BIOINFORMATICS ORIGINAL PAPER

Vol. 27 no. 7 2011, pages 973–979
doi:10.1093/bioinformatics/btr048

Systems biology

Advance Access publication February 4, 2011

Model annotation for synthetic biology: automating model to nucleotide sequence conversion

Goksel Misirli[1], Jennifer S. Hallinan[1], Tommy Yu[2], James R. Lawson[2], Sarala M. Wimalaratne[3], Michael T. Cooling[2] and Anil Wipat[1,*]

[1]School of Computing Science, Newcastle University, Newcastle upon Tyne, UK, [2]Auckland Bioengineering Institute, The University of Auckland, New Zealand and [3]European Bioinformatics Institute, Hinxton, UK

Associate Editor: Jonathan Wren

ABSTRACT

Motivation: The need for the automated computational design of genetic circuits is becoming increasingly apparent with the advent of ever more complex and ambitious synthetic biology projects. Currently, most circuits are designed through the assembly of models of individual parts such as promoters, ribosome binding sites and coding sequences. These low level models are combined to produce a dynamic model of a larger device that exhibits a desired behaviour. The larger model then acts as a blueprint for physical implementation at the DNA level. However, the conversion of models of complex genetic circuits into DNA sequences is a non-trivial undertaking due to the complexity of mapping the model parts to their physical manifestation. Automating this process is further hampered by the lack of computationally tractable information in most models.

Results: We describe a method for automatically generating DNA sequences from dynamic models implemented in CellML and Systems Biology Markup Language (SBML). We also identify the metadata needed to annotate models to facilitate automated conversion, and propose and demonstrate a method for the markup of these models using RDF. Our algorithm has been implemented in a software tool called MoSeC.

Availability: The software is available from the authors' web site http://research.ncl.ac.uk/synthetic_biology/downloads.html.

Contact: anil.wipat@ncl.ac.uk

Supplementary information: Supplementary data are available at *Bioinformatics* online.

Received on November 6, 2010; revised on January 18, 2011; accepted on January 21, 2011

1 INTRODUCTION

Synthetic biology involves the design and implementation of genetic circuits to enable organisms to perform novel, desirable functions for biotechnology applications. Such applications include the production of medically relevant biomolecules (Anderson *et al.*, 2006; Ro *et al.*, 2006), environmental bioremediation (Sinha *et al.*, 2010) and biofuel production (Lee *et al.*, 2008).

Most genetic systems are currently designed manually by a domain expert with a deep understanding of the system to be engineered. However, as the aims of synthetic biologists become more ambitious, and designs correspondingly more complex, the manual design of systems at a genetic level becomes more challenging. Consequently, interest in the computational design of genetic circuits has grown rapidly over the last few years (Andrianantoandro *et al.*, 2006; Bolouri and Davidson, 2002; Endler *et al.*, 2009; Goldbeter, 2002; Hasty *et al.*, 2002; Weiss *et al.*, 2003).

Genetic circuits are usually designed and simulated *in silico* as abstract models. Computational models of genetic systems, such as Biobricks (Knight, 2003), are valuable because they allow rapid simulation of a system and verification of its behaviour under a range of circumstances. In synthetic biology, *in silico* models of modular components are typically assembled in a bottom-up fashion to produce a larger computational model of the desired system. Such models are usually constructed using abstract modelling formalisms such as Systems Biology Markup Language (SBML; Hucka *et al.*, 2003) and CellML (Cuellar *et al.*, 2003).

Once a suitable model for a system has been designed, the conceptual model must be transformed into a DNA sequence. This sequence encodes the necessary genetic features required for a designed circuit to be implemented *in vitro* or *in vivo*.

At first glance this transformation appears to be a relatively straightforward task to complete manually: components are selected and their DNA sequences are concatenated. Appropriate restriction sites can then be added, a cloning vector selected and the entire sequence synthesized or cloned, as appropriate.

In practice, the situation is far more complex, and for large models is time consuming and difficult to complete by hand. A typical computational model will contain numerous species or components that do not have a physical representation at the DNA sequence level. Examples include entities representing proteins, protein and RNA degradation, information flow, environmental inputs and chassis-related factors. In this article, we refer to model components representing biological parts with a DNA sequence as 'DNA-based parts'.

It is not always immediately obvious which entities in the model map directly to the genes and sequence features necessary to encode the system represented by the model. The fact that the mapping between component abstractions and sequence-based features is not necessarily one-to-one adds additional complexity. Other factors such as the spacing and ordering of physical features and the genetic elements used to ensure their replication, can also impact on a system's behaviour and must be considered.

