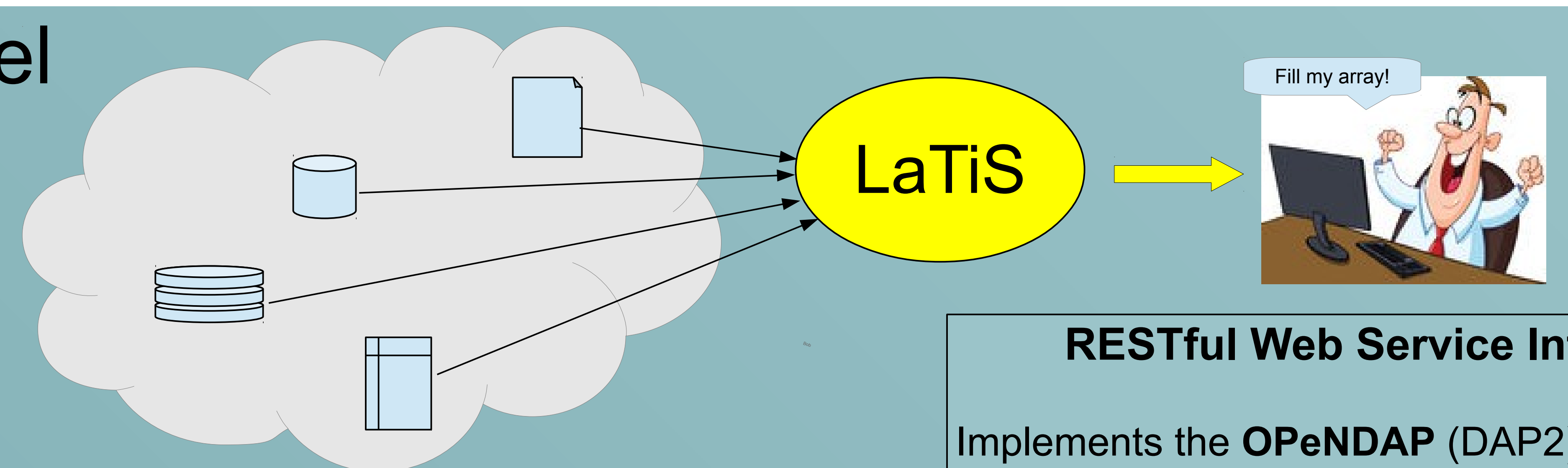
Laboratory for Atmospheric and Space Physics
University of Colorado **Boulder**

LaTiS: Enabling Interoperability via a Universal Functional Data Model

July 2014

Providing unified access to **data** via a common interface by describing datasets in terms of a basic, yet extensible, data model that expresses the functional relationships that are inherent in scientific data.

Doug Lindholm (doug.lindholm@lasp.colorado.edu)



Abstract

LaTiS is a software framework for data access, processing, and output. The modular architecture supports reusable and custom Readers to read a dataset from its native source, Operations to manipulate the dataset, and Writers to output the dataset in the desired form. Datasets can be read from diverse sources, combined in various ways to derive new datasets, and written to any number of formats. LaTiS can enable simple access to a single data file or it can be used to orchestrate an entire data processing workflow.

The core feature of LaTiS that enables these capabilities is its Functional Data Model. This data model extends the Relational Data Model to add the concept of Functional Relationships between independent and dependent variables which are prevalent in scientific data. This model provides a mathematical foundation for describing any dataset in terms of only three variable types:

Scalar: A single Variable.
Tuple: A group of Variables.
Function: A mapping from one Variable (domain) to another (range).

Since Variables can be any one of these three types, they can be composed in arbitrarily complex ways to represent the underlying nature of any dataset. LaTiS does not simply provide an alternate data model, it can be used to represent the fundamental mathematical structure of any other data model. In other words, any data model can be mapped to the lower level LaTiS Data Model. The Variable types can then be extended to add semantics that are specific to a particular scientific discipline (e.g. Temperature as a specific type of Scalar). The resulting datasets can still be used by those outside that discipline, albeit with a loss of the extra semantics, because the model representation can always be reduced to the basic three variable types. In this way, LaTiS provides a new level of interoperability between disparate datasets.

