BEAST 2.0

April 29, 2010

Vision

To provide tools for computational science that are

- 1 *easy to use*, that is, well documented, have intuitive user interfaces with small learning curve.
- 2 *open access*, that is, open source, open xml format, facilitating reproducability of results, runs on many platforms.
- 3 *easy to extend*, by having extensibility in design.

Scope

Efficient Bayesian computation for sequence data analysis involving tree models.

Basic design

Bayesian computation centers around MCMC algorithsm. The basic structure of an MCMC algorithm is shown in Figure 1. A glance at this bit of pseudo code reveals that the least that is required are the following components:

```
Read data D
Initialize state S
L = likelihood(S, D)
while (not tired) {
    S' = proposed state
    L = likelihood(S', D)
    Accept or reject S'
    Log state
}
```

Figure 1: Basic structur of MCMC.

- a data object that contains sequence data,
- a state object to represent the current and proposed state. The state consists of at least one tree (see scope) and parameters, which are integer or real valued.
- likelihood/density objects to calculate the posterior.
- operator objects to work on the state and propose new states.
- log objects, since we are interested in the results which have to be recorded somewhere.
- an MCMC object to control the flow of the computation.

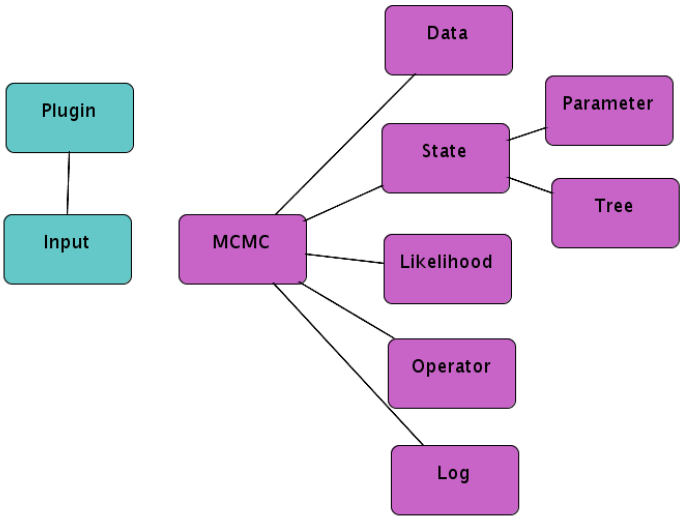

Figure 2: Core classes. Every class except `Input` is derived from `Plugin`, but these relations are not shown in the diagram.

So, that leaves us with at least a `Data`, `State`, `Parameter`, `Tree`, `Likelihood`, `Operator`, `Log` and `MCMC` object. Since trees consist of nodes, a `Node` object is desirable as well.

Since we want the system to be extensible (Vision 3), everything in the system is a `Plugin` and can be replaced by a custom implementation. So, each of the above objects are derived from `Plugin`. To allow flexibility of plugins, every plugin can specify inputs. `Input` objects contain information on type of the input and how they are stored in BEAST XML files.

Basic structure of BEAST - basic design

A closer look at the Core classes - introducing interfaces for each of the classes - MCMC/Runable - Likelihood/Probability - Operator - Log - State - Tree - Parameter - Plugin - Input

BEAST 2.0 XML - file format and how the parser processing model - Roll your own: extending through namespaces, maps

Nucleotide TreeLikelihood: an in depth look - go in the gory details of TreeLikelihood, substitution models, etc

Recommended design patterns - validation & initialization patterns - efficiency through store/restore