As models for computationally designed systems increase in size and complexity, automatically deriving the physical DNA sequence

*To whom correspondence should be addressed.

necessary to encode a system designed as an *in silico* model becomes highly desirable, if not essential. However, carrying out the model-to-sequence conversion process automatically is even more difficult than manual conversion. The move from an abstract concept such as 'pSpac promoter' or 'gfp_CDS' to a sequence of nucleotide bases involves accessing and integrating theoretical and practical domain knowledge that must be captured and made available in a computationally amenable format.

This mapping is not one-to-one. A single model component may represent a whole set of biological concepts (e.g. there are numerous processes involved in protein degradation, which is generally modelled as a single entity), while one biological process may require multiple model components (protein production involves transcription, translation and mRNA degradation components). The need for expert domain knowledge to elucidate this mapping constrains the size, complexity and novelty of the systems that can be developed.

When the sequences are manually designed this knowledge comes from a human brain, aided by manual lookup of literature and databases. When the task is automated such knowledge must be incorporated into the model itself; the model requires metadata. Metadata is data about data that is information about the format, organization and meaning of the fundamental data, which in this case are the model components.

In this article, we present an algorithm for the automated conversion of an abstract computational model into a synthesizable DNA sequence. In order to perform this task, model components or species must be marked up with metadata. We therefore also propose a generic method to facilitate this model annotation, incorporating existing standards wherever possible. The algorithm presented is applicable to models constructed using either of the XML-based modelling languages CellML or SBML. We provide a use case that demonstrates our approach and describe a software application, MoSeC, which implements the model-to-sequence conversion algorithm.

1.1 Synthetic biology models and metadata

Computational modelling can be performed in one of two ways: bottom-up or top-down. Top-down modelling involves the breaking up of a high-level overview of a system into a number of modules, and then further reduction of the modules into submodules until all of the components of the system are specified at an elemental level. This approach is valuable for applications where the biological system under study already exists.

Bottom-up modelling, in contrast, involves constructing a model from well-defined, fine-grained components. In the context of synthetic biology, such components may be, for example, promoters, terminators and coding sequences (CDSs). At a higher level of abstraction, models can be constructed from entire devices with clearly defined and more complex functions. Recently, an approach to the definition of fine-grained model components, known as standard virtual parts (SVPs) has been developed. SVPs are virtual representations of components with clear biological counterparts (Cooling *et al.*, 2008, 2010).

There are a number of modelling formalisms in use, but two in particular are rapidly becoming standards in the systems and synthetic biology communities; SBML and CellML. SBML has been under development since 2003, and is widely used among the systems biology community. SBML was the first widely used machine-readable format for representing models, and offers a common intermediate format for models developed using different approaches and software.

CellML has been under development since 2000 and also aims to store and facilitate the exchange of computer-based mathematical models (Lloyd *et al.*, 2008). Recently, a CellML repository has been created to provide a repository of SVPs for the composition of synthetic biology models (Cooling *et al.*, 2010). CellML has a modular structure, in which equations and their variables are encapsulated inside components, while SBML uses explicit model elements to describe the reactions, their inputs, modifiers and outputs. Both of these modelling formalisms are encoded in XML. They have different strengths and weaknesses, making them complementary rather than competitive for synthetic biology. Both are popular enough that we consider it essential to provide model-to-sequence conversion for both formalisms, despite their somewhat different approaches to modelling systems of ordinary differential equations (ODEs).

1.2 Annotation of synthetic biology models

Perhaps the greatest challenge in automating the process of converting a dynamic model to the specification for a synthesizable DNA sequence is the availability of informative model metadata. To automate the model-to-sequence conversion process, information about entities and relationships must be represented in the model in a computationally amenable format.

Wherever possible this metadata should draw on existing standards, such as those laid down by the W3C, and employ defined terms that are backed by an ontology. For CellML and SBML models, appropriate metadata must be added both to the species or components and to their relationships.

Annotations are used to add metadata to models in order to incorporate computationally tractable information about their constituents. A number of approaches for the annotation of systems biology models have already been described and can be extended for use in synthetic biology (Endler *et al.*, 2009). For example, the MIRIAM project proposes a standard for the minimal information required to annotate systems biology models (Novere *et al.*, 2005). In this work, we annotate models in a similar fashion, embedding ontology terms, or links to ontology terms in the model as fragments of resource description framework (RDF) data. A number of examples of this approach to model annotation is presented in the Supplementary Material.

1.3 Bottom-up vs top-down model annotations

There are large repositories of models in both CellML and SBML. Our annotation approach can be applied to these existing models. However, there is considerable recent interest in the bottom-up composition of dynamic models using SVPs (Cooling *et al.*, 2010; Marchisio and Stelling, 2008). For bottom-up composition, it is both easier and more efficient to apply the annotation approach to individual SVPs. Therefore, for the work presented here, annotations are applied to virtual parts prior to model assembly, extending the approach defined by Cooling *et al.* (2010).

