The need to categorize types

Here is a function that constructs an object representing the state of a Markov process:

```
template <class Process>
Process create(Parameters parameters) {
    return Process(parameters.r(), ...);
}
```

Works with various flavors of a branching process (a class of Markov processes) ...

... but not for processes that do not contain a parameter “r”, e.g. Moran processes

What we really need is this (pseudo code):

```
if(Process is a an example of a branching process)
    Process(parameters.r(), ...)

if(Process is a an example of a Moran process)
    Process(...)
```

But how do we actually implement this?

Traits permit category-dependent code execution

Nest the category information inside each Process, e.g.

```
class Branching_vanilla {
     public:
          typedef branching_category category;
     ...
}
```

Overload ‘create’ to construct a Process in a category-dependent way:

```
template <class Process>
Process create(branching_category, Parameters parameters) {
     return Process(parameters.r(), ...);
}

template <class Process>
Process create(moran_category, Parameters parameters) {
     return Process(...);
}
```

Use Process’s ‘category’ to compile the right implementation of Process creation:

```
Process process = create <Process> (typename Process::category, parameters);
```