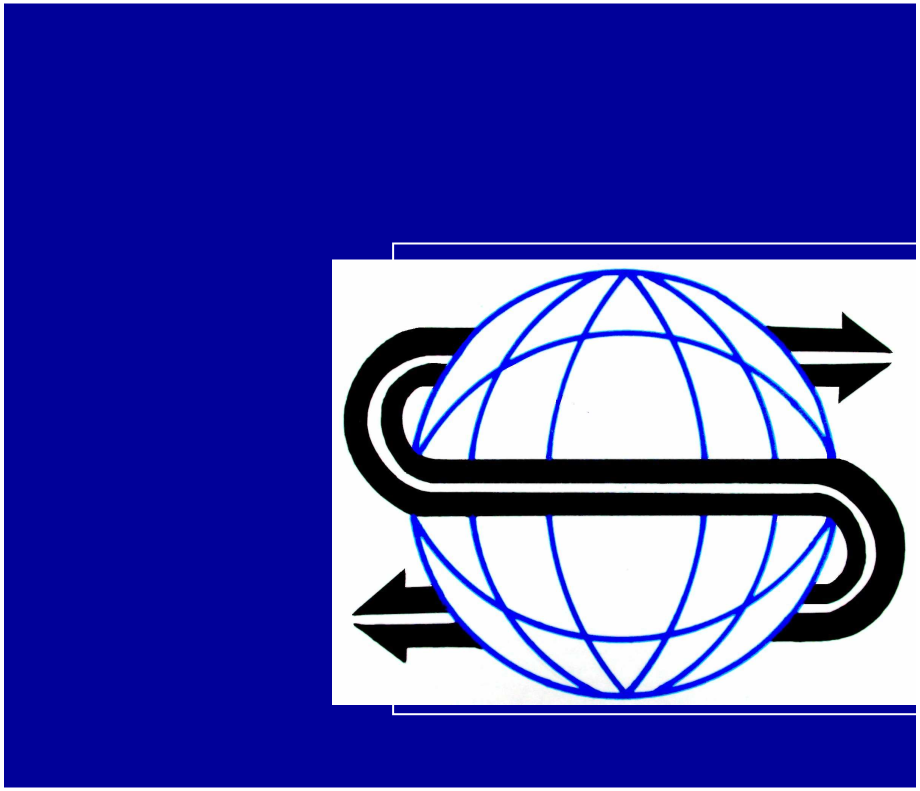


Optimization Services (OS)

-- The Internet for OR

-- The Next Generation NEOS (Funded by NSF)

