BBF RFC 30: Draft of an RDF-based framework for the exchange and integration of Synthetic Biology data

Raik Grünberg

April 24, 2009

1 Purpose

This Request for Comments (RFC) suggests a framework for the description, exchange and interlinking of Synthetic Biology data. The framework would create an open process for the evolution and "rolling" standardization of data models. It describes how data and data models are to be published, how they are exchanged and integrated between different parties, and how they can be extended, corrected and interlinked in a decentralized fashion. These goals are achieved by embracing the RDF (Resource Description Framework), a set of W3C standards. A one-sentence summary of this proposal would therefore be: "Use RDF according to the W3C standards." The PoBoL project (Provisional BioBrick Ontology Language, `http://pobol.org`) is based on this idea.

This RFC does *not* describe a data model per se but only outlines a possible architecture and rules of data exchange.

2 Relation to other BBF RFCs

BBF RFC 30 does not update or replace any earlier BBF RFC.

3 Copyright Notice

4 Background and motivation

The idea of sharing standardized parts and protocols is central to Synthetic Biology. Yet, we have so far not been able standardize the exchange of data about these parts and protocols. This lack of data exchange protocols already affects the progress of Synthetic Biology software tools and limits the exchange of information within the community.

Unfortunately, an all-encompassing data model for Synthetic Biology or even only for standard biological parts seems completely out of reach. The field is simply too young and needs are too diverse. Even a consensus model for the description of standard biological parts, as much as it is needed right now, will soon become insufficient and outdated. Furthermore, a fixed data model runs contrary to our need of "playing around" and exploring different strategies, for example, in part design, device measurement, or system simulations. What we need instead is an open communication model which:

- Allows us to quickly fix operational data models for well defined problems.
- Remains open for extension and modification.
- Supports a decentral community-driven development.

An additional challenge is that Synthetic Biology data are not forming "closed systems" but are actually diverse and interconnected. BioBrick parts are often related to other parts, are embedded into various devices, which in turn may be subjected to various measurements which in turn provide data for various simulations which may feed back into different designs, and so on. An open communication model should, ideally, not only allow the extension of data *models* but also:

- Allow the extension and interlinking of the *data* itself.

Fortunately, we are not the first to face these challenges. The World Wide Web Consortium (W3C) has developed the Resource Description Framework (RDF) [1] – a set of standards that addresses all four points raised above. RDF breaks with the idea of putting data into fixed tables. Instead"facts" are put into "triples" of subject–predicate–object (i.e. noun or instance – verb or property – object or value). Subject and predicate must be identified by a unique address which typically translates to a location on the web. The object can be a simple value but more often links to another subject. What it boils down to is that data are uniquely identified, remain connected to their meaning and can be linked through the web.

RDF documents can be serialized into XML, JSON and a number of other file formats. Most available tools support the reading and writing to

XML and the N3/Turtle format. The simple examples below are written in the "Notation 3" or N3 format. N3 allows to group statements about a certain subject into blocks which gives a more object-oriented ("subject-oriented"?) view. The result looks very much like a classic database record. In fact, relational data are easily exported into RDF documents. Problems may only arise in the other direction: The relational data model is more rigid and restrictive than RDF. Third-party RDF data will therefore rarely immediately map into a pre-existing relational database scheme – unless, of course, both sides are complying with a common data standard.

This RFC outlines some simple rules that would establish an RDF based data communication framework. The technical requirements are indeed quite simple. We need (1) the RDF definition of a core data model, (2) some guidelines for the extension of this data model, (3) a few recommendations for data publication and synchronization and (4) software or servers that can read and write RDF documents.

