

Biological Layer Abstraction and Standards Hierarchy v.8

Austin Che

This is a work in progress

Please send comments to austin@csail.mit.edu

April 22, 2007

1 Introduction

Synthetic biology distinguishes itself from genetic engineering in its attempt to bring a more structured approach in engineering large scale biological systems [13,15]. Early examples of engineered biological systems such as a ring oscillator [10] and toggle switch [16] showed the power of rational design applied to specific problems. However, an abstraction framework for engineering reusable modules and constructing more complex systems is needed.

Key concepts from engineering complex systems include modularity, abstraction, and standards. *Modularity* allows breaking large problems into manageable modules. The “black box” module performing some self-contained function is a common idea in engineering. Good modules have well-defined inputs and outputs with the outputs predictably dependent only on the inputs. Modularity leads to two important ideas that will be explored in detail: abstraction and standards.

Useful modules have well-defined *abstraction* boundaries. Effective abstraction relies on defining input and output interfaces for connecting modules together. We can nest abstractions, to form an *abstraction hierarchy*. An abstraction hierarchy allows building modules at different levels of complexity. Abstraction layers facilitate working at one level of complexity without worrying about unnecessary details at other levels, provided there are well-defined and useful interfaces between abstraction layers.

Figure 1 shows modules connected through interfaces. Modules within an abstraction hierarchy have

several important interfaces. There are interfaces at a module boundaries for its inputs and outputs, represented in the diagram by the vertical interface boxes. These interfaces connect modules at one abstraction level. In addition, there are interfaces for allowing modules to be nested inside other modules in the hierarchy. These interfaces, represented by the horizontal boxes in the diagram, allow for working at different abstraction levels.

Connecting modules together requires them to have compatible interfaces. Thus, limiting the possible types of interfaces increases the probability that two modules have compatible interfaces and are composable. *Standards* are critical for allowing the interoperability of different modules that conform to the same standard interfaces. Combining standards with an abstraction hierarchy allows us to have a *standards hierarchy* where standards at different levels of complexity work together to simplify large-scale engineering.

1.1 Biological Layer Model

The biological layer model is an abstraction and standards hierarchy for engineering complex biological systems at different levels of detail. The model is inspired by the standard ISO computer network model [17]. Layers are stacked on top of one another, with the higher layers building upon the functionality of the lower layers. Table 1 summarizes the layers in the biological layer model.

The biological layer model provides a conceptual framework for specifying abstraction interfaces for

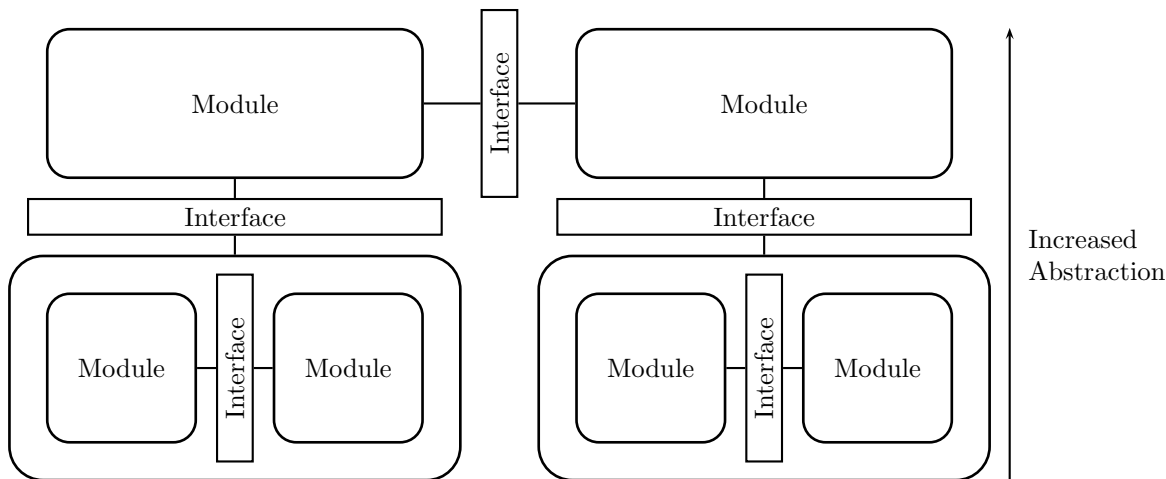


Figure 1: Modules can be treated as black boxes connected together through well-defined interfaces. Standardizing the interfaces allows easily interconnecting compatible modules and working at different abstraction levels.