2 METHODS

A typical dynamic model contains entities such as species and reactions (in the case of SBML), or components and connections (in the case of CellML). The process of converting SBML or CellML dynamic models to a specification for a DNA sequence, in the form of a GenBank record for instance, requires that those constituents which represent sequence-based features such as coding sequences, ribosome binding sites and promoters, and the interactions between them, are distinguished from other elements of the model. These constituents must then be arranged in an appropriate order.

If the sequence construction process is to be carried out automatically then the information to guide the arrangement process must be derived from the model. The virtual parts used to build models are annotated with additional annotations that are not normally associated with synthetic biology model annotation. The addition of these annotations, while necessary to facilitate the automation of the model-to-sequence conversion may also be applicable to other model-based problems in the computational design of biological systems. These specific annotations and the issues they address are discussed below.

2.1 Genomic context

Both *cis* and *trans* relationships are an inherent part of any synthetic biology model. Ultimately, genetic features must be ordered and represented at a physical level in a DNA sequence of a genetic element (chromosome, plasmid, etc.). It is therefore necessary to consider the nature of the interactions between the elements in a model since the type of relationship may constrain the ordering of the physical representations of the model components. For example, *cis* interactions must be distinguished from non-*cis* since *cis* interactions require sequence-based features to be co-localized on a nucleotide sequence. The direction of a *cis* interaction between entities in a model can be used to derive ordering information, for example, specifying that a promoter should be placed upstream of a coding sequence. Conversely, some non-*cis* elements represent macromolecules or cellular components that operate in *trans*, and are not directly relevant to DNA sequence feature organization. A model also contains abstract elements, and connections between those elements, that allow the model to operate mathematically. These elements do not directly represent physical biological entities and are therefore neither *cis* nor *trans* in their behaviour.

In our annotation approach, SVPs contain a term indicating whether they are physical parts (possess a DNA sequence). Interactions between two physical parts are considered to be *cis* interactions.

2.2 Shims

A major confounding factor in both automated and manual model-to-sequence conversion is the need for the insertion of spacer DNA between components. Spacers may be necessary for biophysical reasons, for example to permit space for protein complexes to bind to the DNA. In this respect, they perform the same function as 'shims' in automotive engineering. The Compact Oxford English Dictionary defines a shim as 'a washer or thin strip of material used to align parts, make them fit or reduce wear'; a definition which appears entirely appropriate to spacer DNA in a biological construct. We therefore introduce the term shim as a spacer component for the assembly of genetic constructs for synthetic biology. It has long been recognized that shims are not always passive construction elements; mutations in the length and composition of these regions can dramatically affect the dynamics of the system under consideration (Grosschedl and Birnstiel, 1980), and hence the kinetic parameters of the corresponding model.

Shims do not function in isolation; their effect depends upon the components that they link, and model metadata must reflect this dependency. The precise length and sequence of shims is therefore likely to be of critical importance to the performance of a genetic system and therefore shims may have behaviour beyond that of simple inert spacer components. For example, a shim between a promoter and CDS can have a modifying effect on the transmission of transcriptional flux units from the promoter to the CDS. For these reasons we include shims as annotated virtual parts during the bottom up construction of the models. However, for the sake of simplicity, in the work presented here, we model shims as inert physical parts that possess a DNA sequence but do not alter the dynamics of the model.

2.3 Transcriptional and translational flux

Deriving a DNA specification from a model also requires that the fluxes between SVPs in a model are analysed. These fluxes reflect the way in which SVPs operate together to shape the transcriptional and translational behaviour of the system. The BioBricks Foundation has suggested that the standard way to measure the inputs and outputs of a BioBrick should be Polymerases per Second (PoPS). The equivalent metric for the translational activity of an mRNA molecule is Ribosomes per Second (RiPS). We have adopted these standards.

Flux analysis is complicated by a number of factors, which are taken into account by the model-to-sequence conversion algorithm. In a physical DNA sequence, a CDS upstream in an operon is likely to be exposed to a higher rate of transcription than those further downstream due to occasional premature termination or RNA processing. In our algorithm, we assume a uniform rate of transcription throughout an operon. The flow of information through parts of the graph corresponding to transcriptional units is therefore split after the promoter, flowing through several coding sequences in parallel. We also assume that terminators are 100% effective. While this assumption is valid for the majority of existing models, it would be possible to extend the algorithm to handle terminators with different degrees of read-through.

We add metadata to models to identify and describe the nature of these fluxes in order to direct the ordering of the genetic elements that ultimately encode the virtual system *in vitro* or *in vivo*.

2.4 Conversions of CellML and SBML models to graphs

In CellML models degradation, activation and production fluxes are connected through model components that act as a common interface. In SBML each flux is represented as an individual reaction, and these reactions do not need be explicitly connected in the model, since the species serve as a common interface to collect the fluxes.

