LaTiS: Enabling Interoperability via a Universal Functional Data Model

Abstract

LaTiS is a software framework for data access, manipulation, and output. The model architecture incorporates reusable and customizable Readers to feed a dataset from various sources. Operations in the model are performed on Java objects representing the model's concept of a variable. This model allows users to perform operations on data with a methodological consistency that more directly meets the needs of data users, and scriptability to provide a simplified language advantage of its command-line interface (REPL).

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

RESTful Web Service Interface

Implements the OpenDAP (DAP2) specification:

Usage:

http://server/lats/dataset.suffix?query=don&selection=&filter

suffix: type of output/reader 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,iradiance&iradiance=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 an 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[i][lon][lat]

But can also be thought of as a 3 argument function that is evaluated by way 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) => u[0](lon:Double, lat:Double)