The Open Source Scala implementation of the LaTiS Data Model (https://github.com/dlindhol/LaTiS) is designed to serve as a Domain Specific Language (DSL) for scientific data analysis. The version 2.x series is in part a testbed for the evolution of the LaTiS DSL. Version 3 of LaTiS will provide a full featured interactive data analysis tool using the Scala REPL (command-line interface) to build on the ability of LaTiS to represent scientific datasets with higher level Functional semantics (e.g. time series of grids) as opposed to the more commonly used multi-dimensional array constructs. Combining the Functional semantics of the data model with the Functional Programming constructs of Scala promises to be quite powerful.

Perhaps the current "killer app" for the LaTiS framework is its web service interface to the underlying LaTiS DSL. This RESTful API implements the standardized OPeNDAP (DAP2) data request and reply specification. Data providers can easily install a LaTiS Server and expose the datasets they wish to serve by creating TSML dataset descriptors. The service layer will accept queries that include selection constraints (e.g. time > 2014-05-01), as opposed to requiring array index notation, and function calls (e.g. format_time(yyyyDDD)) that can subset, aggregate, and perform other operations on the server before writing the resulting dataset to the client.

Evolution of Data Abstractions

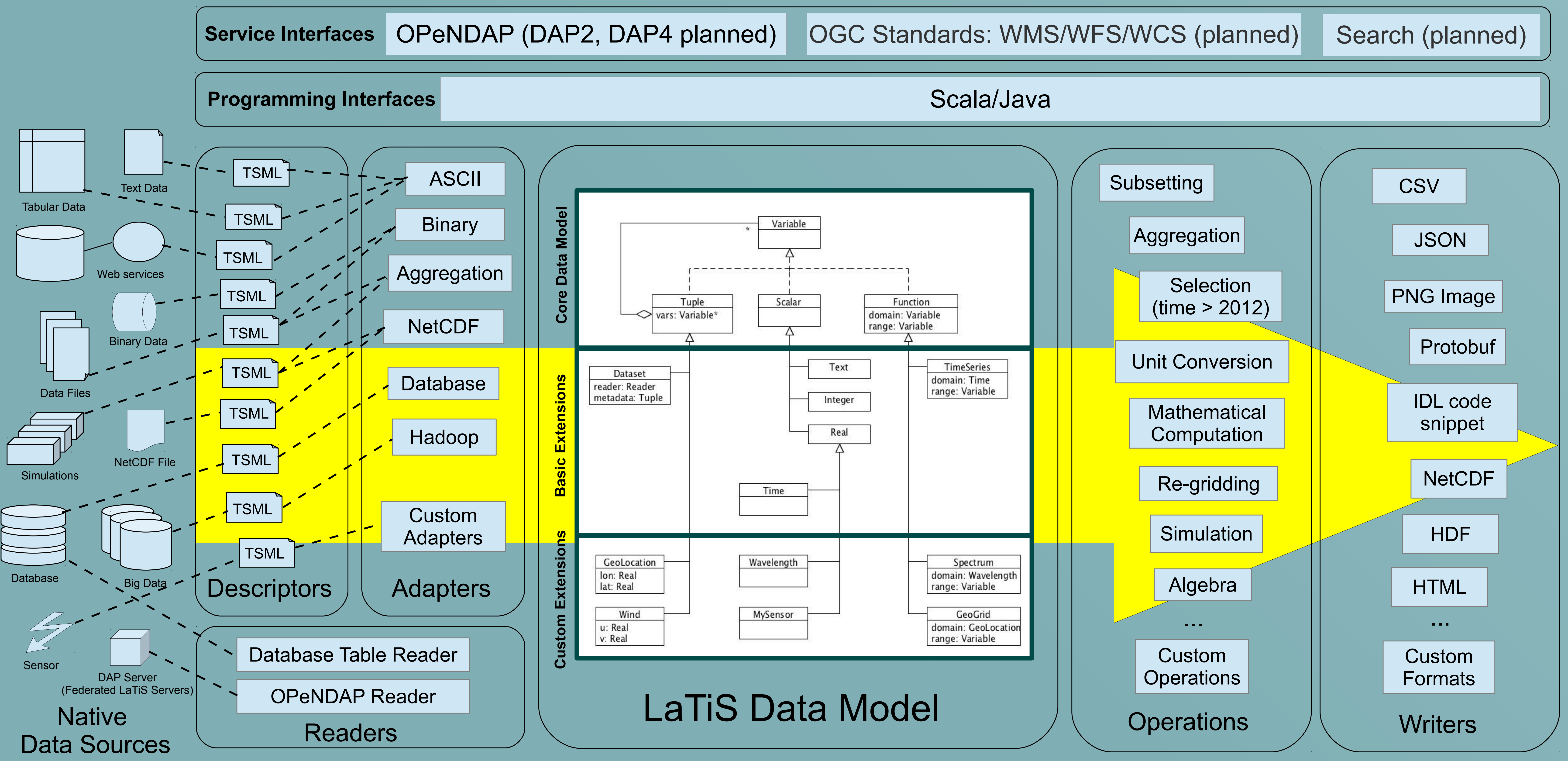
```
Bits: 10110101000001001111001100110011

Bytes: 05e0 e6b0 343b 9c74 0804 e7bc

int, long, float, double, scientific notation:
  1, -506376193, 13.52, 0.177483826523, 1.02e-14

Array: [1.2, 3.6, 2.4, 1.7, -3.2]

Structure: {
  city: “Frisco”
  time: 2014-07-09T08:00
  temperature: 60
}
```