5 Detailed description

5.1 Establishing a core data model

The BioBricks Foundation (BBF) MUST publish an RDF document defining the core concepts of the data model. This document is henceforth termed the *core standard*. The core standard SHOULD, for example, define the concepts of a "BioBrick part", a "BioBrick Format", "BioBrick Vector" and other concepts that are deemed useful or on which the community has reached consensus. The definition is not required to be complete. It only serves as a point of reference for the evolving data scheme. The document MUST adhere to the Web Ontology Language (OWL) standard. It SHOULD be formatted in XML but MAY also be offered in a more readable RDF serialization like N3/Turtle. This document MUST be made available at a permanent and immutable location on the web.

In order to avoid compatibility issues, the document SHOULD be created with established RDF editing programs like, for example, Protégé and it MUST be validated against an independent RDF validator. The W3C keeps a long list of RDF/OWL editors, validators and development tools [2].

The core data model SHOULD be described in a separate RFC.

Example:

The BBF places an Unicode encoded, N3 formatted file at: `http://biobricks.org/rdf/core/v1.n3`. This file defines the concept of a *BioBrick* and declares that each BioBrick can have a single property (or field) of type *dnaSequence*:

```
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .

:Biobrick
      rdf:type owl:Class ;
      rdfs:comment "representation of a Biobrick" ;
      rdfs:subClassOf owl:Thing .

:dnaSequence
      rdf:type     owl:FunctionalProperty ;  ## only one value per part
      rdfs:comment "property type for a part's DNA sequence";
      rdfs:range   xsd:string;
      rdfs:domain  :Biobrick.
```

Another XML-formatted version of the same file may be made available at: `http://biobricks.org/rdf/core/v1.xml`.

The same file MAY be routed to a static address that is always pointing to the latest version of the standard: `http://biobricks.org/rdf/core/current.xml`.

Please note, unlike a classic data base schema, RDF is "open" by definition. The above file does *not* imply that BioBrick records can have *only* dnaSequence properties. Anyone is free to define and assign additional properties. For example, the very popular FOAF (Friend of a friend) ontology already defines a property "foaf:maker" which connects any "owl:Thing" to a person who has created it. This property can be immediately used for BioBrick records because BioBricks are also derived from "owl:Thing".

5.2 Extension of the core data model

Third parties MAY extend the *core standard* whenever this proves necessary. The extension MUST be published as an RDF/OWL document at a permanent and immutable location on the web. The document MUST link back to the core standard and MUST only contain the additional or modified definitions. If the extension model is building on further third-party extensions, these extensions SHOULD also be referenced and their content SHOULD NOT be repeated.

Whenever appropriate, extension authors SHOULD re-use definitions from well supported other RDF ontologies. Examples are the sequence ontology project [3] for sequence related information and Foaf [4] for user descriptions.

The extended data model SHOULD be used immediately without approval of any kind. It is RECOMMENDED to announce the extension on the BBF technical standards mailing list, along with a set of example data.

Example:

A Synthetic Biology team at Hong Kong University concludes that BioBrick parts should also contain a human-readable description. They publish a OWL/RDF file at the address `http://hkust.edu.hk/sb/rdf/bbv1.n3` which defines a new property of BioBrick parts:

```
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .
@prefix bbf:     <http://www.biobricks.org/rdf/core/v1#> .

:description
      rdfs:domain bbf:Biobrick ;
      rdfs:range xsd:string .
```

Their own data about BioBricks can now mix the core standard with the extension. So their first Hong Kong BioBrick may look like this:

```
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix bbf:     <http://www.biobricks.org/rdf/core/v1#> .
@prefix hk :     <http://hkust.edu.hk/sb/rdf/bbv1#> .

:HK1000
      rdf:type        bbf:BioBrick;
      bbf:dnaSequence "AGGAGGTGG";
      hk:description  "highly optimised RBS".
```

They can also annotate existing MIT BioBricks with their new data field:

```
@prefix mit:     <http://partsregistry.org/rdf/parts#> .
mit:P1010   hk:description   "ccdB death casette".
```

RDF-aware software could then automatically merge the information that is available from the MIT with the annotation provided by the team from Hong Kong. This example assumes that the MIT registry indeed provides an RDF "view" of their parts. See below.

