IFC4 Parametric Modeling

Tim Chipman

buildingSmart Model Support Group

What are parametric extensions?

Capability to set an object attribute equal to a formula

Formulas may reference:

- Constants (with or without specific units)
- References to other object attributes
- Arithmetic Operations (Add, Subtract, Multiply, Divide)
- Lookups from tables (e.g. product configurations)

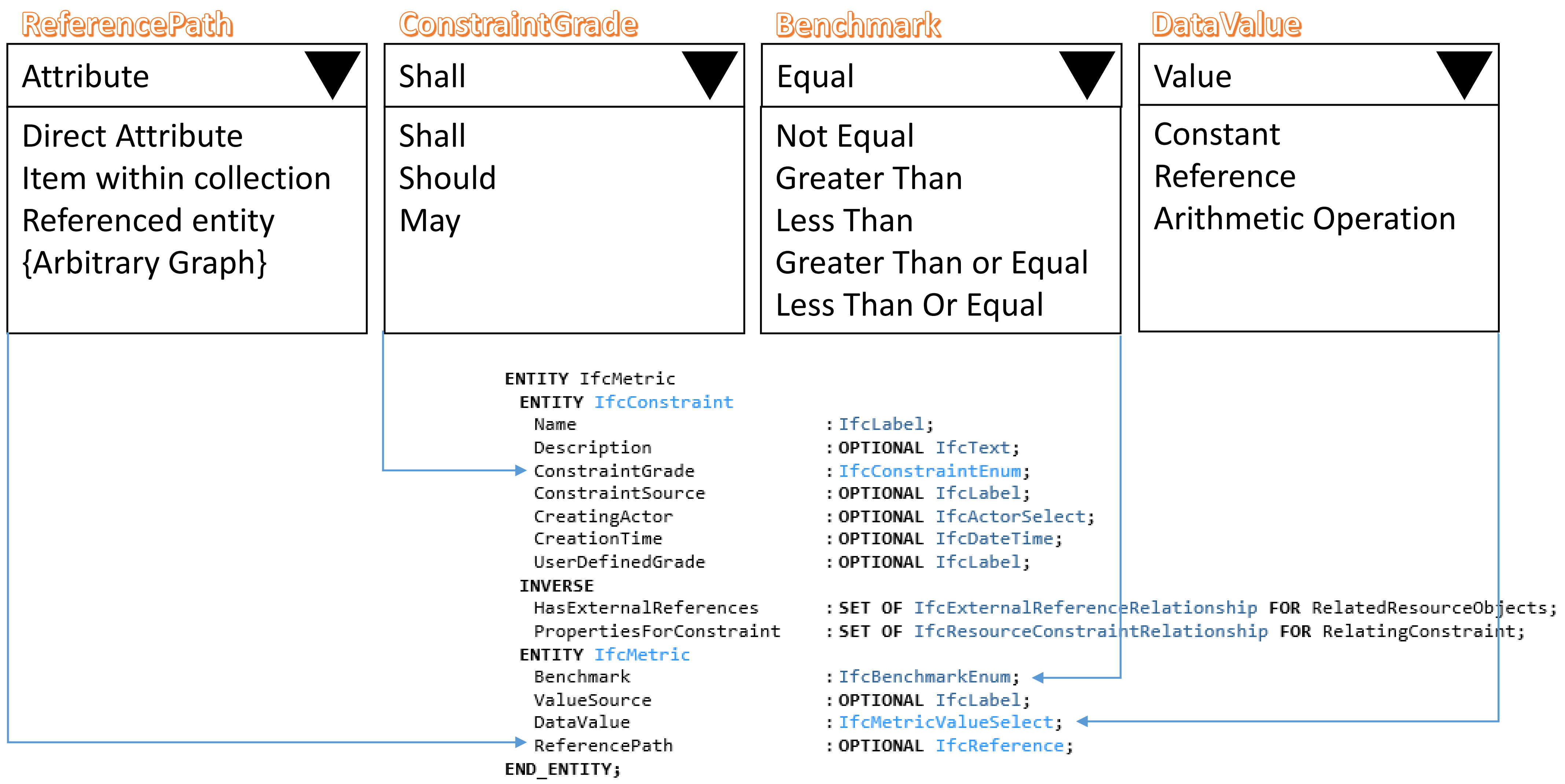
How are parametrics defined?

