BioResolve

A logical framework for Reaction Systems

Functionality

Given:

- a set of *entities*
- a set of *reactions* defined over the entities
- a possibly emtpy *environment*
- a *context*

the framework executes the parallel process(es) defined over these items, finally creating a ***DOT file graph*** illustrating the path(s) leading to the result(s).

Definitions

A reaction is defined as

```
([reactants], [inhibitors], [products])
```

that is a triple containing in each field a (possibly empty) list of entities.

The set of reactions is a comma-separated list of these triples.

As defined in the paper, a reaction is fired only if the set of entities over which it is tested does contain all of the reactants and none of the inhibitors, producing the list of entities defined as products.

Definitions

The context is defined as a dot-separated sequence of *entities*, *variables*, *repetitions*, and *non-deterministic choices*. Furthermore, the context can itself be a list contexts, making them *parallel* to each other.

For example:

```
{a,b}.{c}.x.nil, <3,x>.{d}.nil
```

are two parallel contextes, where the two initial components of the first are sets of entities, the third is a variable defined inside the environment, and *nil* denotes the end of the sequence. Considering the second, its first component is a triple repetition of the "*x*" variable (which can actually be rewritten as *x.x.x*), the second component is a single entity, and the termination is again through *nil*.

Definitions

The environment is defined as a comma-separated sequence of constant variables, each having assigned a context-like sequence.

For example:

```
x = {a,b}.{c}.nil, y = {a,b}.{c}.x + {d}.y
```

defines two environment variables containing their right hand side context each. The right hand sight of the "*y*" variable is an example of non-deterministic choice.

These variables are used by the context and can be recursive (possibly leading to non-termination).

How the framework works

The program contains a hierarchical set of components, each of them having some specific responsibility.

In particular, a unique, global **coordinator** is created before starting the computations.
The coordinator's responsibility is to manage all the *process managers* and to cache their results.
The results are cached as *node pairs*, that is the exact representation that will be finally yielded by the DOT graph.

How the framework works

A **node pair** contains a source node, which represents the current state of a process, and a destination node, containing the result of the advancement of the process.

The arc connecting them represents the union of the current state and the current context's element which lead to the next result.

To disambiguate node pairs having the same source and the same destination but resulting from different *Interactive Processes* (thus having different internal state), the relative remaining context sequence is appended to both nodes.

Node pairs are used both to create the final DOT graph and to cache the results, allowing to terminate recursive definitions.

How the framework works

A **process manager** is in charge of executing the possibly parallel *Interactive Processes* which it is in charge of.

In particular, a process manager groups the current states from all of its processes, computes the resulting entities by testing it against the Reaction System, and advances the processes states by pushing the result to each of them.

Furthermore, each manager is in charge of creating a *node pair* for each computation.

A new process manager is created each time a non-deterministic choice is present as the current *Interactive Process*' step.

How the framework works

The **Interactive Process** contains the context sequence γ and the result sequence δ and is in charge of yielding the state sequence $W_i = D_i \cup C_i$ to the manager for it to compute D_{i+1}.

To do so, it manages each different component of the context, acting accordingly. For example, if the current position C_i corresponds to a non-deterministic choice context component, a new *process manager* is created for the management of each choice.

Improvements over Prolog's implementation

- Introduced a (basic) ***GUI***
- Up to ***5x*** execution time speedup
- Automatic entities extrapolation from the reaction string → no need to define the entities anymore