5.3 Rolling standardization of the data model

Extensions that are popular or deemed useful by several labs SHOULD be submitted for standardization. The authors SHOULD describe their extension in a short BBF RFC. The RFC SHOULD provide a brief motivation, and describe or point to example data that are using the extension. The RFC MUST contain a link to the permanent definition of the extension and it is RECOMMENDED that the actual RDF document is also appended to the RFC, preferably, in a more human-readable format like N3/Turtle.

Depending on the popularity and scope of the extension, the BBF MAY include it directly into a revised version of their *core standard* or they MAY choose to host it in a separate RDF document. In any case, owl:sameAs links SHOULD connect the new standard back to the RDF document of the original proposal.

5.4 Exchange and publication of data

Synthetic Biology data SHOULD be published in RDF documents that are importing concepts and properties (fields and types) from the BBF core standard and, as needed, from standard or non-standard extensions. The data documents SHOULD be serialized to XML but, depending on the situation, other formats, like Turtle/N3 or JSON MAY be preferred.

Data MAY be exchanged in freely floating "unbound" documents that are copied back and forth, attached to e-mails etc. However, unbound data cannot be referenced by third parties, are subject to version problems, redundancy and inconsistencies. Whenever possible, RDF data documents SHOULD thus be hosted at permanent immutable locations. In particular, data associated to published articles MUST be made available at a permanent immutable location on the web. Different institutions like, for example, part registries, journals, or specialized databases MAY offer to host the RDF data on behalf of the authors.

Third party data stores MAY chose to import partial or full copies of these data into their own system. Copies of these data MAY then be meshed up and re-published as RDF but MUST always refer back to the original location.

Example

The Hong Kong University team from the example above decides to host their BioBrick data on their own server at the address `http://hkust.edu.hk/sb/rdf/parts`. The MIT parts registry may nevertheless choose to re-publish the new part "HK1000". However, the MIT registry uses a classic relational database back-end and doesn't know how to deal with the new "description" property. So it simply leaves it out. Note, that the copy of the BioBrick is pointing back to the original description. Instead of a local ":HK1000" it is identified by a full address "hkpart:HK1000" which translates to `http://hkust.edu.hk/sb/rdf/parts#HK1000`, the original location in Hong Kong.

```
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix bbf:     <http://www.biobricks.org/rdf/core/v1#> .
@prefix hkpart:  <http://hkust.edu.hk/sb/rdf/parts#> .

hkpart:HK1000
      rdf:type        bbf:BioBrick;
      bbf:dnaSequence "AGGAGGTGG".
```

5.4.1 Legal issues

In cases, where data are covered by legal terms and conditions of any sort, these terms MUST NOT put any restrictions whatsoever on modification, mesh up and re-publication. The original data MAY contain references to a certain license. The obligatory back-link ensures automatically that this license remains accessible also from copies of the data. Note, Creative Commons (CC) are using RDF to embed license information in documents [5]. The CC RDF terms can probably be re-used and adapted into a Synthetic Biology license data model. Preferably though, and following good scientific practise, data should not be encumbered by any legal terms.

5.5 Interlinking and synchronization of data

Third parties can link to and extend the data of others. Examples are already given above in section 5.2. The mesh up SHOULD again be published in RDF documents at permanent locations. Centralized data stores MAY consolidate certain types of data, for example, about BioBrick parts. This MAY include a review process and quality control measures, perhaps at the cost of leaving out less standardized types of information.

Data stores MAY use RSS feeds to synchronize data among each other. The RSS format (another W3C standard) is based on RDF and can be directly imported. This strategy is simple and avoids any kind of write access.

The receiving data store MAY read and import part or all of the new information at its own discretion.