Formula is defined just like any other constraint

IfcObjective.ObjectiveQualifier=PARAMETER tells applications to automatically adjust values to conform to constraint

- IfcRelAssociatesConstraint links objects
- IfcObjective qualifies constraint as parametric
- IfcMetric identifies attribute and defines formula expression

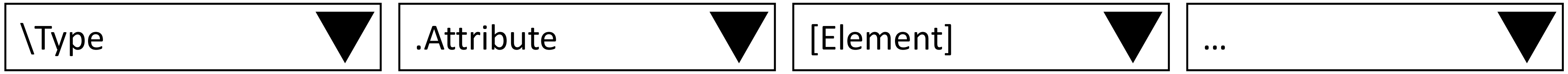
IfcMetric: fundamental definition



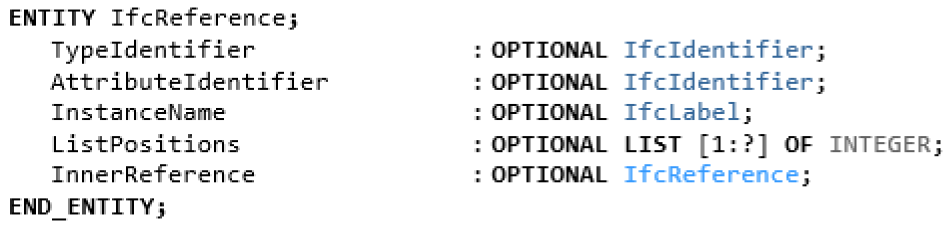
ReferencePath	ConstraintGrade	Benchmark	DataValue
Attribute ▼	Shall ▼	Equal ▼	Value ▼
Direct Attribute	Shall	Not Equal	Constant
Item within collection	Should	Greater Than	Reference
Referenced entity	May	Less Than	Arithmetic Operation
{Arbitrary Graph}		Greater Than or Equal	
		Less Than Or Equal	

```
ENTITY IfcMetric
 ENTITY IfcConstraint
  Name                       : IfcLabel;
  Description                : OPTIONAL IfcText;
  ConstraintGrade            : IfcConstraintEnum;
  ConstraintSource           : OPTIONAL IfcLabel;
  CreatingActor              : OPTIONAL IfcActorSelect;
  CreationTime               : OPTIONAL IfcDateTime;
  UserDefinedGrade           : OPTIONAL IfcLabel;
 INVERSE
  HasExternalReferences      : SET OF IfcExternalReferenceRelationship FOR RelatedResourceObjects;
  PropertiesForConstraint    : SET OF IfcResourceConstraintRelationship FOR RelatingConstraint;
 ENTITY IfcMetric
  Benchmark                  : IfcBenchmarkEnum;
  ValueSource                : OPTIONAL IfcLabel;
  DataValue                  : IfcMetricValueSelect;
  ReferencePath              : OPTIONAL IfcReference;
END_ENTITY;
```

IfcReference: identify the attribute



- Identity the data type, e.g. “IfcWindow”
- Identify the attribute within the data type, e.g. “Width”
- Identify an object within a collection according to its name
- Identify an object within a collection according to its position
- For object reference, link to an inner reference (forming a graph)



```
ENTITY IfcReference;
   TypeIdentifier          : OPTIONAL IfcIdentifier;
   AttributeIdentifier     : OPTIONAL IfcIdentifier;
   InstanceName            : OPTIONAL IfcLabel;
   ListPositions           : OPTIONAL LIST [1:?] OF INTEGER;
   InnerReference          : OPTIONAL IfcReference;
END_ENTITY;
```