This is where data modeling forked. Computational science data applications embraced **multi-dimensional arrays**. Applications with less numeric data embraced **Relational Databases**. A large divide remains between these data management approaches.

The LaTiS data model is designed to capture the underlying mathematical structure of all data models. LaTiS extends the **Relational Data Model** by capturing the **functional relationships** that are inherent in most scientific data.

$$\text{Independent Variable (domain)} \Longrightarrow \text{Dependent Variables (range)}$$

From the relational data model perspective, instead of Representing data as only a sequence of Tuples (rows in a table), LaTiS can express that some parameters depend on others. From a multi-dimensional array perspective, the dimensions of the array become first class Variables as the domain of a Function. In this way, the LaTiS data model captures the common abstraction represented by both data models providing a new level of interoperability. It also adds higher level **functional** semantics to a dataset which is often a better abstraction for reasoning about scientific problems.



RESTful Web Service Interface

Implements the **OPeNDAP** (DAP2) specification:

Usage:
http://server/latis/dataset.suffix?projection&selection&filter

suffix: type of output/writer
projection: list of variables to return
selection: relative constraint
(*e.g.* time>=2012-01-01)
filter: One or more functions to be applied to the data

Example:
http://lasp.colorado.edu/lisird/tss/historical_tsi.csv?time,Irradiance&Irradiance>1361.5

- Easily deployed as a Java Servlet with a highly extendable plug-in architecture.
- Other service interfaces can be layered on top of the LaTiS programming API.