Example

The Hong Kong BioBrick server offers a RSS feed that announces each new BioBrick. The parts registry server at the MIT has subscribed to this RSS feed and automatically imports every new part as soon as it is announced. The MIT registry's own RSS feed could then announce the new part to other data stores that were not aware of the server in Hong Kong.

5.6 Data import and export

Synthetic Biology data MUST be available as RDF documents with a simple read access from the unique address of the data. That means a simple *HTTP GET* MUST serve the document just as it would serve an html formatted web page about it. That also means data access SHALL NOT require the initialization of web services or any other kind of remote procedure calls.

Software that consumes Synthetic Biology data records MUST be able to open, parse and interpret RDF documents. Depending on its purpose, the software is NOT REQUIRED to interpret every single (standard or non-standard) statement in the document. However, it MUST make a best effort to gracefully ignore any non-standard statement in order to allow for the expansion of the data model. The software SHOULD, at least, parse XML-formatted RDF documents. Support of more specialized and readable formats like Turtle/N3 is RECOMMENDED. It is RECOMMENDED that the software be able to also directly read remote RDF documents via http.

Larger data stores are RECOMMENDED to serve only part of the RDF document when a particular data item is requested with the standard syntax `http://address/document#item`. At least in the long term, data stores are also RECOMMENDED to support the SPARQL W3C standard for more complex queries.

Example

The Hong Kong team publishes their only 10 or 15 BioBrick records in a single RDF document which is served statically through their web server at `http://hkust.edu.hk/sb/rdf/parts.xml`. By contrast, the MIT registry uses a database backed server and generates RDF documents dynamically. Accessing the address `http://partsregistry.org/rdf/parts` would return a long RDF document containing all registered BioBricks. A request of `http://partsregistry.org/rdf/parts#P1010` returns a short RDF containing only a single BioBrick entry, for example:

```
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix bbf:     <http://www.biobricks.org/rdf/core/v1#> .

:P1010
      rdf:type        bbf:BioBrick;
      bbf:dnaSequence "AAGTCCTAAAT....".
```

A Python programmer could access this BioBrick remotely using standard RDF libraries. For example:

```
from rdflib.Graph import Graph

g = Graph()
g.parse("http://partsregistry.org/rdf/parts#P1010")
```

Preferably though, the Synthetic Biology community should develop libraries that encapsulate the RDF data into more customized objects.

6 Discussion

6.1 Advantages of the RDF architecture

The classic approach to data standardization is adapted to the classic means of storing these data, namely in relational databases. It requires the design of tables, fields within tables, and relations between tables *before* the first data can be entered. For this reason, data standardization efforts like the Systems Biology Markup Language (SBML) [6] aim to work out a detailed data model which suits all needs. This data model is then serialized to a common exchange format, typically XML, and recommended as a standard. However, the process is difficult and it often takes years before a first standard is available and even longer for software to catch up on it.

I here try to make the point that we should choose a different, more incremental approach. We SHOULD base our data exchange efforts on RDF and this Request For Comments tries to sketch how this COULD look in detail. RDF “solves” our immediate issues of openness, extension, distribution and scalability of data and data models. Obviously, it doesn’t make these problems disappear but it offers a clear strategy for handling them. Moreover, this architecture has several additional advantages:

1. It avoids any custom-made protocols or formats and is based on existing standards.

2. The architecture adheres to "REST" principles [7], that means it transmits data directly over HTTP without an additional messaging layer [8]. Data are fully uncoupled from software. There is no need for web services, Soap or other remote procedure calls which makes it simple, lean, less error prone, and more flexible.

3. Synthetic biologists will automatically benefit from the ongoing, and right now accelerating, development of RDF tools and infrastructure. By adopting RDF, we are effectively outsourcing our data integration issues to expert computer scientists and enterprises.

4. The RESTful interconnection of data will, in the long run, lead to mesh-up applications and services that are otherwise simply impossible. Examples are search engines that crawl distributed data sets for suitable biological parts; social networks that connect biological engineers through the parts or protocols they are using; and many more.

