Letting Your Application do the Modeling

(Model Edit Functions)

Chris Kuip

AIMMS and Optimization Specialist

Overview of Demos

Use in model development

1. Hello World: Scalar Reference (theory)
2. More and more Functionality (application)

Use in working with formulas

1. Formulas as Data (theory)
2. Blending on Specification (application)

Textbook example: Transportation problem

Indices:

f	*factories*
c	*customers*

Parameters:

S_f	*supply at factory* f
D_c	*demand by customer* c
T_{fc}	*unit transport cost between* f *and* c

Nodes:

FN_f	*factory supply node for* f
CN_c	*customer demand node for* c

Arcs:

$Flow_{fc}$	*transport between* f *and* c

- "Abstract" and "Concrete" are both used here.
- Abstract: the number of nodes;
  - applicability much wider
- Concrete: we are concrete about what the indices and parameters mean; makes reasoning easy.

Model development

- Real world applications:
  - Several sets
  - Dozens of variables and constraints
  - Several times more parameters
- We want to reason about groups of such identifiers
  - For example to, add a new parameter, and use this in several procedures,
  - For example to, add information about a new product to all of these identifiers

Hello World Demo: Scalar Reference

ME Operations:

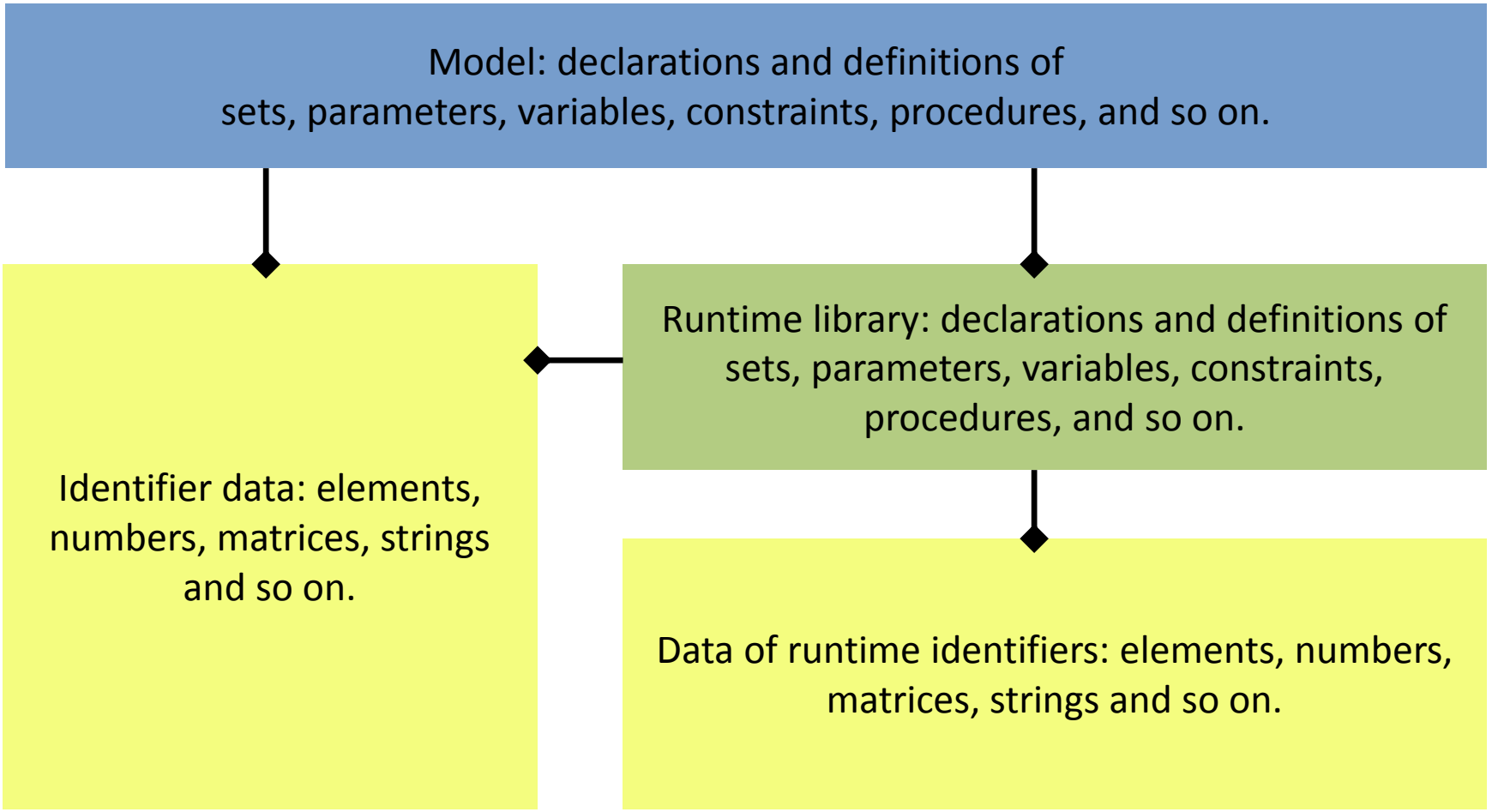
1. Creation: `me::CreateLibrary, me::Create`
2. Modification: `me::SetAttribute`
3. Compilation: `me::Compile`
4. Execution: `APPLY ep_Proc`

Traditional Framework: Basic Model – Data separation

Model: declarations and definitions of
sets, parameters, variables, constraints, procedures, and so on.

