Element



Generated with XMLSpy Schema Editor www.altova.com



Text files

Compression

Specific to OSiL

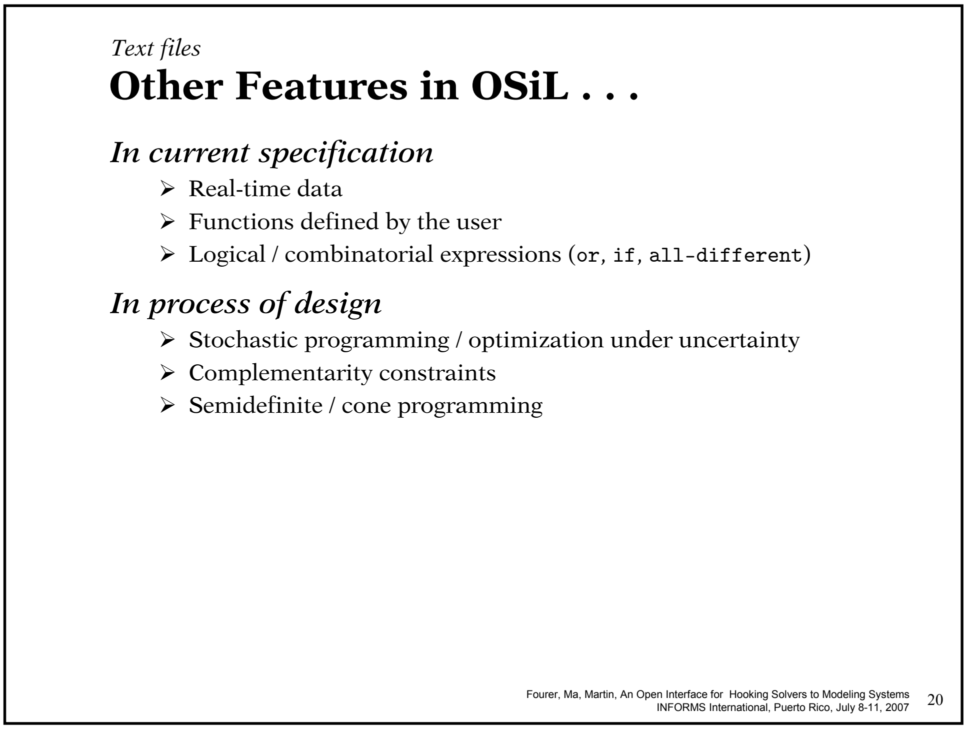
- Collapse sequences of row/column numbers
- Collapse repeated element values
- Encode portions using base-64 datatype

General for XML

- Compression schemes designed for XML files

Comparisons

- XML base-64 < MPS
- XML with multiple values collapsed < 2 × MPS
- Compressed XML < Compressed MPS



Text files

Other Features in OSiL . . .

In current specification

- Real-time data
- Functions defined by the user
- Logical / combinatorial expressions (`or`, `if`, `all-different`)

In process of design

- Stochastic programming / optimization under uncertainty
- Complementarity constraints
- Semidefinite / cone programming

In-Memory Data Structures

OSInstance object class

- ➢ Parallels the OSiL schema
- ➢ complexType in schema ⟵⟶ class in OSInstance
- ➢ attributes / children of an element ⟵⟶ members of a class
- ➢ choices / sequences in the schema ⟵⟶ array members

OS expression tree

- ➢ Parallels the *nonlinear* part of the OSiL schema
- ➢ Designed to avoid lengthy “switch” statements