IfcReference: Example

\Type ▼	.Attribute ▼	[Element] ▼	-> Inner Reference

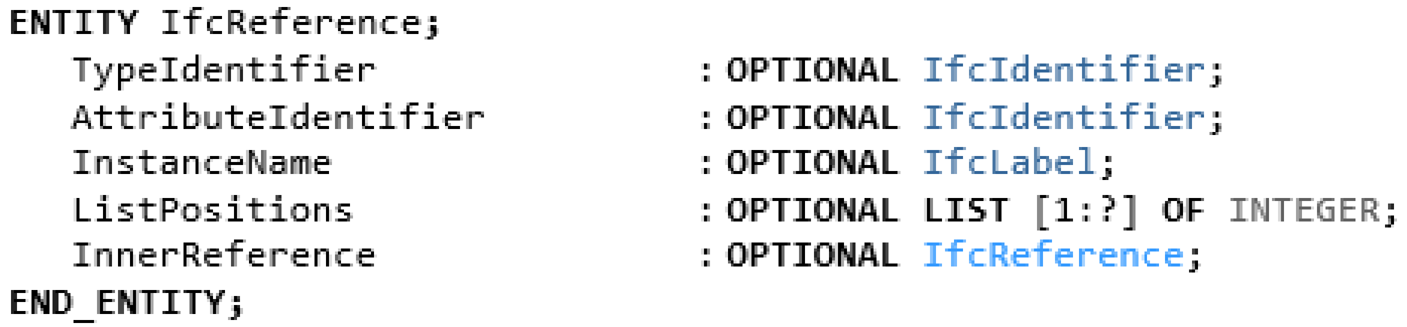
#1=IfcReference(“IfcFurnitureType”,”HasRepresentation”, “Body”,$,#2);

#2=IfcReference(“IfcRepresentationMap”,”MappedRepresentation”,$,$,#3);

#3=IfcReference(“IfcShapeRepresentation”,”Items”,$,(1),#4);

#4=IfcReference(“IfcBlock”,”XLength”,$,$,#5);

#5=IfcReference(“IfcPositiveLengthMeasure”,$,$,$,$);