-- An Open Source Computational Infrastructure

Robert Fourer
Jun Ma
Northwestern University
Kipp Martin
University of Chicago

Jun Ma

Annapolis,
01/07/2005

OUTLINE

1. Motivations

2. Introduction

3. Optimization Services and OSxL

4. An OSxL Example -- Optimization Services instance Language (OSiL)

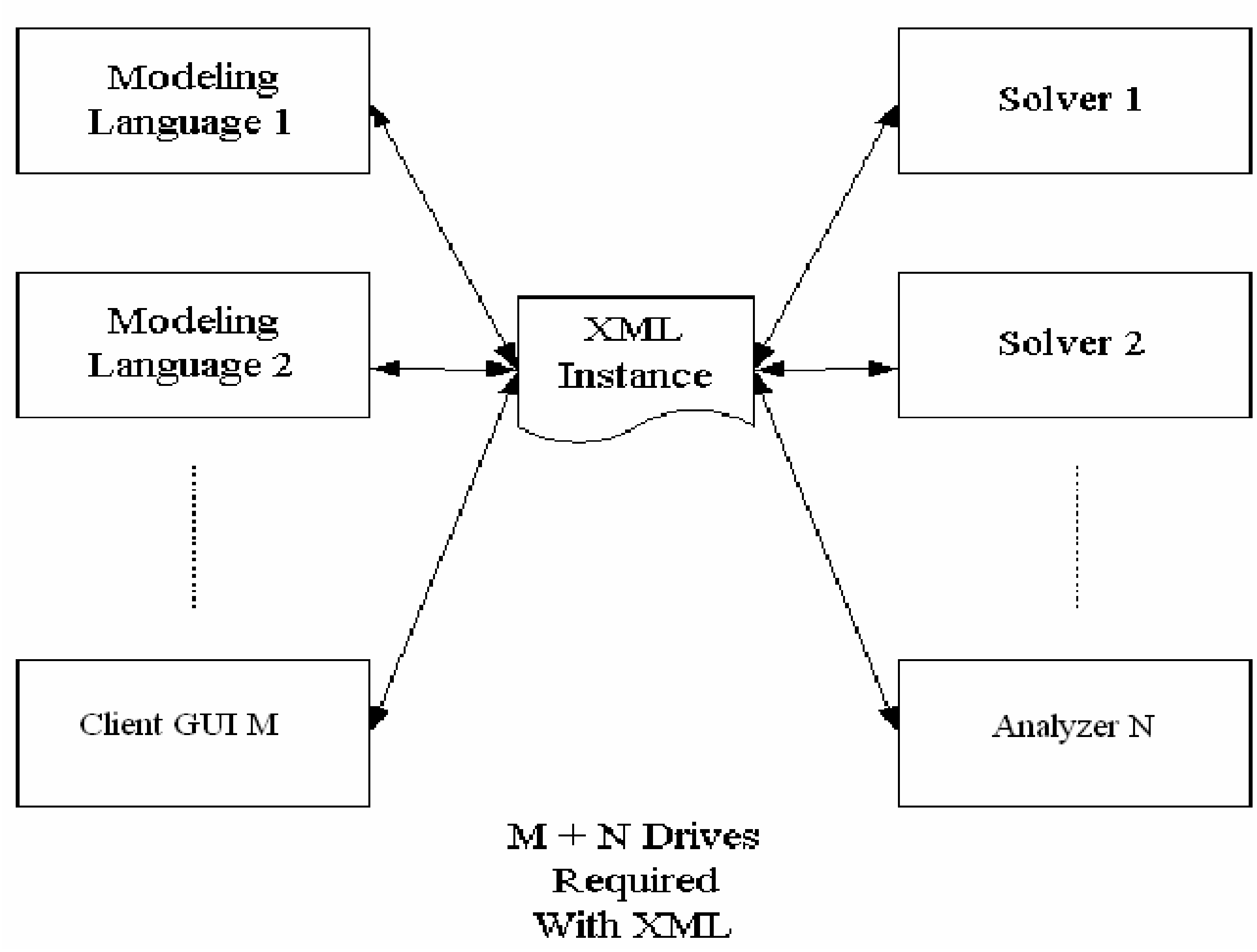
5. Conclusion

Motivation

- An Open, Scalable and Standard Environment that Facilitates Development & Use of OR Software and Promotes Collaboration and Other Related Researches
- Convenience And Power
  - Just like Using Utility Services (therefore the name – Optimization Services)
  - Knowledge in Optimization Algorithms and Software (solvers, options, etc.) Required of Users Should be As Little As Possible
  - Better and More Choices of Modeling Languages and Solver
  - More Types of Optimization Services (Analyzers/Preprocessors, Problem Providers, Bench Markers, Registry, Simulation etc.)
  - Solve More Types of Problems
- Distributed and Decentralized Environment
  - Automatic Optimization Services Discovery
  - Optimization Services Development and Registration

Motivation

For example, it would be nice to have an instance representation language. This is specified by the Optimization Services instance Language (OSiL)



Modeling Language 1

Modeling Language 2

Client GUI M

XML Instance

Solver 1

Solver 2

Analyzer N

M + N Drives Required With XML

Introduction

- Optimization Services is

  A framework, NOT a system

  - cf. constitution, NOT government/Court System. Only that the framework specifications are written in XML languages (NOT English).
  - But we are in the middle of developing the modeling system according to this framework.
  - We are also building libraries for other people to put up their optimization services.
- Distributed environment (Local environment being just a special Case)
- Service Oriented, Optimization Centered, Decentralized Architecture.