Creating an OSInstance

- ➢ Writing a generator
- ➢ Translating from AMPL

. . . similar handling of OSOption, OSResult

In-memory data structure

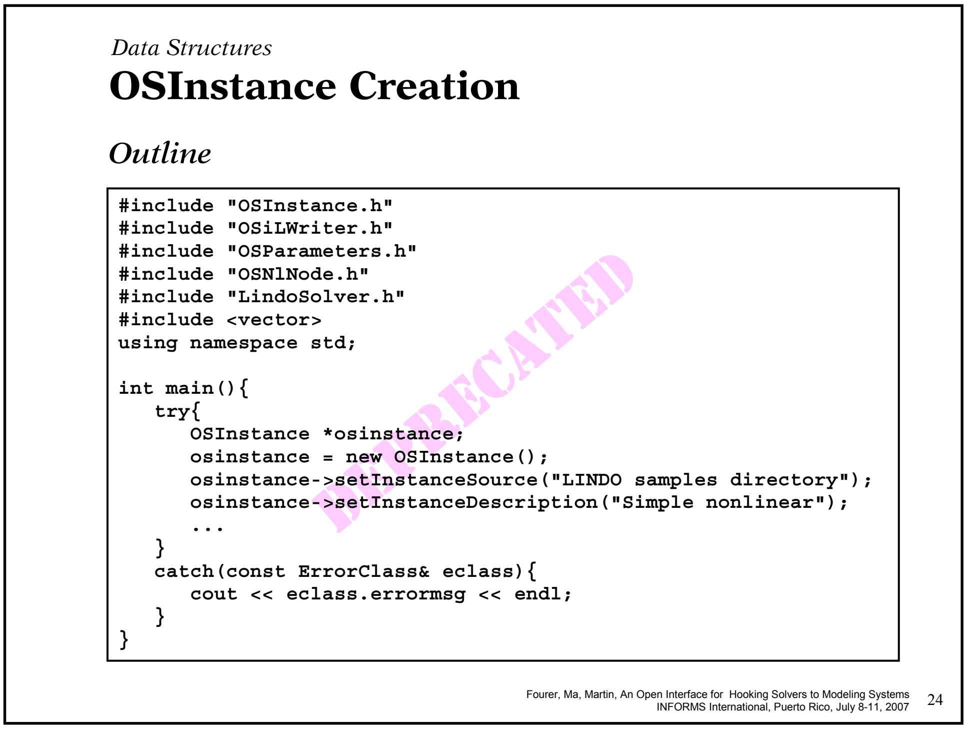
Example

Schema complexType

```
<xs:complexType name="Variables">
  <xs:sequence>
    <xs:element name="var" type="Variable" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="number" type="xs:positiveInteger" use="required"/>
</xs:complexType>
```

In-memory class

```
class Variables{
public;
   Variables();
   Variable *var;
   int number;
}; // class Variables
```

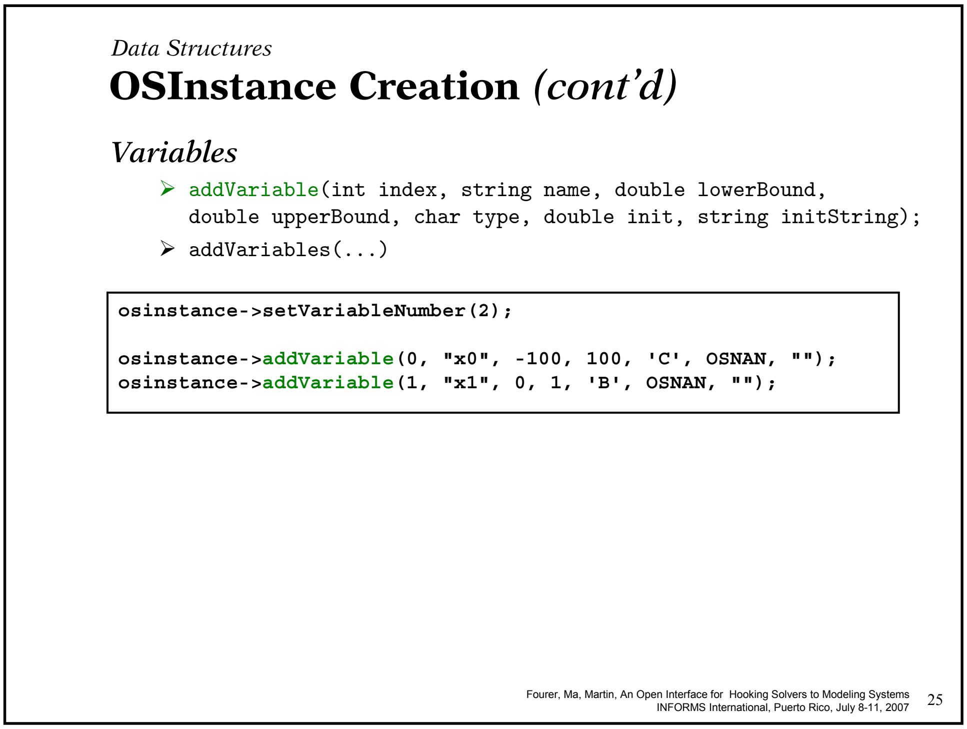


Data Structures

OSInstance Creation

Outline

```
#include "OSInstance.h"
#include "OSiLWriter.h"
#include "OSParameters.h"
#include "OSNlNode.h"
#include "LindoSolver.h"
#include <vector>
using namespace std;

int main(){
   try{
      OSInstance *osinstance;
      osinstance = new OSInstance();
      osinstance->setInstanceSource("LINDO samples directory");
      osinstance->setInstanceDescription("Simple nonlinear");
      ...
   }
   catch(const ErrorClass& eclass){
      cout << eclass.errormsg << endl;
   }
}
```