Layer	Example
Layer 7: Application	cell-cycle counter
<i>Application Interlayer</i>	safety; stability; noise
Layer 6: Environment	media; system I/O
<i>Environment Interlayer</i>	innoculating media
Layer 5: Chassis	cell strain load capacity
<i>Chassis Interlayer</i>	orthogonal expression
Layer 4: Cell	cell-cell signaling
<i>Cell Interlayer</i>	export tag
Layer 3: Protein	dimerization interface
<i>Protein Interlayer</i>	RiPS
Layer 2: RNA	ribozyme
<i>RNA Interlayer</i>	PoPS [12]
Layer 1: DNA	BioBricks [18]
<i>DNA Interlayer</i>	DNA polymerase
Layer 0: Materials	nucleotides/amino acids

Table 1: The eight layers in the biological layer model are stacked with the application layer at the top and the materials layer at the bottom. The interlayers between adjacent layers connect the layers up and down the hierarchy. Module interfaces and corresponding standards can be defined at both the layers and the interlayers.

modules. Defining interfaces enables creating standards for those interfaces, and standard interfaces allow engineered biological modules to be joined seamlessly together. The layers in the biological layer model were selected to have a well-defined function and to be useful as an abstraction interface for modules. In addition to the layers are the *interlayers* between adjacent layers. The interlayers also form abstraction interfaces allowing layers to interface with higher and lower layers. By standardizing the interlayer interfaces, higher layer modules can be constructed from simpler modules.

1.1.1 Example Module

Figure 2(a) shows a “protein detector” module with one input interface taking in protein A and one output interface producing a fluorescent protein. This protein detector module is connected to an “A-generator” module. The biological layer model identifies the types of standards that are appropriate for these interfaces. The output interface of the A-generator module and the input interface of the protein detector module must be compatible layer 3 protein interfaces. In addition, the output interface of the protein detector module is a green fluorescent protein. Standards needed for this output interface include excitation and emission wavelengths for measuring the fluorescence of GFP.

In the ISO network model, communication between two parties at the same layer is mediated by proceeding down the layer stack on one side of the communication channel and then back up the layer stack for the other party. For example, suppose two network applications are communicating with each other. The high level application layer on the sender side utilizes lower level layers like the network layers by passing information down. At the lowest physical layer, bits are sent along some medium which is directly connected to the other party. Then the information is passed through the layers back up to the application layer of the receiver.

Similarly, communication in the biological layer model flows through the layer stack. Figure 2(b) shows a mapping of the example protein detector network into the layer model. The output of the A-

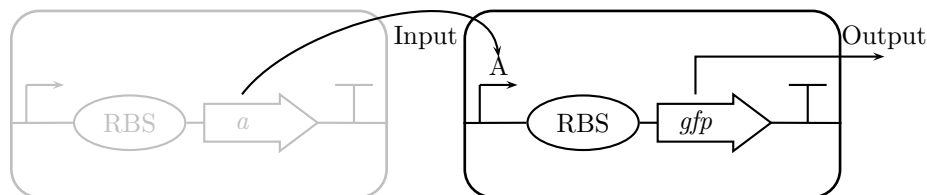
generator module enters as an input signal to the protein detector at the protein layer. The input protein A interacts at the RNA interlayer which depends on layer 1 DNA. The layer 1 DNA depends on materials in layer 0. The materials layer of the two modules are physically identical. Just as there is a common physical link to connect two parties in the computer network model, e.g. a wire, the materials layer can form a common physical connection in the biological layer model. The signal is abstractly carried by the physical materials from one module to another. Thus, communication at higher layers is mediated by going down the layer hierarchy to the common materials layer and back up the layer stack.