6.2 Software support

The RDF standard has been set 10 years ago and there is now a plethora of software development tools available [2]. Libraries for the low-level reading and writing of RDF documents abound and pretty much every programming language is supported. Some packages also offer API bindings for several programming languages. For instance, the Redland RDF libraries (http://librdf.org) are written in C but can be used from Python, Ruby, PHP and Perl.

By comparison, tools dealing with the classic relational database model are, nevertheless, more mature and better known. After all, this model of data handling has been around since more than 40 years. In fact, this RFC is not pitching one model against the other. After decades of optimization, relational databases are very efficient at handling large sets of *local* and homogeneous data. This RFC assumes a mixed landscape, where local data are often handled by relational databases but data exchange is based on RDF. In fact, the underlying technology becomes irrelevant as long as data are published as RDF. A software application would, in this scenario, not even notice whether the data it imports were generated by a relational database, a semantic triple store, or are coming from a plain static file.

In the near term, the scene will probably be dominated by classic database servers and software parsers emitting and consuming a certain file format (which happens to be RDF). Rolling standardization keeps this format in sync and helps database or application developers to manually adapt their static data models. In theory, there should be only little overhead involved – developers may need to abandon their favorite XML parser because it doesn't support RDF (although in most cases it probably will) and data base admins will need to ensure that the data they export are passing an RDF validator and are pointing to the right ontology.

In the mid term, as more and more RDF data come online, this will motivate developers to play with the additional possibilities of RDF. This, in turn, should lead to increasingly advanced applications which can integrate data from different sources automatically. In the long term, synthetic biology data will more and more blend into a larger web of data that is weaved by scientific and non-scientific communities. Well curated parts registries may, in fact, turn into hubs and nucleation points for such data webs. But then, as J.M. Keynes put it, "In the long term we are all dead." So let's focus on the next steps.

7 Further developments of this RFC

This RFC is primarily meant as a basis for discussion. It could be developed into a new master RFC that would be complemented by additional standards on:

- definition of the core data model
- description of a versioning strategy
- guidelines for software / server developers

8 Authors' Contact Information

Raik Grünberg: `raik.gruenberg@crg.es`

References

[1] The World Wide Web Consortium. Resource description framework (RDF) / W3C semantic web activity. http://www.w3.org/RDF/, 2004. Available from World Wide Web: `http://www.w3.org/RDF/`.

[2] W3C. SemanticWebTools - ESW wiki. http://esw.w3.org/topic/SemanticWebTools. Available from World Wide Web: `http://esw.w3.org/topic/SemanticWebTools#head-142cfa85b3be9cef7dc46bff5bba70bf03e3e7cf`.

[3] The Sequence Ontology Project. The sequence ontology. http://www.sequenceontology.org/. Available from World Wide Web: `http://www.sequenceontology.org/`.

[4] The FOAF project. FOAF vocabulary specification. http://xmlns.com/foaf/spec/. Available from World Wide Web: `http://xmlns.com/foaf/spec/`.

[5] Creative Commons. Describing copyright in RDF. http://creativecommons.org/ns. Available from World Wide Web: `http://creativecommons.org/ns`.

[6] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, , the rest of the SBML Forum:, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, A. A. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel, V. Gor, I. I. Goryanin, W. J. Hedley, T. C. Hodgman, J.-H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, A. Kremling, U. Kummer, N. Le Novere, L. M. Loew, D. Lucio, P. Mendes, E. Minch, E. D. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, and J. Wang. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, March 2003. Available from World Wide Web: `http://bioinformatics.oxfordjournals.org/cgi/content/abstract/19/4/524`.

[7] R. T. Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, 2000.

[8] the free encyclopedia Wikipedia. Representational state transfer. http://en.wikipedia.org/wiki/Representational_State_Transfer. Available from World Wide Web: `http://en.wikipedia.org/wiki/Representational_State_Transfer`.