Data Structures

OSInstance Creation *(cont'd)*

Variables

- addVariable(int index, string name, double lowerBound, double upperBound, char type, double init, string initString);
- addVariables(...)

```
osinstance->setVariableNumber(2);

osinstance->addVariable(0, "x0", -100, 100, 'C', OSNAN, "");
osinstance->addVariable(1, "x1", 0, 1, 'B', OSNAN, "");
```

Data Structures

OSInstance Creation *(cont'd)*

Objective

- bool addObjective(int index, string name,
  string maxOrMin, double constant, double weight,
  SparseVector* objectiveCoefficients);

```
osinstance->setObjectiveNumber(1);

SparseVector *objcoeff;
objcoeff = new SparseVector(1);
objcoeff->indexes = new int[1];
objcoeff->values = new double[1];
objcoeff->indexes[0] = 1;
objcoeff->values[0] = .4;

osinstance->addObjective
   (-1, "objfunction", "max", 0.0, 1.0, objcoeff);
```

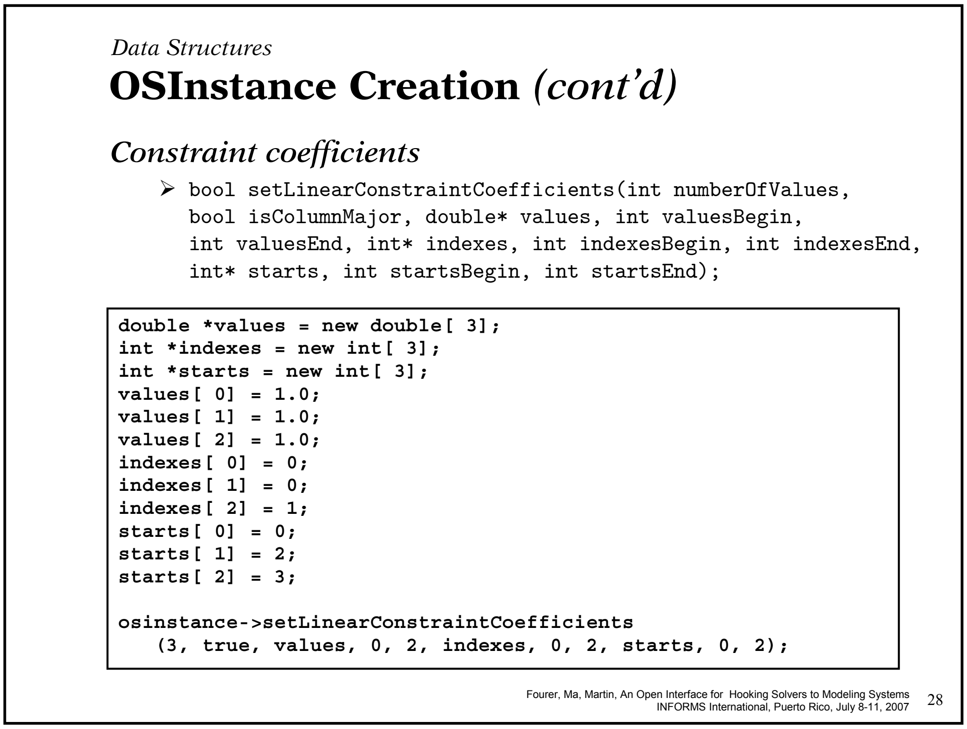
Data Structures

OSInstance Creation *(cont'd)*

Constraints

- bool addConstraint(int index, string name,
  double lowerBound, double upperBound, double constant);
- bool addConstraints(...)

```
osinstance->setConstraintNumber(6);

osinstance->addConstraint(0, "row0", -OSINFINITY, 4, 0);
osinstance->addConstraint(1, "row1", -OSINFINITY, 6, 0);
osinstance->addConstraint(2, "row2", -OSINFINITY, 0, 0);
osinstance->addConstraint(3, "row3", 0 , OSINFINITY, 0);
osinstance->addConstraint(4, "row4", -OSINFINITY, 0, 0);
osinstance->addConstraint(5, "row5", -OSINFINITY, 0, 0);
```