2 Standards

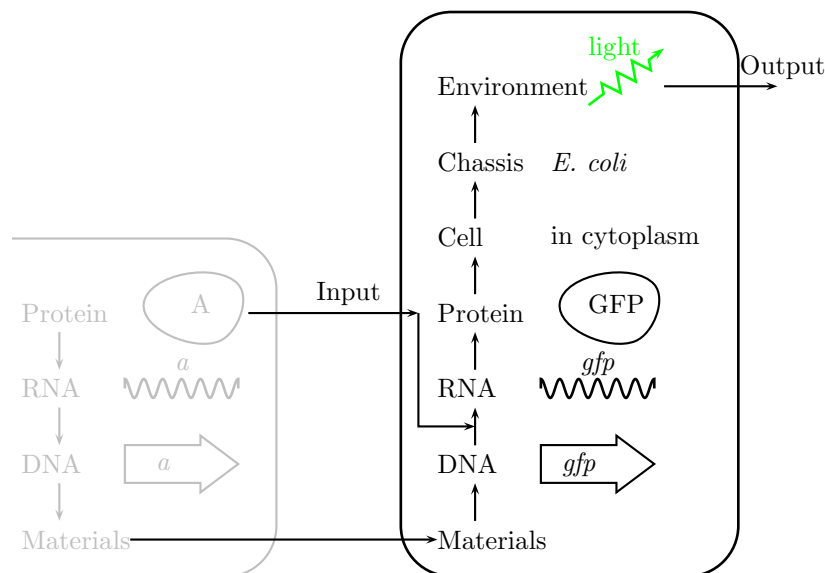
Although we could engineer biological modules that follow the biological layer model without conforming to standards, non-standard modules cannot take advantage of the power of modularity in both allowing module reuse and for connecting modules from disparate sources together. Some example types of standards will be provided at each layer and interlayer to motivate the development of standards in support of engineering complex biological systems. Anything can be standardized, but ideal standards *should have* the following properties:

- The standard allows for interconnections among all modules conforming to the standard.
- The standard should be module and implementation independent. For example, a standard that specifies the use of a particular variant of a specific protein is not as useful as a standard that requires a certain protein function, which could be satisfied by many proteins.

There are two types of module connections and corresponding standards for those connections. One way modules can be connected is via a *functional* connection whereby modules’ functions are combined. A functional standard provides an interface for modules to inter-operate. One common functional standard



(a) The protein detector module is shown connected to a module producing protein A.



(b) The above network structure is mapped to the biological layer model.

Figure 2: The protein detector module (on the right) has one input (protein A) and one output. The output, green fluorescent protein (GFP), is produced in proportion to the amount of the input protein.

for information processing applications is the specification of a signal carrier. For example, in boolean logic, the unit of information is a *bit*, which can have one of two values. Different implementations of boolean logic can have different standards for how the bit is represented as a signal. In digital electronics, the common signal is often voltage. Compatible electronic devices all use the same signal carrier with the same mapping from a voltage to a bit. Compatible signals allows devices, whether electronic or biological, to work in combination.

A second type of standard is the *structural* standard allowing modules to be physically connected together. For example, nut and bolt standards are

structural standards that allow nuts and bolts to work together. Similarly, biological structural standards facilitate physically connecting biological modules together. Whereas functional standards allow modules to communicate with each other by themselves, structural standards may require external materials, tools, and user work, such as enzymes or cloning.

2.1 Terminology

To provide a common terminology, we define our use of the words “parts,” “devices,” and “systems.”

Part: *A part is anything that conforms to a structural standard.* For example, a cell strain, a plasmid, or a piece of DNA can all be parts. Parts include all physical objects that can be used in building a biological system. Things which do not conform to a structural standard, similar to an oddly shaped nut or bolt, will not be considered a part as it is unusable in a meaningful way.