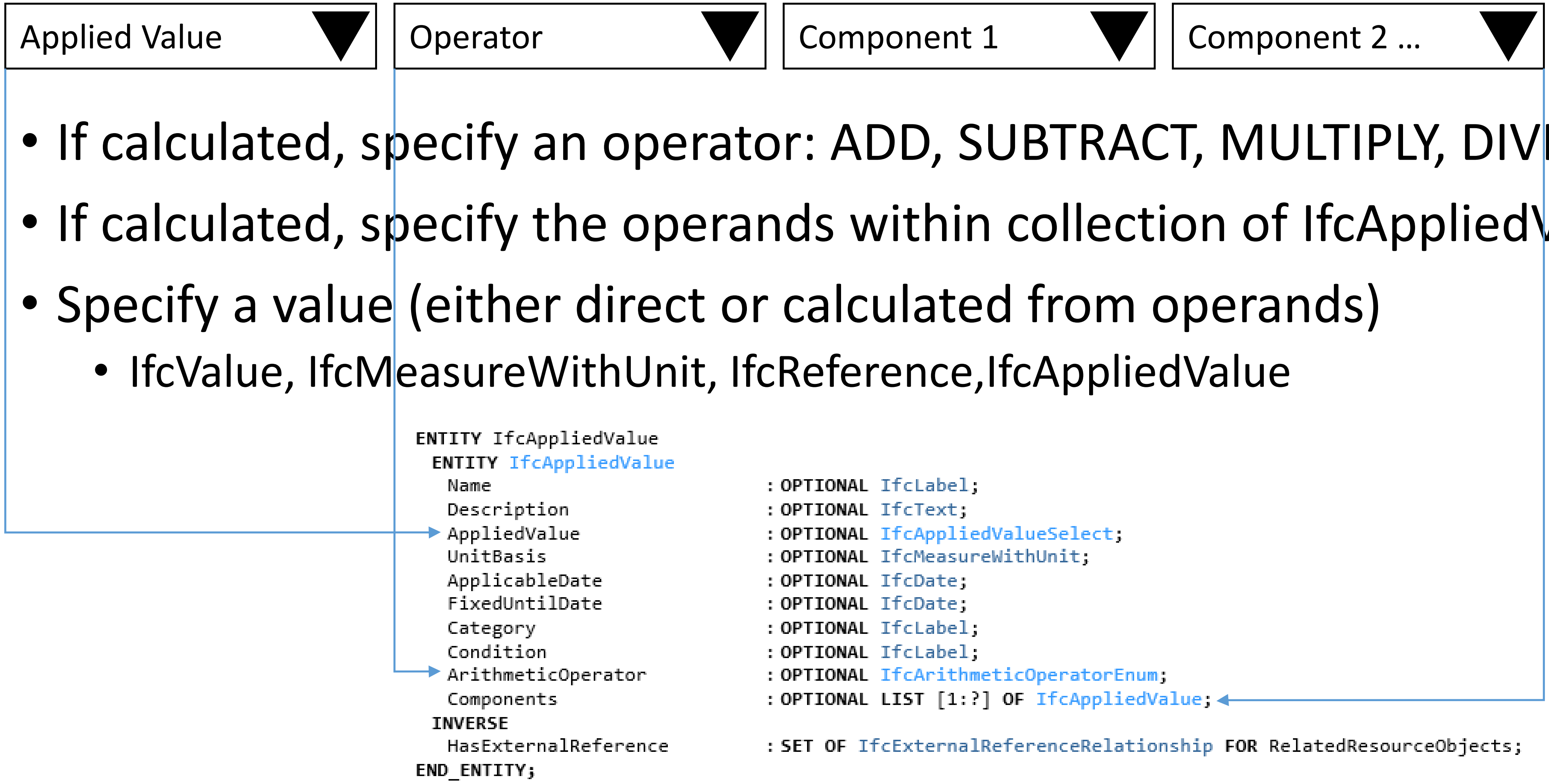


```
ENTITY IfcReference;
   TypeIdentifier                  : OPTIONAL IfcIdentifier;
   AttributeIdentifier             : OPTIONAL IfcIdentifier;
   InstanceName                    : OPTIONAL IfcLabel;
   ListPositions                   : OPTIONAL LIST [1:?] OF INTEGER;
   InnerReference                  : OPTIONAL IfcReference;
END_ENTITY;
```

IfcMetricValueSelect: the expression

- IfcValue: a constant value using project default units
- IfcMeasureWithUnit: a constant value using specified units
- IfcAppliedValue: a formula expression
- IfcTable: a lookup table

IfcAppliedValue: a formula expression



- If calculated, specify an operator: ADD, SUBTRACT, MULTIPLY, DIVIDE
- If calculated, specify the operands within collection of IfcAppliedValue
- Specify a value (either direct or calculated from operands)
  - IfcValue, IfcMeasureWithUnit, IfcReference,IfcAppliedValue

```
ENTITY IfcAppliedValue
  ENTITY IfcAppliedValue
    Name                       : OPTIONAL IfcLabel;
    Description                : OPTIONAL IfcText;
    AppliedValue               : OPTIONAL IfcAppliedValueSelect;
    UnitBasis                  : OPTIONAL IfcMeasureWithUnit;
    ApplicableDate             : OPTIONAL IfcDate;
    FixedUntilDate             : OPTIONAL IfcDate;
    Category                   : OPTIONAL IfcLabel;
    Condition                  : OPTIONAL IfcLabel;
    ArithmeticOperator         : OPTIONAL IfcArithmeticOperatorEnum;
    Components                 : OPTIONAL LIST [1:?] OF IfcAppliedValue;
  INVERSE
    HasExternalReference       : SET OF IfcExternalReferenceRelationship FOR RelatedResourceObjects;
END_ENTITY;
```