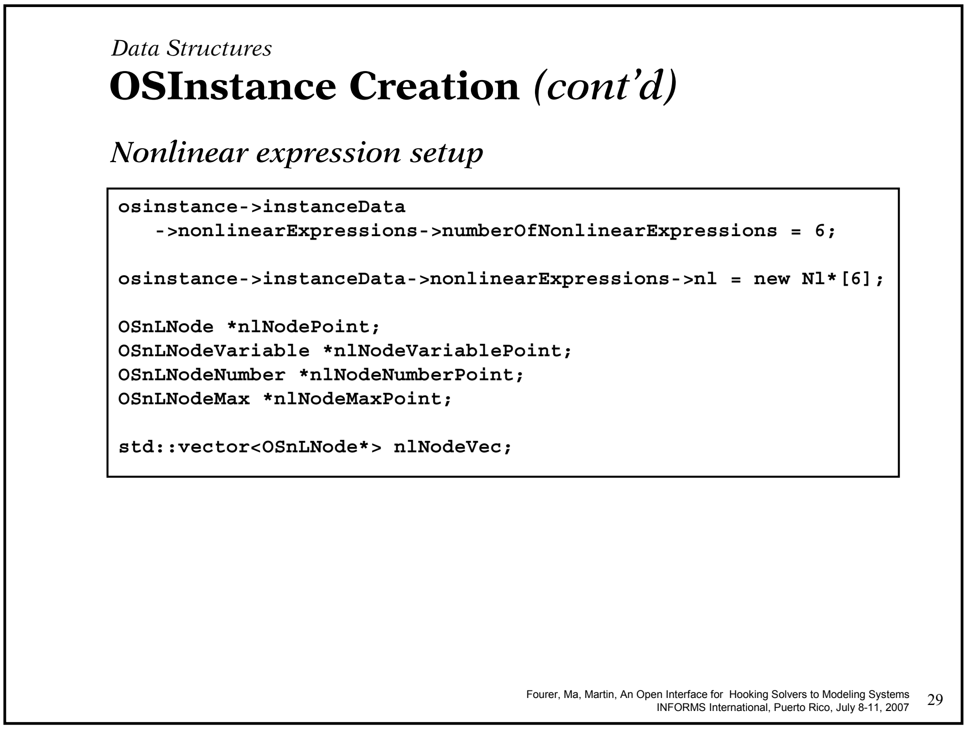
Data Structures

OSInstance Creation *(cont'd)*

Constraint coefficients

- `bool setLinearConstraintCoefficients(int numberOfValues, bool isColumnMajor, double* values, int valuesBegin, int valuesEnd, int* indexes, int indexesBegin, int indexesEnd, int* starts, int startsBegin, int startsEnd);`

```
double *values = new double[ 3];
int *indexes = new int[ 3];
int *starts = new int[ 3];
values[ 0] = 1.0;
values[ 1] = 1.0;
values[ 2] = 1.0;
indexes[ 0] = 0;
indexes[ 1] = 0;
indexes[ 2] = 1;
starts[ 0] = 0;
starts[ 1] = 2;
starts[ 2] = 3;

osinstance->setLinearConstraintCoefficients
   (3, true, values, 0, 2, indexes, 0, 2, starts, 0, 2);
```

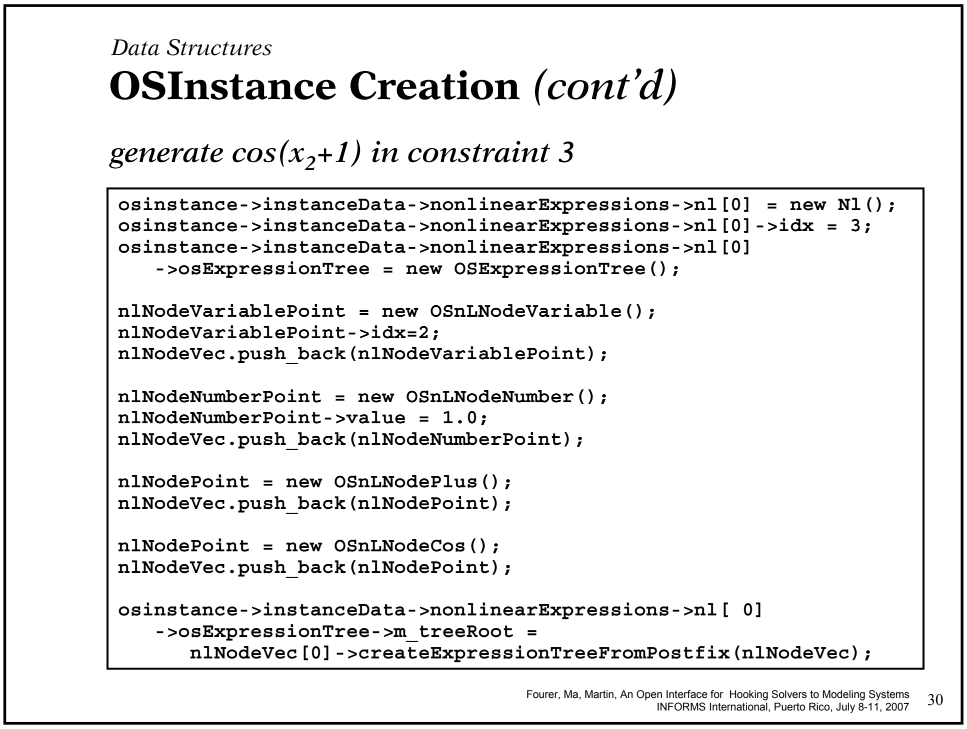


Data Structures

OSInstance Creation *(cont'd)*

Nonlinear expression setup

```
osinstance->instanceData
   ->nonlinearExpressions->numberOfNonlinearExpressions = 6;

osinstance->instanceData->nonlinearExpressions->nl = new Nl*[6];

OSnLNode *nlNodePoint;
OSnLNodeVariable *nlNodeVariablePoint;
OSnLNodeNumber *nlNodeNumberPoint;
OSnLNodeMax *nlNodeMaxPoint;

std::vector<OSnLNode*> nlNodeVec;
```