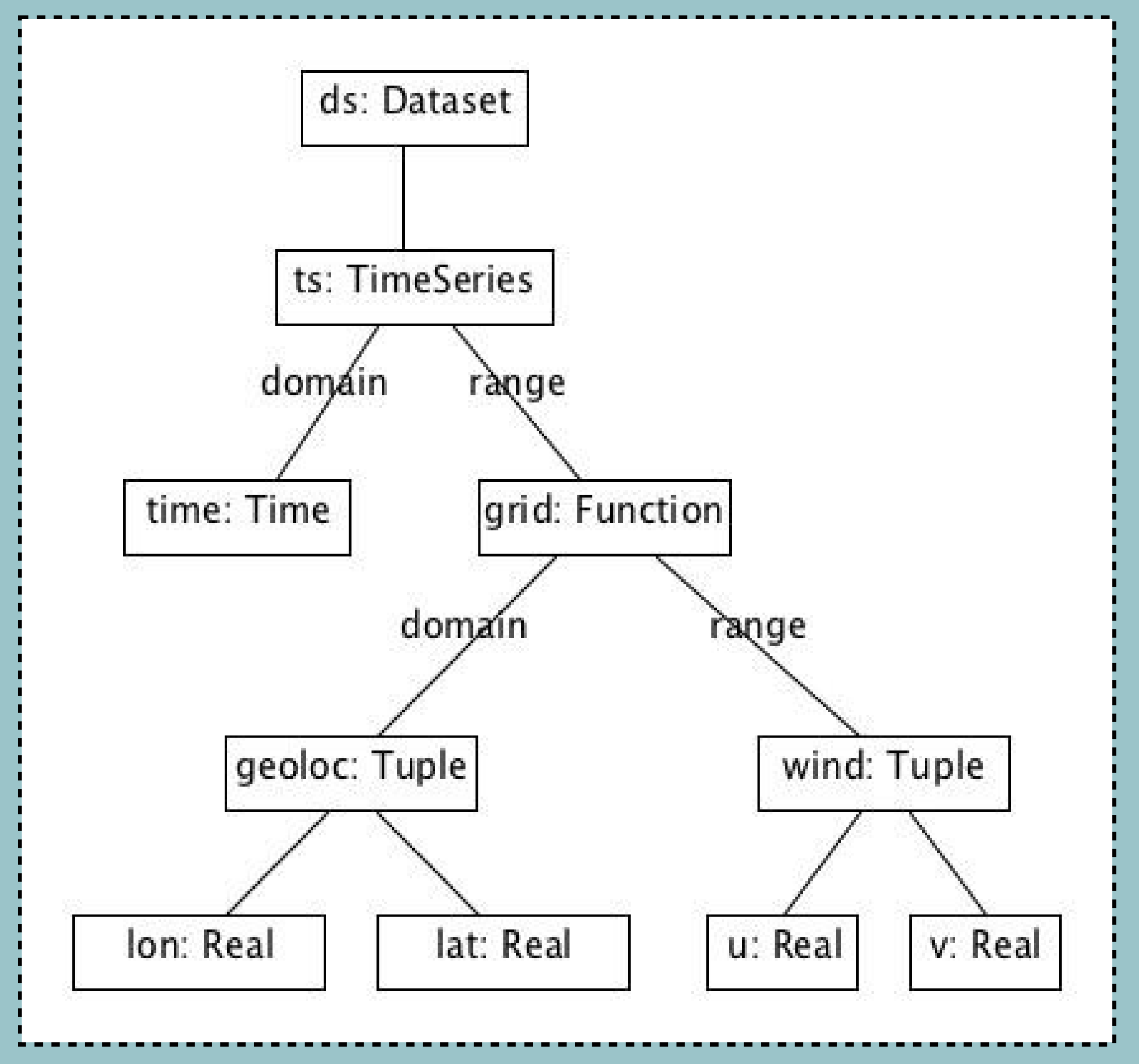
Scala/Java Programming API

- Designed around Functional Programming principles including typed lambda calculus and Category Theory
- Immutable data structures with no side-effects promote provable and parallelizable code
- Lazy evaluation means that data will be read only as needed, enabling the manipulation and streaming of arbitrarily large datasets
- Syntax enables natural mathematical expressions with data model components

Example: Time series of gridded winds

Without structural semantics, just a collection of variables (Tuple):
`(Time, Lon, Lat, U, V)`
Add “time series” semantics by factoring out Time as the independent variable:
`Time → (Lon, Lat, U, V)`
Likewise, factor out geo-location as the domain of the gridded wind values:
`Time → ((Lon, Lat) → (U, V))`
Which is logically equivalent to the 3D array:
`U[nTime][nLon][nLat]`
But can also be thought of as a 3 argument **function** that is evaluated by **values** instead of indices:
`U(time: Double, lon: Double, lat: Double)`
Which can be curried:
`U(time: Double)(lon: Double)(lat: Double)`
And be partially evaluated to result in a new function:
`U(time=0) => U0(lon: Double, lat: Double)`
`...`

Object Diagram



Dataset Descriptor

```
<dataset>
  <adapter class=”...” location=”...” />

  <function type=”TimeSeries”>

    <time units=”days since 1980-01-01”/>

    <function name=”grid”>

      <tuple name=”geoloc”>
        <real name=”lon”/>
        <real name=”lat”/>
      </tuple>

      <tuple name=”wind”>
        <real name=”u”/>
        <real name=”v”/>
      </tuple>

    </function>

  </function>

</dataset>
```

Domain Specific Language (DSL) for Scientific Data Analysis (evolving)

Make use of Scala's syntactic sugar and take advantage of it's command-line interface (REPL) and scriptability to provide a simplified language that more directly meets the needs of data users by allowing them to solve problems based on higher level semantics that match their domain.

See Also

LaTiS Open Source Project:
https://github.com/dlindhol/LaTiS

LASP Interactive Solar Irradiance Data Center (LISIRD): http://lasp.colorado.edu/lisird/

Time Series Data Server (TSDS, first generation of LaTiS based on NetCDF CDM): http://tsds.net