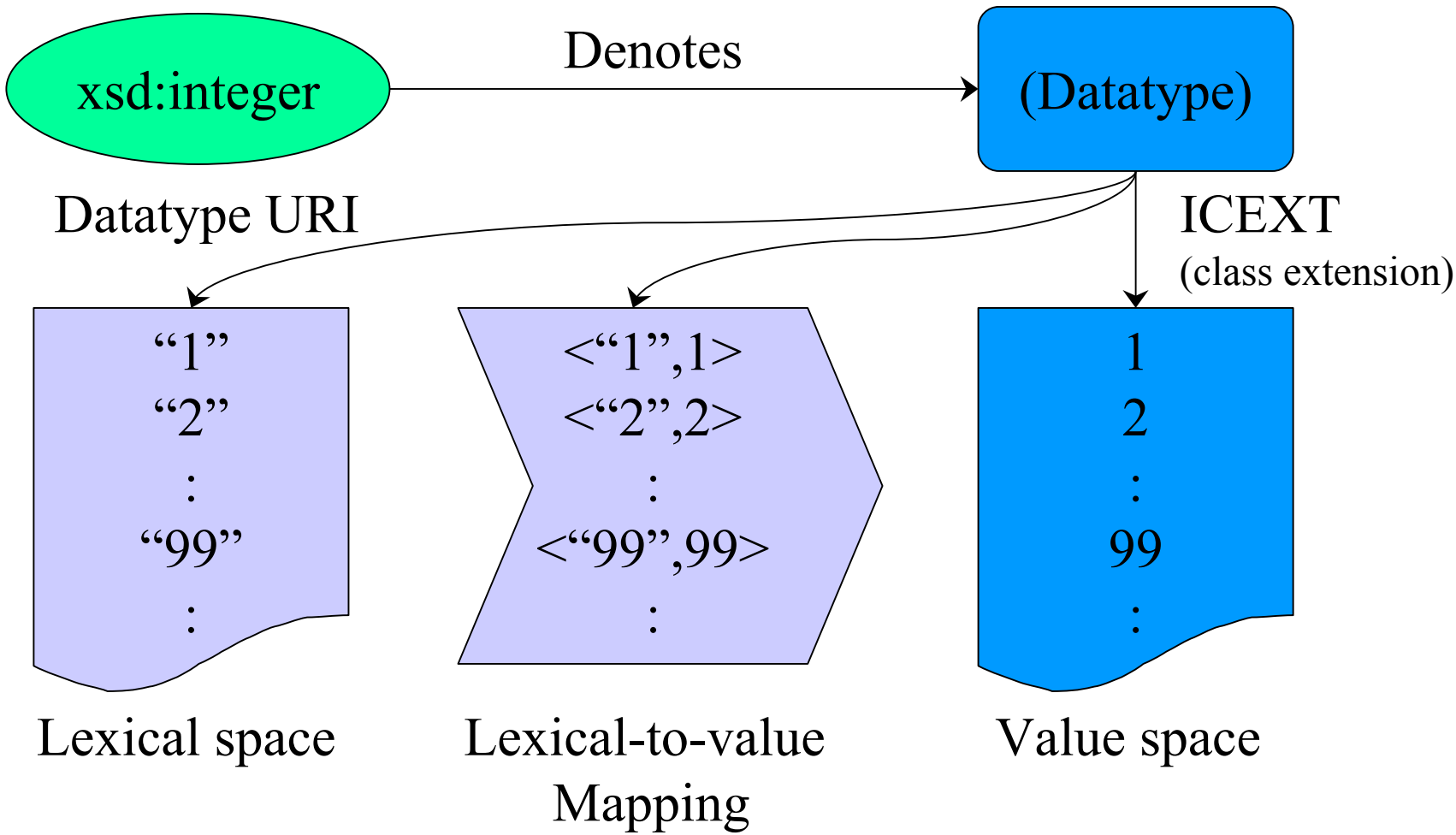
Using RDF Datatypes
(Deduction beyond syllogism)