IfcTable

- Each column maps to a referenced attribute on object.
- Each row indicates a valid combination of values
- Exactly one of the rows must be satisfied
- A table-constrained value may be changed on an object only if another row exists with the proposed value
- If there is a single row with the proposed value, then that shall be selected, otherwise if there is another row with all other values the same, then that shall be selected, otherwise change not allowed

IfcTableColumn

- Identifier: allows reference to column within another table
- Name: human-readable heading
- Description: human-readable description of column
- Reference: identifies attribute to be set when row is selected
- Unit: Optional unit (if different than project units)

```
ENTITY IfcTableColumn;
   Identifier                 : OPTIONAL IfcIdentifier;
   Name                       : OPTIONAL IfcLabel;
   Description                : OPTIONAL IfcText;
   Unit                       : OPTIONAL IfcUnit;
   ReferencePath              : OPTIONAL IfcReference;
END_ENTITY;
```

Parametric Geometry

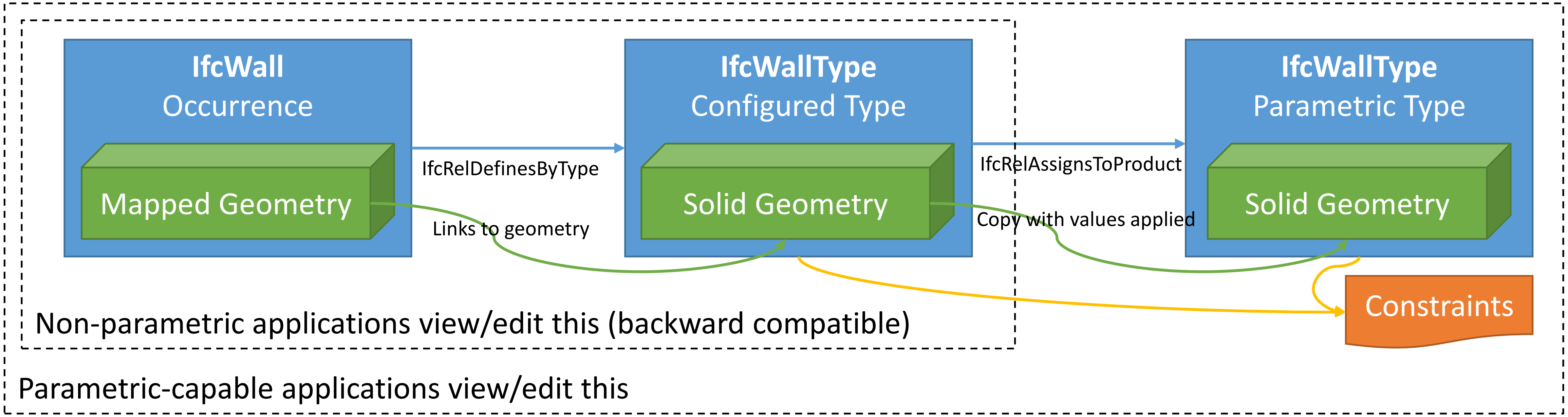
- Items may be positioned relative to:
  - 1D bounded curve (“Axis” representation), e.g. beams, studs, walls
  - 2D bounded area (“FootPrint” representation), e.g. flooring, wall covering
  - 3D bounded box (“BoundingBox” representation), e.g. cabinetry, windows
- IFC defines “standard” parametric geometry: IfcBeamStandardCase, IfcSlabStandardCase, etc. where geometry and formulas are implicit
- For custom parametric geometry, geometry and formulas are explicit
- Bounding representations may have constraints as well

Where are parametrics used?