Identifier data: elements, numbers, matrices, strings and so on.

Model Edit Framework: Next Level Model – Data separation



Taking more data into account

With a set of products, index pr, a lot of data is available, and we want to establish a new product, say ‘pr2’, similar to ‘pr1’. How can we ensure, that, initially, all relevant data is copied?

- Start with a model and some input data
  - `P1(pr), P2(mat, pr), …`
- Add product pr2 similar to pr1
  - `P1('pr2') := P1('pr1');`
  - `P2(mat,'pr2') := P2(mat,'pr1');`
- Take into account other product data, say
  - `P5(pr, reg)`

Demo 2 “More and more functionality”

ME Support used:

1. Element Parameters, `AllIdentifiers`
2. String Parameters, string manipulation
3. Model Query Functions

`DomainIndex, IndexRange, IdentifierDimension`

`AttributeToString / me::GetAttribute`

Part 2: Working with formulas

MACRO

```
MACRO loglog {
      arguments  : x
      definition : if ( x > 10 )
                   then log( log( x ) )
                   else 0 endif
}
Body:
     P := loglog( q );
Constraint:
     loglog(z) = sum(i, 2 * loglog( x(i) )
)
```

Demo “Formulas as Data”

- Abstractly working with formulas, not just numbers.
- We will be:
  - Entering a single formula,
  - use that formula in an assignment, and
  - in a trivial model.

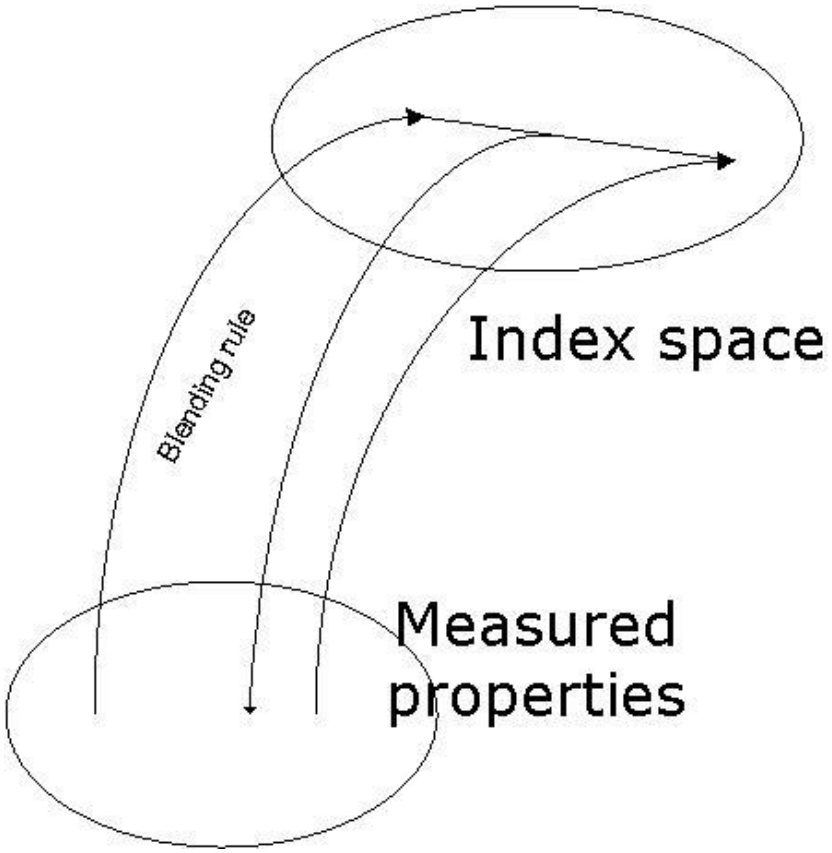
Purpose of formulas as data

At least two application areas:

- Blending on Specification
  The intellectual property of the blending rules involved is property of the model user, not of the model developer
- Asset management
  The valuation of a portfolio is intellectual property of the model user, not of the model developer.

Demo 4 “Blending on Specification”

- Some properties blend linear in measured values (for instance density blends linear on volume)
- Other properties blend linear on the so-called index of the property
- Picture: blending two equal amounts
- Function “Blending rule” is injective
- Inverse can be explicit, but need not.



Demos: First with explicit mon rule, then with MEF

Some final remarks:

Word of caution, MEF not suited for:

- Replacement of model explorer
- Generating individual columns / rows during solution process: use GMP instead.

Required: Error handling, covered by previous webinar.

But does offer, concrete:

- Selections of identifiers become an object that can be operated upon.
- Programmatic control over the tools in AIMMS:
  - model tree,
  - compiler, and
  - execution system.

But does offer, abstract; it allows applications to be more:

- Flexible
- Generic

Further reading

AIMMS Tech blog: techblog.aimms.com

- http://techblog.aimms.com/2011/11/getting-value-of-a-dynamic-identifier/
- http://techblog.aimms.com/2014/05/repetitive-patterns-captured-by-model-query-and-model-edit-functions/
- http://techblog.aimms.com/2014/08/21/formulas-as-data/

AIMMS The Language Reference:

- Section: “Working with the set AllIdentifiers”
- Section: “MACRO declaration and attributes”
- Section: “Runtime Libraries and the Model Edit Functions”

AIMMS The Function Reference:

- Chapter: Model Query Functions
- Chapter: Model Edit Functions