Graham Klyne

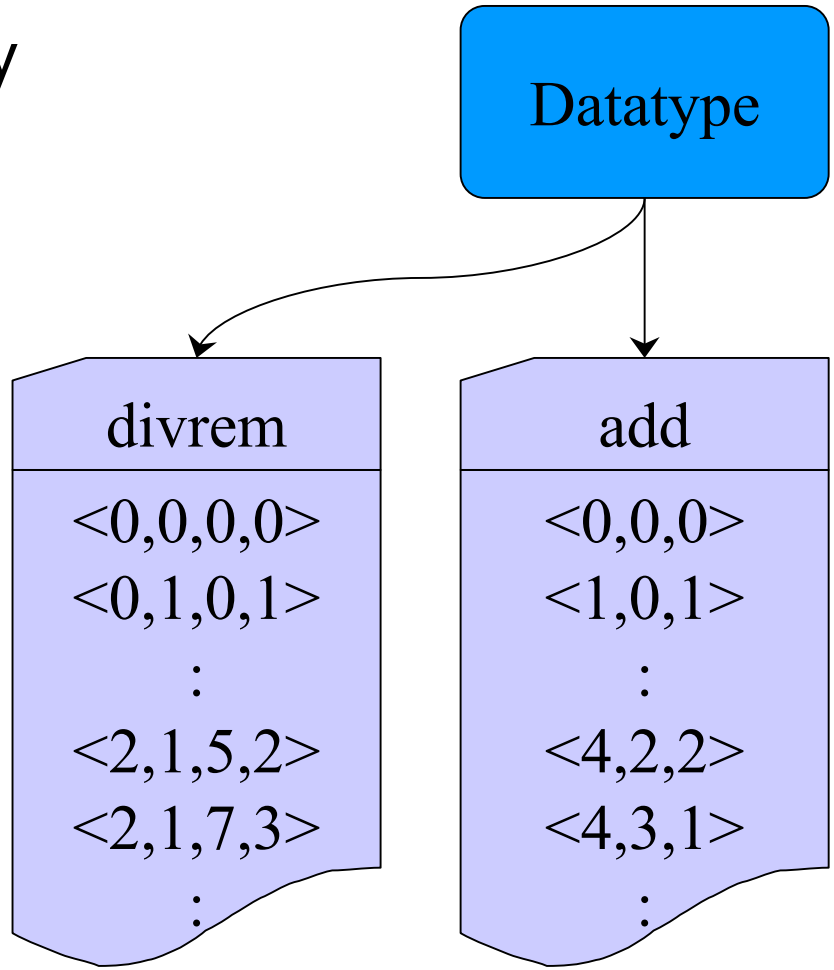
http://www.ninebynine.org/

RDF Datatypes (recap)



Sketch of Datatype Extension

- A number of named *n*-ary relations
  - e.g. xsd_integer:add:
    { <a,b,c> | a=b+c }
  - e.g. xsd_integer:divrem:
    { <a,b,c,d> |
      a=c div d,
      b=c rem d }
- Functions to compute relations
  - e.g. for xsd_integer:add:
    <4,?,2> → <4,2,2>



Using RDF Datatypes

- Two approaches, implemented in Swish
  - Swish is a framework, implemented in Haskell, for experimenting with RDF inference, and other stuff
  - http://www.ninebynine.org/RDFNotes/Swish/Intro.html
- Others are possible
  - e.g. CWM / Euler
  - Survey:
    http://www.ninebynine.org/RDFNotes/
    RDF-Datatype-inference.html

Motivating Example

- Class of passenger vehicles
- Properties for vehicle passenger capacity:
  - Seated
  - Standing
  - Total
  - where: Total = Seated + Standing

Some Criteria to Consider

- Ease of definition?
- Use RDF syntax?
- Direction of inference
  - "Forward" vs "backward" deduction
  - Variation of available information
    - ?=2+3 → <5,2,3>
    - 5=?+3 → <5,2,3>
    - 5=2+? → <5,2,3>
- Separate domain knowledge from generic
- Formal model for RDF datatypes?

1st Approach:
Rule with Variable Binding Modifier

```
ex:Rule01Ant :-
  { ?pv a :PassengerVehicle ;
      :seatedCapacity   ?c1 ;
      :standingCapacity ?c2 . }

ex:Rule01Con :-
  { ?pv :totalCapacity ?ct . }

@rule ex:Rule1 :-
  ( ex:Rule01Ant ) => ex:Rule01Con
  | ( xsd_integer:sum ?ct ?c1 ?c2 )
```

2nd Approach:
Generalized Class Restriction

```
ex:VehicleRule2 :-
  { :PassengerVehicle
      a rdfd:GeneralRestriction ;
      rdfd:onProperties
        (:totalCapacity
         :seatedCapacity
         :standingCapacity) ;
      rdfd:constraint xsd_integer:sum ;
      rdfd:maxCardinality
        "1"^^xsd:nonNegativeInteger . }
```

Observations

- Both approaches:
  - support forward and some backward chaining
  - based on similar extension of RDF datatype
- Variable binding modifier:
  - is easier for writing more complex rules
  - can combine query and calculation in a single rule
  - requires fewer intermediate steps (transient subgraphs)
- General restriction:
  - can be implemented with standard RDF parser
  - handles backward chaining more flexibly (but slowly)
  - does not handle query/selection of RDF data

Datatype Extension Comments

- Formalism of adding named relations seems to have some practical value as a way to capture datatype idiosyncrasies
- Completeness of datatype inference cannot, in general, be guaranteed
- No approach for multiple-datatype inferences is worked out yet

Demonstration…

- Preview script
  - file: VehicleCapacity.ss
- Run Swish
  - file: SwishMain.hs