Tracing the path of fluxes through the components of a model is difficult when the model is represented in XML. A more computationally amenable representation is a network, in which the nodes represent model components and the edges interactions between the components. A network representation also facilitates visual inspection of the model for completeness and topological consistency. Therefore, the first step in our approach to deriving the specification for a genetic system from a model is to convert the model to a graph-based representation.

Because of the differences between their XML representations, CellML and SBML models require slightly different approaches to convert the XML representation into a graph. Also, models of the same process in SBML or CellML may not produce identical graphs. After the XML to graph conversion step, however, the same algorithms can be applied to graphs derived from both SBML and CellML models.

2.5 An algorithm for graph to sequence conversion

The nodes of the graph that contribute to the final DNA sequence are first identified. Information on the order and spacing of sequence features must also be derived from the model. This information is inherent in the structure of the graph. The model-to-sequence algorithm is based upon an analysis of the flow of information through the model, as represented by its graph.

The first step is to identify the starting node that represents the beginning of the flow of information in the model: that component which will be placed first in the resulting DNA sequence. The starting component is identified via annotation, or it can be derived from the model by analyzing the network topology. Components which do not have an incoming edge from any other component are candidate starting points.

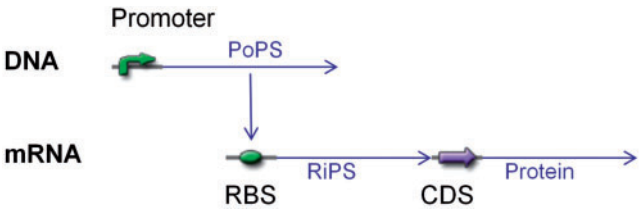

Fig. 1. Flow of information through the model at both the DNA and RNA/protein level in parallel.

To produce a linear DNA sequence specification, models are first turned into tree-like structures. This conversion is achieved by removing non-sequence-based components and any interactions forming loops that make it difficult to disentangle the flow of information through the system.

The analysis of information flow through the model is relatively straightforward for CellML, in which the representation of components as nodes and interactions as edges is inherent. SBML, however, is more challenging, and the model requires the following preprocessing:

- Assignment rules, rate rules, species and reactions are represented as nodes.
- If one assignment rule is used in another assignment rule, an edge is created between them.
- If a species is used in an assignment rule, an edge is created from the species to the assignment rule.
- If an SBML assignment rule is used in a reaction, an edge is created from one to the other.
- If a species is a reactant or modifier of a reaction, an edge is created from the species to the reaction.
- If a species is an output of a reaction, an edge is created from the reaction to the species.

The model-to-sequence converter algorithm involves the following steps, starting with either a CellML or an SBML file:

(1) Convert the XML representation to a graph representation.
(2) Identify nodes that correspond to DNA-based parts.
(3) Remove all non-*cis* interactions (interactions between RNA-based components count as *cis* interactions, and so are not removed).
(4) Identify the start node.
(5) Join the DNA-based parts in the direction specified by the edges. (The direction of flow of PoPS and the RiPS are derived from the relations of parts in the model.)
(6) Identify the subgraphs of physical components joined by mRNAs. Branches from an mRNA entity represent operon structures with more than one gene.
(7) Join the branches to form the operon structures. (At present, this is done by taking the left branch first.)
(8) Add terminators to the end of each transcriptional unit.
(9) Concatenate the sequences of the transcriptional units to form a linear DNA structure.

The algorithm uses the models built with SVPs to track the activity at the DNA level (PoPS) and at the RNA/protein level (RiPS) in parallel. In these models, PoPS are converted into RiPS through mRNA molecules (Fig. 1).

Table 1. Annotations required for DNA-based parts for the model-to-sequence conversion process

Attribute	Mandatory?
VisualName	No
IsDNABased	Yes
Type	If IsDNABased is TRUE
Sequence	If IsDNABased is TRUE and SequenceURI is empty
SequenceURI	If IsDNABased is TRUE and Sequence is empty
IsDNABasedPartTemplate	Yes
IsTemplate	No

3 RESULTS

3.1 Implementation

The model-to-sequence conversion algorithm has been implemented as an application called MoSeC. MoSeC is written in Java, and so will run on any platform.

MoSeC produces EMBL-Genbank formatted DNA sequences from both CellML and SBML models marked up with RDF according to the guidelines outlined above. MoSeC works optimally with models composed from sets of virtual parts in CellML or SBML in the format described by Cooling *et al*. (2010).

A standard set of annotations is required for our SVP annotation approach (Table 1).

