Semantics of Rauzy language

(Semantics defined by Jean-Baptiste)

1.Vocabulary

We reference here RFC 2119.

1. MUST: This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.
2. MUST NOT: This phrase, or the phrase "SHALL NOT", mean that the definition is an absolute prohibition of the specification.
3. SHOULD: This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
4. SHOULD NOT: This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.

2. Parsing rules

The absence of mandatory keys in the json objects or arrays MUST raise an error. The presence of other keys not defined in this standard MUST NOT raise an error. In particular it can be used to develop extensions.

Note that all characters used in the syntax are case sensitive

3. Loading a file

When loading a file (i.e. interpreting the content of a file as an object described using the Rauzylanguage), the “library” member of the root object, if defined, MUST be used to import the prototypes/classes defined in the library file. The “library” value MUST be used as a relative path from the location of the file.

1.1 Loading the library

When loading a library, two environments are created, one for the object classes and one for the relation classes. Every pair (name, obj) in the list of named objects MUST create in the environment for objects classes the association between the string name and the object obj. The set of names will be referred to as the environment of classes or simply classes if it is not ambiguous.

Similarly, every pair (name, relation) in the list of named relations MUST create in the environment for relation classes the association between the string name and the relation relation. The set of names will be referred to as the environment of relations or simply classes if it is not ambiguous.

The objects defined in the list of named objects MUST be able to refer to relations extending relations present in the environment of relations. For that reason, the list of named relations SHOULD be parsed first.

1.1.1 Loading the relations

The relations present in the list of named relations, SHOULD have empty values for the “from” and “to” fields since their represent names of objects and that no objects are defined in this scope.

1.1.2 Loading of the objects

It MUST be possible for a class A to extend an other class B of the list, without any consideration for the order of A and B in the list of named objects. Also, it MUST be possible for a class A to include objects extending a class B. Thus, the loading of the object classes MUST take into account dependencies and work as long as there are no cyclic dependencies. In the case of cyclic dependencies, an error SHOULD be raised.

1.2 Loading an object

We suppose that the environment of object classes and relation classes is defined.

It is recommended that the presence of a non-empty “library” member in an object that is not the root object, does not raise an error.

The relations can refer to objects contained in the object one is currently parsing. Then, the relations SHOULD be parsed after the parsing of the “objects” field.

When parsing relations, they can refer to names of objects defined in the object that is currently parsed, or in descendants of these objects. Thus, the name SHOULD not be ambiguous (i.e. exist twice in the descendant or the contained objects). If the name is ambiguous, the behaviour is not defined.
When creating an object A that extends B:
- B MUST be present in the environment of object classes (i.e. must have been defined in the libray file). If not, this SHOULD raise an error like 'Reference to an undefined class B'.
- only properties can be added in A. One SHOULD raise an error if the fields “objects” or “relations” are not empty in the json description. Indeed, the only objects and relations are the one that come from the object class associated to the name B