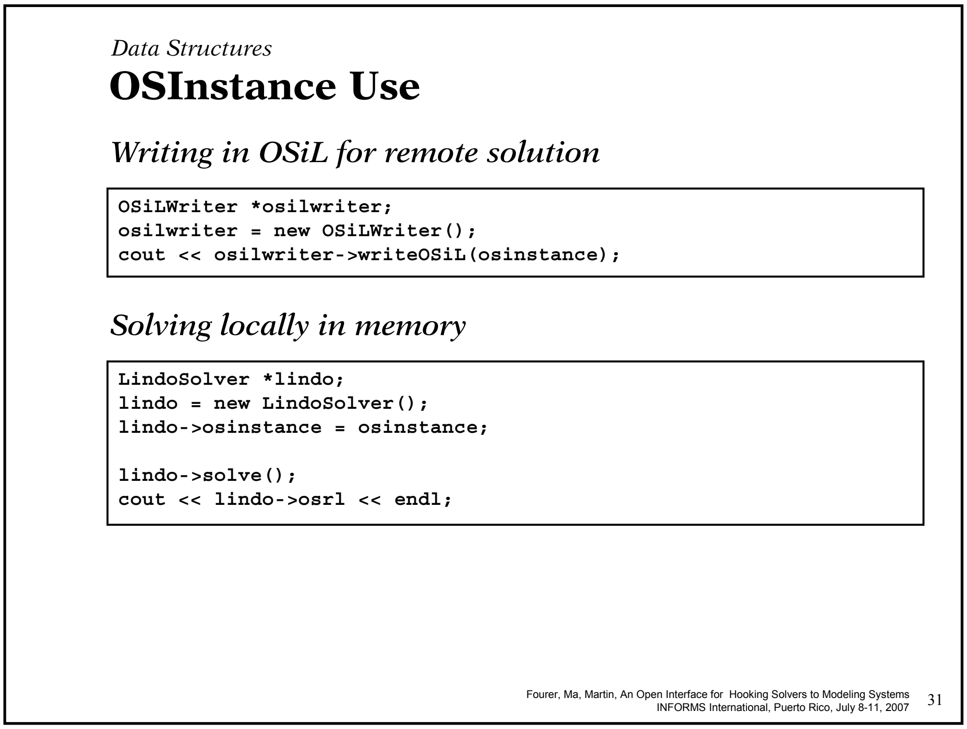


Data Structures

OSInstance Creation *(cont'd)*

generate cos(x_2+1) in constraint 3

```
osinstance->instanceData->nonlinearExpressions->nl[0] = new Nl();
osinstance->instanceData->nonlinearExpressions->nl[0]->idx = 3;
osinstance->instanceData->nonlinearExpressions->nl[0]
   ->osExpressionTree = new OSExpressionTree();

nlNodeVariablePoint = new OSnLNodeVariable();
nlNodeVariablePoint->idx=2;
nlNodeVec.push_back(nlNodeVariablePoint);

nlNodeNumberPoint = new OSnLNodeNumber();
nlNodeNumberPoint->value = 1.0;
nlNodeVec.push_back(nlNodeNumberPoint);

nlNodePoint = new OSnLNodePlus();
nlNodeVec.push_back(nlNodePoint);

nlNodePoint = new OSnLNodeCos();
nlNodeVec.push_back(nlNodePoint);

osinstance->instanceData->nonlinearExpressions->nl[ 0]
   ->osExpressionTree->m_treeRoot =
      nlNodeVec[0]->createExpressionTreeFromPostfix(nlNodeVec);
```