The VisualName attribute holds the name that is used when the model is visualized in MoSeC. If this attribute is not present, the name assigned within the XML file is used. The IsDNABased attribute establishes whether an SVP is a physical part; that is, whether it has an associated DNA sequence. If this attribute is TRUE either Sequence or SequenceURI must have a value. The Sequence attribute holds the actual sequence information; if the sequence is long enough to be unwieldy, the SequenceURI may be used instead, to point to the URI at which the sequence data can be found. The Type attribute must also be set for DNA-based part. Part types currently supported are; Promoter, Shine_Dalgarno_Sequence, CDS, Shim, Signal_Carrier and Chassis. The final two attributes, IsTemplate and IsDNABasedPartTemplate, are used to indicate that a model is used as a template for SVPs. These attributes apply to CellML Version 1.1 only.

A step-by-step guide to the annotation of virtual parts and their relationships to support the automatic model-to-sequence conversion process is presented in the Supplementary Material.

3.2 Application

We applied our approach to the derivation of a DNA sequence for the subtilin receiver model described in detail by Cooling *et al.* (2010). The subtilin receiver model was built from annotated SVPs, assembled into a model and then used to generate a DNA sequence. The model was based upon that used to design a subtilin receiver device Biobrick by the 2008 Newcastle University iGEM team (part No. BBa_K104001) (Fig. 2). The device encodes a construct which responds to the lantibiotic subtilin by producing green fluorescent protein (GFP). The specification for the virtual parts used in the model was derived from the subtilin sensing and regulatory system

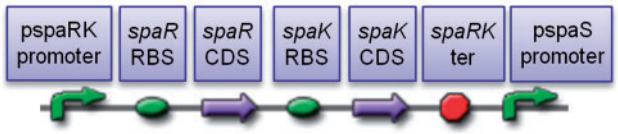

Fig. 2. The final subtilin receiver device BioBrick.

from *Bacillus subtilis* strain ATCC 6633, which uses subtilin for quorum sensing.

This construct contains a promoter that transcribes an operon containing the CDS for *spaR* and *spaK* that encode the response regulator and sensor kinase, respectively, of the subtilin two-component system. The SpaR regulated promoter, *PspaS*, is included downstream of the operon separated by a transcription terminator. Upon sensing the subtilin signal peptide, the SpaK subtilin sensor activates the SpaR regulatory protein by phosphorylation. The activated SpaR can then activate the *PspaS* promoter.

The subtilin receiver BioBrick was modelled in CellML Versions 1 and 1.1 and SBML Version 2 (Fig. 3). We annotated the virtual parts used to build this model according to the approach described above and used the model-to-sequence algorithm to automatically derive a DNA sequence specification. The essential steps of the model-to-sequence conversion algorithm are illustrated, using this model, in Figure 4. The annotated SVPs, the complete model and the output of the model-to-sequence conversion process are available in the Supplementary Material accompanying this article. The nucleotide sequence produced from this model is currently being biologically validated.

4 DISCUSSION

Many tools have already been developed to aid biologists with the manual design, simulation and construction of synthetic biological systems (Chandran *et al.*, 2009; Rodrigo *et al.*, 2007a, b). These tools are useful when the system under design is limited in size and the user possesses enough knowledge about the system to provide initial insights into the formulation of a preliminary design. However, manual, visually guided assembly of representations of sequence-based parts is unlikely to be scalable for large-scale computational design, especially where the detailed structure of the system to be designed is poorly understood.

Computational approaches, particularly those based on evolutionary computation (EC), are particularly promising for automating large-scale biological system design. Evolutionary algorithms were developed for applications in complex problem domains where the desired behaviour of the system is known, but the details of the system itself may be poorly understood. As such, it can produce novel, non-intuitive circuits (Macia and Sole, 2009). Indeed, there is increasing evidence that complex systems constructed manually are less robust than those constructed using an evolutionary approach (Yan *et al.*, 2010).

In order to have a completely automated circuit design process models must be constructed, their behaviour assessed via simulation, modified if necessary and then converted to a DNA sequence specification without the need for human intervention. However, the computational assembly of parts that directly represent the molecular entities that comprise biochemical systems is problematical due to