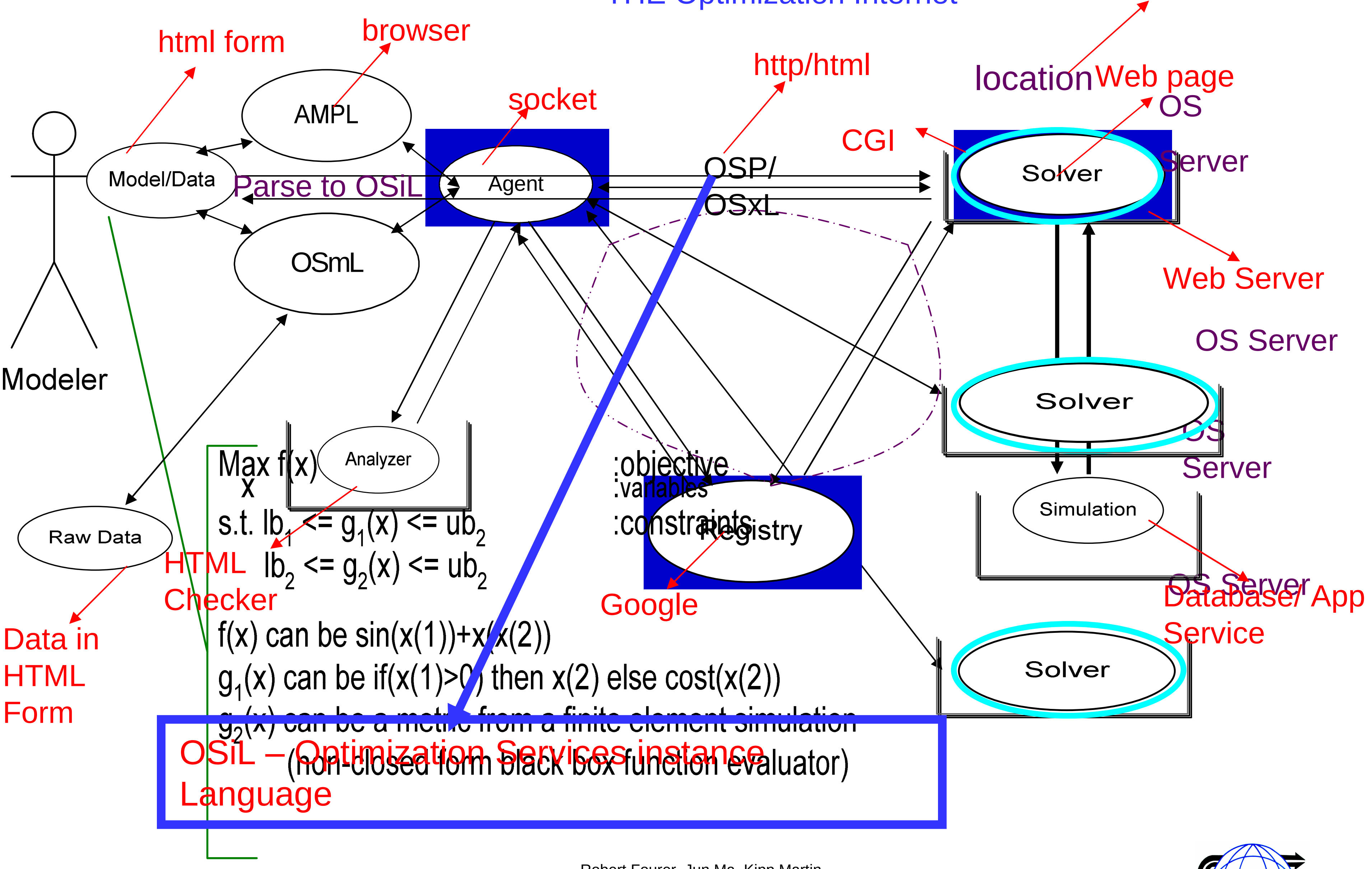
Introduction

- Optimization Services Components
  1. Modeling Language Environment (MLE) (e.g. AMPL, OSmL) -- OSModeler
  2. Optimization Registries (e.g. The next generation NEOS) – OSRegistry
  3. Analyzers/Preprocessors (e.g. Mprobe, Dr. AMPL) -- OSAnalyzer
  4. Optimization Solvers (e.g. Lindo) -- OSSolver
  5. Simulation (e.g. Finite Element Analysis) -- OSSimulation
  6. Communication Software Agent – OSAgent
  7. All of the above are communicating in a common language -- OSCommon

XML-based standard

Optimization Services (OS)

THE Optimization Internet



Max f(x)
x

s.t. lb_1 <= $g_1(x)$ <= ub_2

lb_2 <= $g_2(x)$ <= ub_2

f(x) can be sin(x(1))+x(x(2))