Device: *A device is a part that also conforms to functional standards for all inputs and outputs.* All devices are also parts, as devices need to be in a manipulatable form to be useful. Many devices will also contain other parts or devices. Unlike a part, a device is defined by its usefulness as a functional module and its ability to be combined with other devices. The modularity of a device stems from an abstraction layer imposed on an existing part. The defining attribute of a device, the specification of its input and output interfaces, is what allows a device to be used in a modular manner. Devices with identical input and output interfaces form a family of compatible devices. Furthermore, we expect the device interfaces to be characterized or at least be capable of being characterized. Note that all layers have inter-layer interfaces up and down the hierarchy. For example, a RNA layer always interfaces with a DNA layer. However, when discussing device interfaces, we are always referring to inter-module interfaces. Thus, a device with a RNA input interface could have transcription as input.

System: *A system is a device with application layer standards.* Thus, a system is constructed for a specific purpose by a designer. This is intentionally a slightly vague definition to match its common usage. In some fields, a system is defined as something that has an input and output and does some processing. However, to be more consistent with existing usage, we use “device” to refer to this meaning.

2.1.1 Examples

Here are some examples of parts, devices, and systems:

- A random piece of DNA, with no notion of input or output, is not a device. However, if it conforms to a structural standard (perhaps its sequence is known and can be synthesized), it is a part which can be used in building other parts.
- A protein scaffold designed to allow other proteins to be inserted by following some structural standard is a part.
- A constitutive promoter, with zero inputs and one output at the RNA layer, and a transcriptional terminator, with one input at the RNA layer and no outputs, are devices.
- A cell that responds to a cell-cell signaling molecule from another cell by emitting light has a defined input at the cell layer and an output at the I/O layer, so it is a device. If this same cell is used for a specific application purpose, such as for detecting a high concentration of the signalling molecule, then the cell with the environment it operates in is a system.
- A cell strain counts the number of divisions it has and kills itself after 5 doublings. This is the specification of a system at the application level.

To summarize:

- A part is the *physical* manifestation that allows constructing and manipulating components.
- A device is the *conceptual* view of a part that allows thinking about functional operations.
- A system is the *specification* that allows communicating a designer’s needs.

3 Layers and Interlayers

Although not every biological system needs to have all the layers in the biological layer model and some systems may not fit cleanly into this framework, the layer model provides a useful abstraction of biological complexity, by focusing on single slices of the complexity at a time. The following sections describe the

layers and interlayers in detail. In addition, both layers and interlayers can be part of module interfaces.

Standards are needed in all areas of the layer hierarchy. Layer standards describe common functions within a layer. Interlayer standards provide standard methods for connecting the layers. Finally, standard module interfaces are critical for connecting modules together. We provide current examples of possible functional or structural standards within the layer hierarchy. However, the choice of standards is not meant to be exhaustive. They merely serve to provide motivating examples of using the layer model as a standards hierarchy.

3.1 Layer 0: Materials

Description The materials layer specifies the basic biochemicals used in the cellular environment. This includes molecules such as nucleotides and any other small molecules that are produced or used by a module.

Standards Structural standards at the materials layer define such things as the nucleotides and amino acids available for building parts. There is an obvious existing standard given to us by nature: the use of the four nucleotides found in DNA (ATGC), the four nucleotides found in RNA (AUGC), and the 20 amino acids used during translation. However, synthetic systems do not necessarily need to continue to use nature's standard. There is existing work on increasing the number of nucleotides, using synthetic nucleic acids and increasing the number of amino acids [1, 3, 5, 6, 19, 20, 25], making it feasible to define a standard using non-naturally occurring materials. However, in practice, it is likely that the standard molecules from nature will be used for the foreseeable future.

Interfaces Module interfaces at this layer rely primarily on molecule diffusion. For example, one module could output a metabolite. The metabolite diffuses to the input of another module that converts the metabolite to another molecule. Functional standards could specify concentrations and thresholds of molecules.

3.2 Interlayer: DNA

Description The physical materials determines the DNA nucleotides available for building DNA. This interlayer, being at the lowest level, has the greatest impact in determining part compatibility. Parts using one set of materials is unlikely to be compatible with parts using a different set of materials.

Standards The standard method for making DNA from nucleotides are the natural DNA polymerases. DNA polymerases that can insert synthetic nucleotides have also been developed.

Interfaces A module interface here controls the making of DNA, perhaps by regulating the expression of a polymerase. One example is a module that produced a transposase as an output. Another module could use this transposase as an input to incorporate new DNA into the module.