- Anywhere!
- However, product types are the most interesting scenario

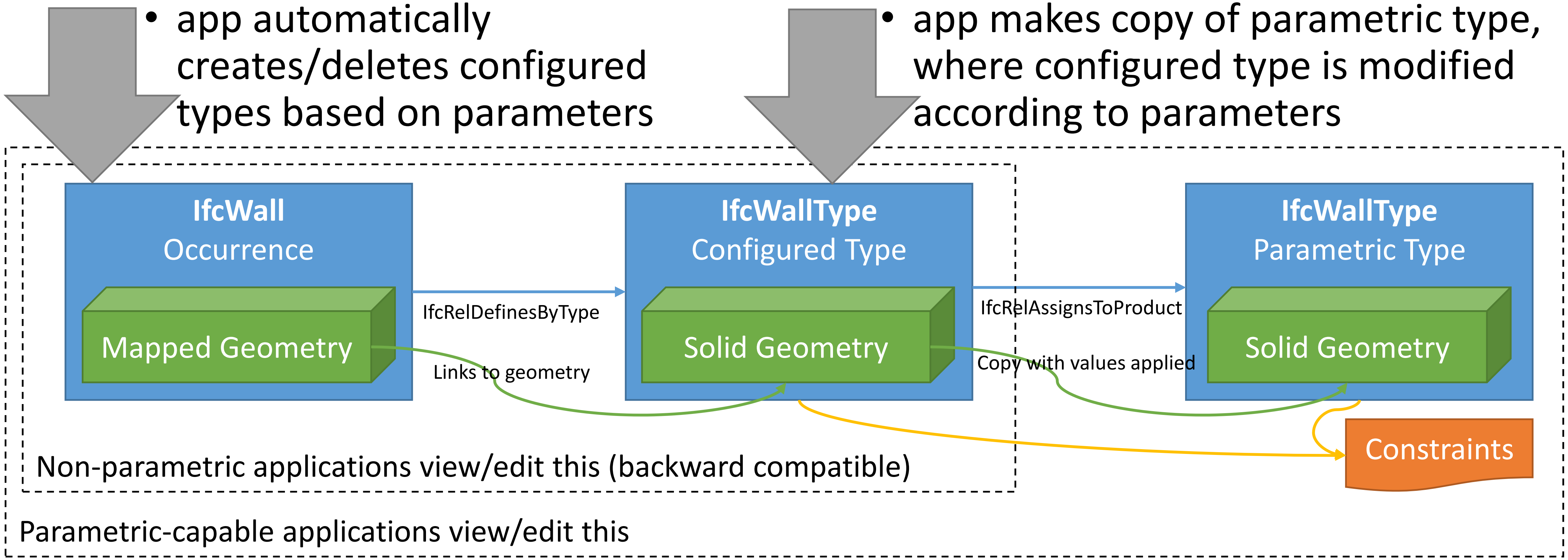
Implementation Requirements

- Must be backward compatible and scalable
- Intermediate “configured type” for compatibility and efficient re-use
- One parametric type may have ~10 configurations, ~1000 occurrences



Application Scenarios

- Define an occurrence
  - app automatically creates/deletes configured types based on parameters
- Define a configured type
  - app makes copy of parametric type, where configured type is modified according to parameters



User-configurable parameters

- How to define a parameter that is user-configurable?
- IfcMetric.Name provides human-readable name of setting
- Current “value” must be determined by evaluating constraint
  - For a table, determine the current row corresponding to values
  - Use cell of first column for display

Part Composition

- Parametric types may be aggregated into parametric parts
- Constraints may be “rolled up” by nature of using same identifiers
- If selecting a value from Table A, if a table column is linked to a column within another table, then cascade the value change to that table recursively
- Example: Cabinet has a style defined (White | Wood-grain), each aggregated Cabinet Frame then uses same style

Part Composition Pattern Placement

- Multiple occurrences may be dynamically defined using IfcMappedItem as documented in IFC4 (e.g. already done for rebar)
- Set IfcMetric.ReferencePath to a collection count:

  \IfcMember.Representation
  \IfcProductRepresentation.Representation[“Body”]
  \IfcShapeRepresentation.Items[]
  \IfcMappedItem

- Placement may be defined relative to order within collection

  \IfcMember.Representation
  \IfcProductRepresentation.Representation[“Body”]
  \IfcShapeRepresentation.Items[*]