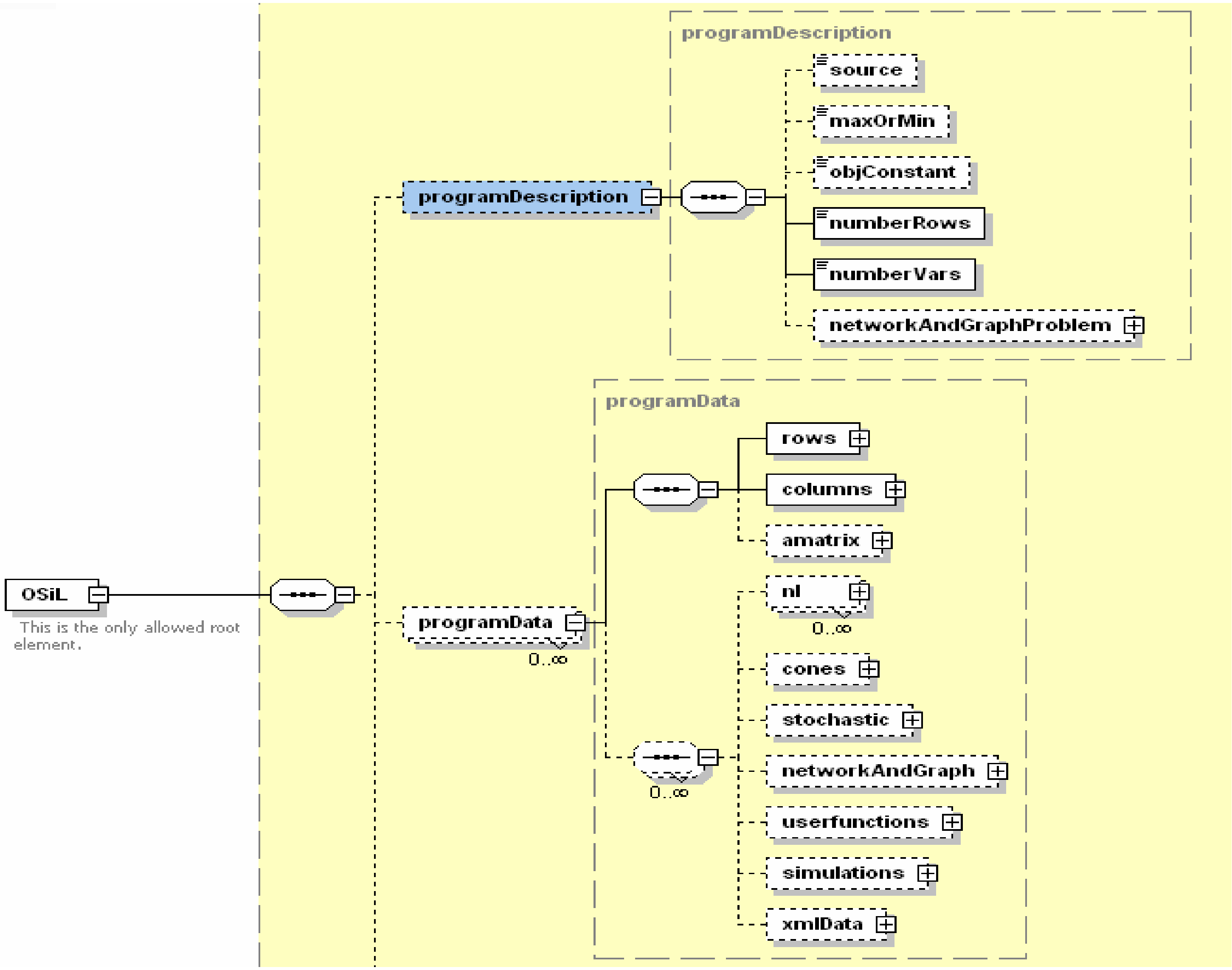
$g_1(x)$ can be if(x(1)>0) then x(2) else cost(x(2))

$g_2(x)$ can be a metric from a finite element simulation

(non-closed form black box function evaluator)

Robert Fourer, Jun Ma, Kipp Martin

Optimization Services instance Language (OSiL) Schema



OSiL Schema

- Nonlinear Expressions and OSnL Schema
  - 220 Major Nodes (Operators/Operands)
  - Arithmetic Operators, Elementary Functions, Statistical and Probability Functions, Constants, Operands, Logic and Relational Operator, Trigonometric Function, Special Elements
  - User Defined Functions
  - Simulations
  - XMLData and xPath Elements
  - Quadratic Programming Nodes
- OS API (OSiLReader/OSiLWriter) and OS Expression Tree
- Connecting to Solvers
- All Major Optimization Types Supported

OSiL Schema

-Linear
-Mixed integer
-Bound constrained optimization
-General quadratic optimization
-Nonlinear unconstrained/constrained
-General mixed integer nonlinear
-General nonlinear with user-defined functions
-Global optimization
-General nonlinear with simulations (black-box functions)
-Optimization over simulation/nondifferentiable optimization
-General nonlinear with xml data (either locally within the OSiL or remotely located)
-General nonlinear with data look up (XPath)
-Network and graph definition
-Network programming
-Constraint/logic programming
-Semidefinite programming
-Semi-infinite programming
-Cone programming
-Complementarity problem
-Stochastic linear/nonlinear (distribution problem, distribution based recourse problem, scenario based recourse problem, chance constrained)
-Combinatorial optimization/Heuristic Optimization (TSP, MST, SP, MF, MCF, VRP, Set Covering, Coloring etc. etc.)

Conclusion

- Sufficient Motivation for Optimization Services
- Optimization Services as the Internet for OR
  - Simple
  - Scalable
  - Standard
  - Smooth
- An OSxL Example – Optimization Services instance Language
  - Cleanly Designed from Scratch
  - Highly Extendable
  - State-of-art Expression Tree Design
  - Supports All Major Optimization Types
  - Built for Distributed and Decentralized Systems
  - Comes with Natively Designed OSiL APIs (OSiLReader/Writer)
  - Already Connected with Solvers