3.3 Layer 1: DNA

Description The DNA layer specifies how DNA is made and processed.

Standards The DNA standard is given by its structure: the 2'-deoxyribose sugar, the phosphodiester backbone, and the nitrogenous base. As the layer is defined based on this natural standard, it is unlikely a significantly different standard can be used.

Interfaces Structural standards for DNA assembly fall into this layer. Two examples of DNA assembly standards are the BioBricks [18, 23] and BioBricks++ [8] methods. These type of schemes have a similar goal: any part that conforms to one of these standards can be assembled with any other similarly conforming part. The standards are similar in specifying a common sequence with several restriction sites in the front and back of a part, allowing assembling any two parts using restriction enzymes and DNA ligase. Figure 3 shows how defining a structural standard allows a module to be composed from smaller modules.

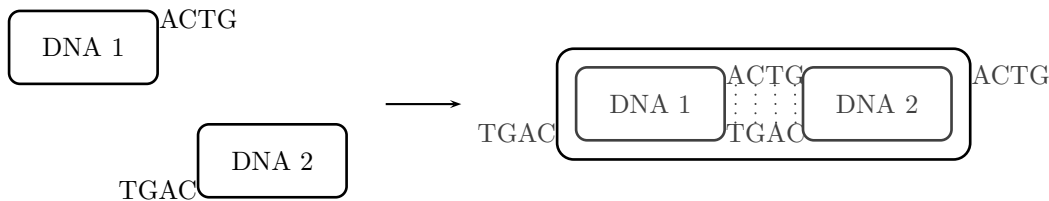


Figure 3: The basic idea of standard idempotent DNA assembly is to be able to use standard restriction enzymes to generate common “sticky ends” on all modules. The compatible sticky ends allows any two modules to be joined together. In addition, the combined module will be of the same form as the parent modules, allowing assembly to continue sequentially.

When *de novo* DNA synthesis makes it economically feasible to synthesize any piece of DNA, these assembly standards may no longer be necessary. Any sequence of DNA would be compatible with any other DNA and could be joined together using a DNA synthesizer that synthesized the two parts together. This is an example of an ideal universal standard, with all DNA sequences automatically conforming to the standard.

DNA functional interfaces could also exist, but they are more difficult to think up and use in a modular fashion. For example, a DNA sequence could be used as the functional target of some enzyme. However, using specific DNA sequences is not device or implementation independent. In addition, natural systems tend to not to use DNA for active components and instead rely on higher layers.

3.4 Interlayer: RNA

Description The RNA interlayer connects the DNA and RNA layers.

Standards Transcription of DNA into RNA by RNA polymerase is the standard DNA to RNA interface found in nature. The process of transcription also specifies the mapping of deoxynucleotides to ribonucleotides.

Interfaces One example of a functional standard at the RNA interlayer is PoPS (polymerases per second), defined as the number of RNA polymerases transcribing past a given point on a piece of DNA per

second [9,12]. As PoPS appears to be a generally useful idea, we will examine it in more detail using a simple device as an example. An inverter device, or NOT gate, receives an input signal (e.g. HIGH) and produces the opposite output signal (e.g. LOW). A biological inverter can be made from a repressor protein that binds a DNA regulatory region around the promoter. Inversion occurs because the repressor:DNA complex precludes transcription of the DNA by RNA polymerase. A high concentration of repressor protein inhibits transcription and a low concentration of repressor allows transcription.

A family of repressor-based inverters can be built by combining four types of components: repressor proteins, promoters and regulatory sequences that those repressors bind, ribosome binding sites (RBS) to initiate translation, and transcriptional terminators. Figure 4 shows part of an inverter cascade. The four models are all physically the same but the module boundaries are chosen differently. Figure 5 shows how the choice of module boundaries lead to different input and output signals.