Data Structures

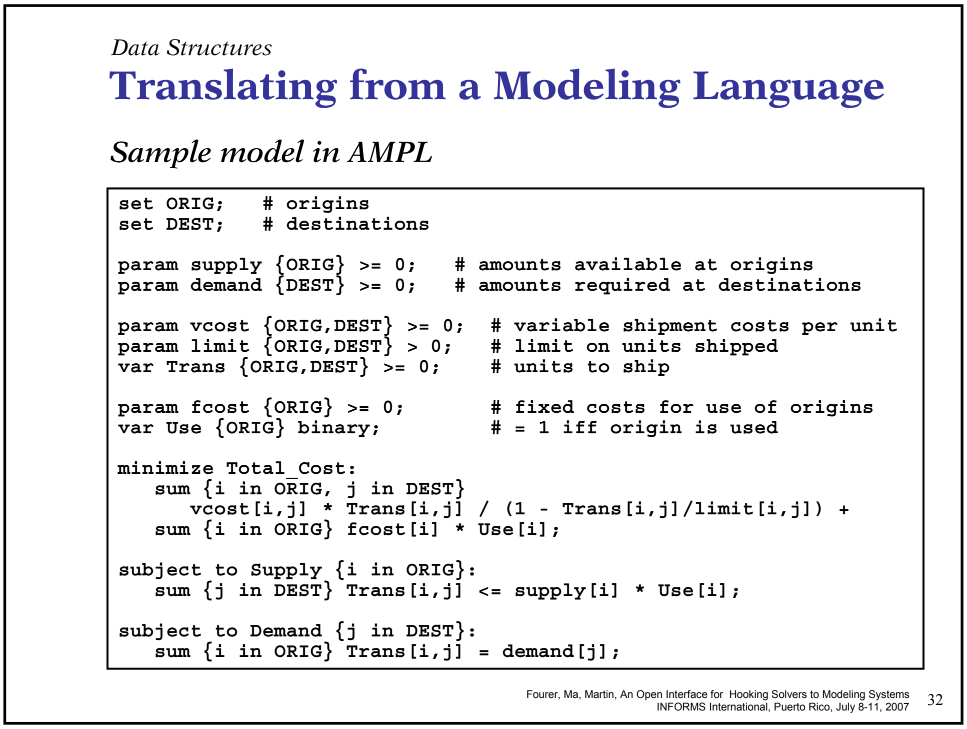
OSInstance Use

Writing in OSiL for remote solution

```
OSiLWriter *osilwriter;
osilwriter = new OSiLWriter();
cout << osilwriter->writeOSiL(osinstance);
```

Solving locally in memory

```
LindoSolver *lindo;
lindo = new LindoSolver();
lindo->osinstance = osinstance;

lindo->solve();
cout << lindo->osrl << endl;
```