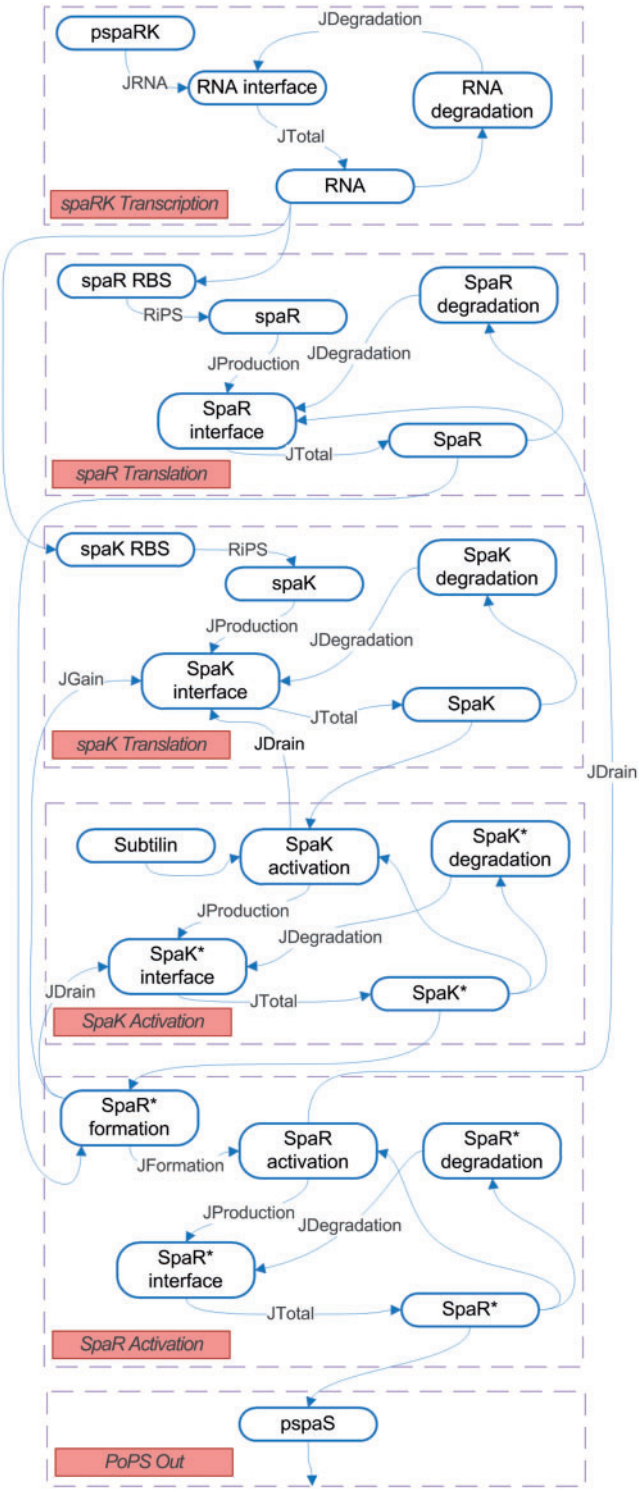

Fig. 3. An overview of the complete CellML model of the subtilin receiver BioBrick assembled from virtual parts.

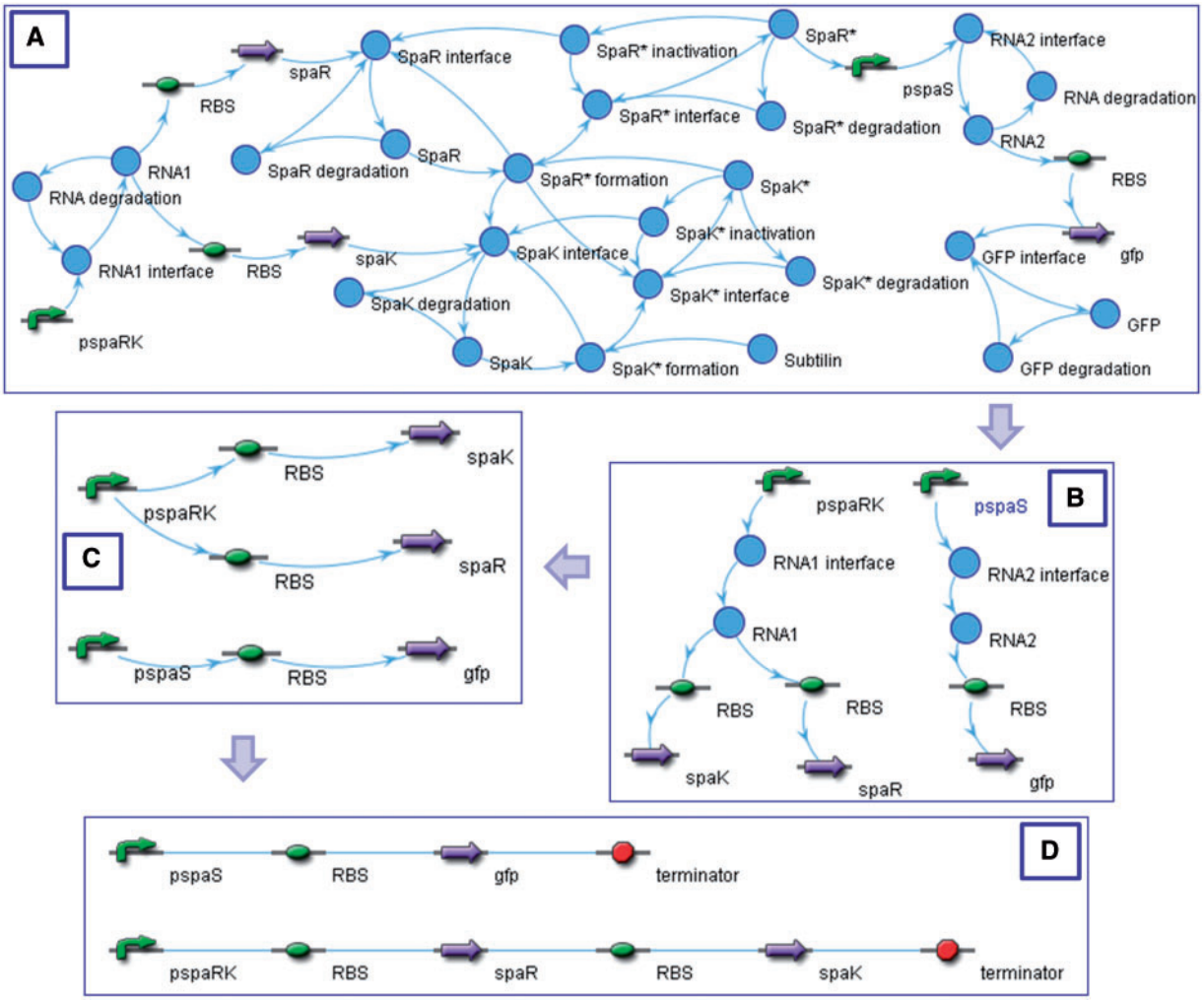

Fig. 4. The model-to-sequence converter algorithm applied to the subtilin receiver device model. (**A**) An XML file encoding the model is converted to a graph representation; (**B**) non-*cis* interactions between model species or components are removed, together with any entities that lie between these edges; (**C**) DNA-based parts only are retained; (**D**) the final sequence. The virtual parts used to build this model were annotated according to the approach outlined above, facilitating its automatic conversion into a DNA sequence specification.

the complexities of mapping DNA-based parts to the models that encode them.

Once a system has been successfully designed and simulated, the next logical step is to test the design by implementing it in the target chassis or organism. The nucleic acid sequences required to encode the system *in vivo* must be specified and ordered to give the correct genomic context, a process that is guided by information in the model. In other words, the model representing the system must be interpreted to identify the nucleic acid molecules necessary to implement the system *in vitro* or *in vivo*.

In order to automatically specify the genetic system at a sequence level, two types of information must be present in the model: information specifying the nucleic acid sequences necessary to encode the system at a genetic level; and connectivity information that allows the correct ordering and spacing of sequences of genetic parts to be interpreted to ensure the correct *cis*-based interactions between these genetic parts. Currently, dynamic models generated for systems or synthetic biology in SBML and CellML, are not usually annotated with this information. In this article, we have described an approach to SBML and CellML model annotation that allows the information to automatically specify genetic systems at the nucleotide sequence level to be derived.

CellML and SBML have proven pedigrees in systems biology, and many complete models of synthetic systems already exist. It is possible to manually add the annotations necessary for the model-to-sequence conversion process after model composition. However, our annotation system has been designed with a bottom-up model assembly process in mind. By marking up the virtual parts prior to model composition, composite models may be directly converted to nucleotide sequences with no further manual intervention. It is our intention to provide repositories of SVPs already marked up using the MoSeC specification as a standard, to support design tools that operate in a bottom up fashion.

Typically, the annotation of virtual parts is done manually by reference to bioinformatics databases. Although this is a time-consuming step, the fact that SVPs are, by definition, re-usable (and CellML has templates that can be used to instantiate SVPs) mean that it is a once-off investment for each part. The practical requirements for annotation reinforces the benefits of using modular parts, rather than annotation of entire models, and of maintaining repositories of already-annotated SVPs. However, in the future it may be possible to automate the annotation process itself through data integration strategies that combine information from remote data sources, such as genome or transcription factor databases, with experimentally derived information about a particular part (Lister *et al.*, 2009).

The MoSeC system as described is rich enough to produce the specifications for complete sequences suitable for synthesis. However, there are further improvements that could be incorporated into the model-to-sequence conversion process. We do not currently include information about the physical location of the final sequence in the genome of the intended chassis organism. Furthermore, sequences could be codon optimized for the intended host organism as part of the conversion process.

Synthetic biology is rapidly becoming an iterative, computationally dependent discipline, in which the outputs of models and *in vivo* implementations feed back to each other to continually improve our ability to understand and predict the behaviour of synthetic genetic systems. The annotation of designs for synthetic systems will play a critical role in the automation of the bio-design lifecycle. The use of annotations in automating model-to-sequence conversion is a small but vital step towards the computational design of useful, predictable synthetic genetic systems.