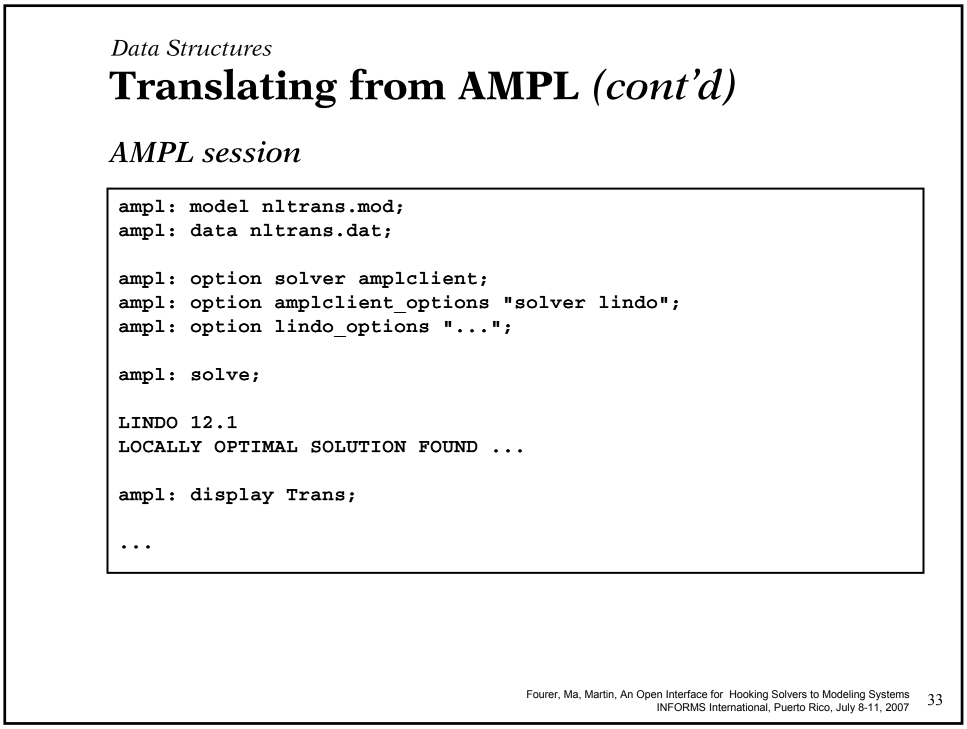


Data Structures

Translating from a Modeling Language

Sample model in AMPL

```
set ORIG;   # origins
set DEST;   # destinations

param supply {ORIG} >= 0;   # amounts available at origins
param demand {DEST} >= 0;   # amounts required at destinations

param vcost {ORIG,DEST} >= 0;  # variable shipment costs per unit
param limit {ORIG,DEST} > 0;   # limit on units shipped
var Trans {ORIG,DEST} >= 0;    # units to ship

param fcost {ORIG} >= 0;       # fixed costs for use of origins
var Use {ORIG} binary;         # = 1 iff origin is used

minimize Total_Cost:
   sum {i in ORIG, j in DEST}
      vcost[i,j] * Trans[i,j] / (1 - Trans[i,j]/limit[i,j]) +
   sum {i in ORIG} fcost[i] * Use[i];

subject to Supply {i in ORIG}:
   sum {j in DEST} Trans[i,j] <= supply[i] * Use[i];

subject to Demand {j in DEST}:
   sum {i in ORIG} Trans[i,j] = demand[j];
```



Data Structures

Translating from AMPL *(cont'd)*

AMPL session

```
ampl: model nltrans.mod;
ampl: data nltrans.dat;

ampl: option solver amplclient;
ampl: option amplclient_options "solver lindo";
ampl: option lindo_options "...";

ampl: solve;

LINDO 12.1
LOCALLY OPTIMAL SOLUTION FOUND ...

ampl: display Trans;

...
```

Data Structures

Translating from AMPL *(cont'd)*

OSiL derived from AMPL's output format

```
<osil xmlns="os.optimizationservices.org"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation=
        "os.optimizationservices.org ../schemas/OSiL.xsd">
    <instanceHeader>
        <description>Generated from AMPL nl file</description>
    </instanceHeader>
    <instanceData>
        <variables numberOfVariables="24">
            <var name="_svar[1]"/>
            <var name="_svar[2]"/>
            ...
            <var name="_svar[22]" type="B" ub="1"/>
            <var name="_svar[23]" type="B" ub="1"/>
            <var name="_svar[24]" type="B" ub="1"/>
        </variables>
        ....
    </instanceData>
</osil>
```

Data Structures

Translating from AMPL *(cont'd)*

OSiL derived from AMPL's output format

```
<objectives numberOfObjectives="1">
    <obj maxOrMin="min" numberOfObjCoef="24">
        <coef idx="21">50000</coef>
                <coef idx="22">3.94e+06</coef>
                <coef idx="23">370000</coef>
    </obj>
</objectives>
<constraints numberOfConstraints="10">
    <con name="_scon[1]" ub="-0"/>
    <con name="_scon[2]" ub="-0"/>
    <con name="_scon[3]" ub="-0"/>
    <con name="_scon[4]" lb="900" ub="900"/>
    <con name="_scon[5]" lb="1200" ub="1200"/>
    <con name="_scon[6]" lb="600" ub="600"/>
    <con name="_scon[7]" lb="400" ub="400"/>
    <con name="_scon[8]" lb="1700" ub="1700"/>
    <con name="_scon[9]" lb="1100" ub="1100"/>
    <con name="_scon[10]" lb="1000" ub="1000"/>
</constraints>
```

Data Structures

Translating from AMPL *(cont'd)*

OSiL derived from AMPL's output format

```
<linearConstraintCoefficients numberOfValues="45">
    <start>
        <el>0</el>
        <el>2</el>
        <el>4</el>
        ...
    </start>
    <rowIdx>
        <el>0</el>
        <el>3</el>
        <el>0</el>
        <el>4</el>
        <el>0</el>
        <el>5</el>
        ...
    </rowIdx>
    <value>
        <el>1</el>
        ...
```

Data Structures

Translating from AMPL *(cont'd)*

OSiL derived from AMPL's output format

```
<linearConstraintCoefficients numberOfValues="45">
    <start>
        ...
    </start>
    <rowIdx>
        ...
    </rowIdx>
    <value>
        <el>1</el>
        <el>1</el>
        <el>1</el>
        <el>1</el>
        <el>1</el>
        ...
        <el>-2800</el>
        <el>-5200</el>
        <el>-5800</el>
    </value>
</linearConstraintCoefficients>
```

Data Structures

Translating from AMPL *(cont'd)*

OSiL derived from AMPL's output format

```
<nonlinearExpressions numberOfNonlinearExpressions="1">
    <nl idx="-1">
        <sum>
            <divide>
                <times>
                    <number value="39" type="real"/>
                    <variable idx="0" coef="1"/>
                </times>
                <minus>
                    <number value="1" type="real"/>
                    <divide>
                        <variable idx="0" coef="1"/>
                        <number value="1300" type="real"/>
                    </divide>
                </minus>
            </divide>
            ...
        </sum>
    </nl>
</nonlinearExpressions>
```