ACKNOWLEDGEMENTS

We acknowledge Poul F. Nielsen for helpful discussions relating to this work.

Funding: Research Councils UK (to J.S.H.); Engineering and Physical Sciences Research Council/National Science Foundation grant number: EP/H019162/1 (to G.M.); University of Auckland Faculty Research Development Fund New Staff grant (to M.T.C.).

Conflict of Interest: none declared.

REFERENCES

Anderson,J.C. *et al.* (2006) Environmentally controlled invasion of cancer cells by engineered bacteria. *J. Mol. Biol.*, **355**, 619–627.

Andrianantoandro,E. *et al.* (2006) Synthetic biology: new engineering rules for an emerging discipline. *Mol. Syst. Biol.*, **2**, 2006.0028.

Bolouri,H. and Davidson,E.H. (2002) Modeling transcriptional regulatory networks. *BioEssays*, **24**, 1118–1129.

Chandran,D. *et al.* (2009) Athena: modular CAM/CAD software for synthetic biology. arXiv:0902.2598v1 [q-bio.QM]. Downloaded 7 May 2010.

Cooling,M.T. *et al.* (2008) Modelling biological modularity with CellML. *IET Syst. Biol.*, **2**, 73–79.

Cooling,M.T. *et al*. (2010) Standard virtual biological parts: a repository of modular modeling components for synthetic biology. *Bioinformatics*, **26**, 925–931.

Cuellar,A.A. *et al*. (2003) An Overview of CellML 1.1, a Biological Model Description Language. *SIMULATION*, **79**, 740–747.

Endler,L. *et al*. (2009) Designing and encoding models for synthetic biology. *J. R. Soc. Interface*, **6**(Suppl. 4), S405–S417.

Goldbeter,A. (2002) Computational approaches to cellular rhythms. *Nature*, **420**, 238–245.

Grosschedl,R. and Birnstiel,M.L. (1980) Spacer DNA sequences upstream of the T-A-T-A-A-A-T-A sequence are essential for promotion of H2A histone gene transcription *in vivo*. *Proc. Natl Acad. Sci. USA*, **77**, 7102–7106.

Hasty,J. *et al*. (2002) Synthetic gene network for entraining and amplifying cellular oscillations. *Phys. Rev. Lett.*, **88**, 148101.

Hucka,M. *et al*. (2003) The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, **19**, 524–531.

Knight,T. (2003) Idempotent vector design for standard assembly of biobricks. *Synthetic Biology Working Group Technical Reports*, MIT. Available at http://dspace.mit.edu/handle/1721.1/21168 (last accessed date February 15, 2011).

Lee,S.K. *et al.* (2008) Metabolic engineering of microorganisms for biofuels production: from bugs to synthetic biology to fuels. *Curr. Opin. Biotechnol.*, **19**, 556–563.

Lister,A.L. *et al.* (2009) Saint: a lightweight integration environment for model annotation. *Bioinformatics*, **25**, 3026–3027.

Lloyd,C.M. *et al.* (2008) The CellML model repository. *Bioinformatics*, **24**, 2122–2123.

Macia,J. and Sole,R.V. (2009) Distributed robustness in cellular networks: insights from synthetic evolved circuits. *J. R. Soc. Interface*, **6**, 393–400.

Marchisio,M.A. and Stelling,J. (2008) Computational design of synthetic gene circuits with composable parts. *Bioinformatics*, **24**, 1903–1910.

Novere,N.L. *et al*. (2005) Minimum information requested in the annotation of biochemical models (MIRIAM). *Nat. Biotechnol.*, **23**, 1509–1515.

Ro,D.-K. *et al*. (2006) Production of the antimalarial drug precursor artemisinic acid in engineered yeast. *Nature*, **440**, 940–943.

Rodrigo,G. *et al.* (2007a) Automatic model-based design of genetic circuits. In *Proceedings of the 2007 Conference on Machine Learning in Systems Biology*, Evry, France.

Rodrigo,G. *et al.* (2007b) Genetdes: automatic design of transcriptional networks. *Bioinformatics*, **23**, 1857–1858.

Sinha,J. *et al.* (2010) Reprogramming bacteria to seek and destroy an herbicide. *Nat. Chem. Biol.*, **6**, 464–470.

Weiss,R. *et al*. (2003) Genetic circuit building blocks for cellular computation, communications, and signal processing. *Nat. Comput.*, **2**, 47–84.

Yan,K.-K. *et al.* (2010) Comparing genomes to computer operating systems in terms of the topology and evolution of their regulatory control networks. *Proc. Natl Acad. Sci. USA*, **107**, 9186–9191.