Data Structures

Translating from AMPL *(cont'd)*

OSrL derived from solver's results

```
<osrl xmlns:os="os.optimizationservices.org"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="os.optimizationservices.org
    ../schemas/OSiL.xsd">

    <resultHeader>
        <generalStatus type="success"/>
        <serviceName>Solved using a LINDO service</serviceName>
    </resultHeader>

    <resultData>
        <optimization numberOfSolutions="1" numberOfVariables="24"
            numberOfConstraints="10" numberOfObjectives="1">
            ...
        </optimization>
    </resultData>

</osrl>
```

Data Structures

Translating from AMPL *(cont'd)*

OSrL derived from solver's results

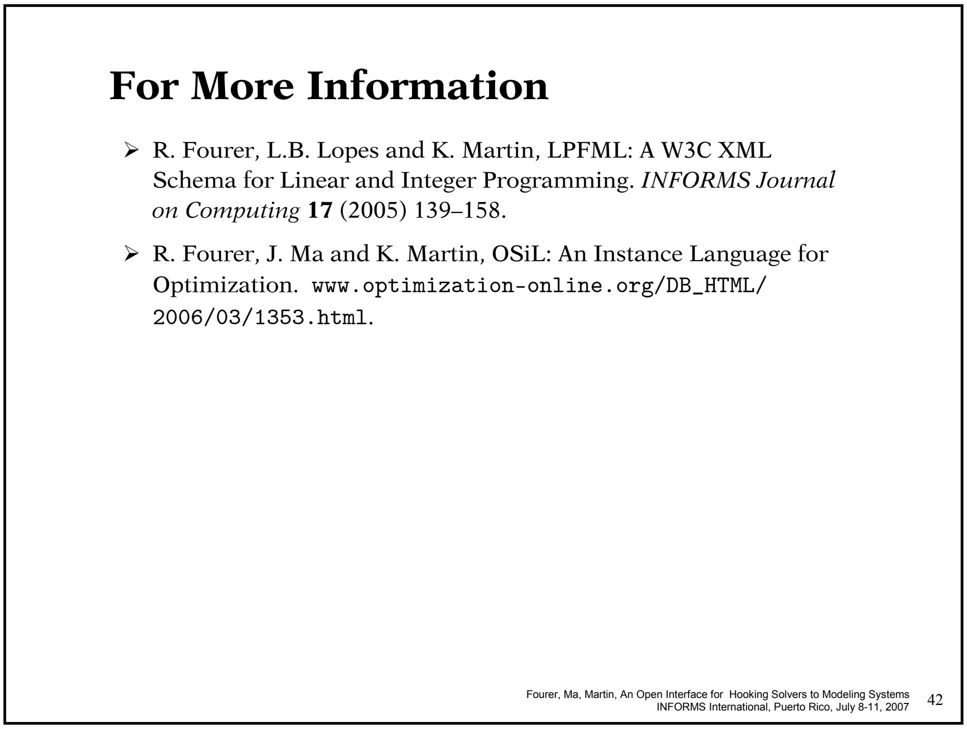
```
<solution objectiveIdx="-1">
    <status type="optimal"/>
    <variables>
        <values>
            <var idx="0">36.8552</var>
            <var idx="1">563.142</var>
            <var idx="2">122.355</var>
            <var idx="3">0</var>
            <var idx="4">991.065</var>
            ...
        </values>
        <other name="reduced costs">
            <var idx="0">0</var>
            <var idx="1">0</var>
            <var idx="2">0</var>
            <var idx="3">8.5573</var>
            <var idx="4">-2.51902e-09</var>
            ...
        </other>
    </variables>
```

Data Structures

Translating from AMPL *(cont'd)*

OSrL derived from solver's results

```
    <objectives>
        <values>
            <obj idx="-1">722383</obj>
        </values>
    </objectives>
    <constraints>
        <dualValues>
            <con idx="0">-12.4722</con>
            <con idx="1">-98.9784</con>
            <con idx="2">0</con>
            <con idx="3">53.7812</con>
            <con idx="4">35.7967</con>
            <con idx="5">25.5129</con>
            <con idx="6">17.9149</con>
            <con idx="7">82.3857</con>
            <con idx="8">193.978</con>
            <con idx="9">29.3393</con>
        </dualValues>
    </constraints>
</solution>
```



For More Information

- R. Fourer, L.B. Lopes and K. Martin, LPFML: A W3C XML Schema for Linear and Integer Programming. *INFORMS Journal on Computing* **17** (2005) 139–158.
- R. Fourer, J. Ma and K. Martin, OSiL: An Instance Language for Optimization. `www.optimization-online.org/DB_HTML/2006/03/1353.html`.