In the first model, the classic operon model, the device is structured in the standard biological view of an operon: a transcriptional regulatory region drives transcription of a RBS, a repressor coding sequence, and ends with a transcriptional terminator. The primary input to this device is an input repressor protein that regulates the module’s promoter. The primary output is expression of the output repressor protein. When the input repressor is present, the output protein is not expressed, and vice-versa, thus implementing an inverter using protein levels as signals. A sec-

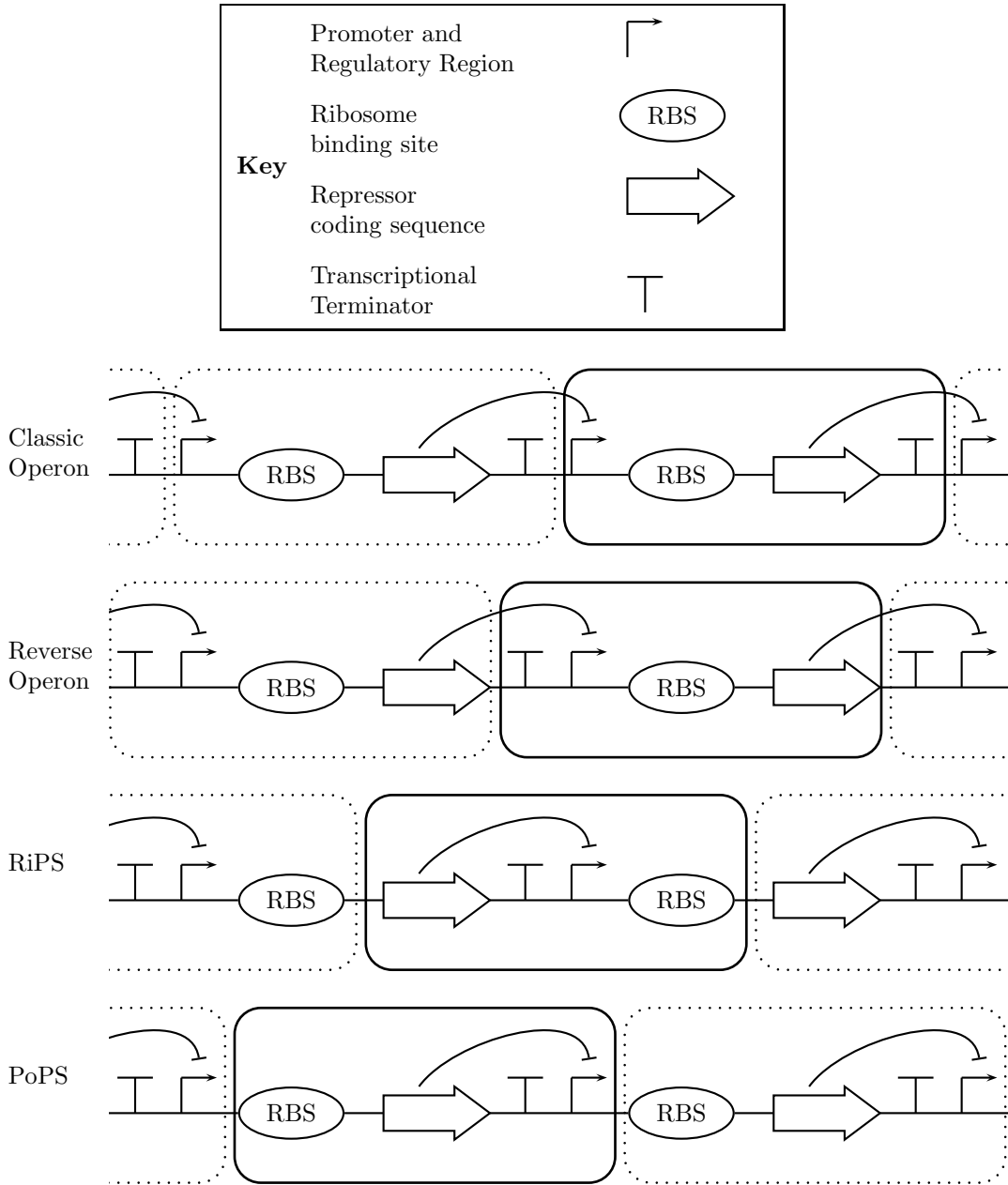


Figure 4: The same physical system based on cascaded protein repressor inverters can be represented in four different ways depending on where the abstract module boundaries are set. One complete module is highlighted in each case.

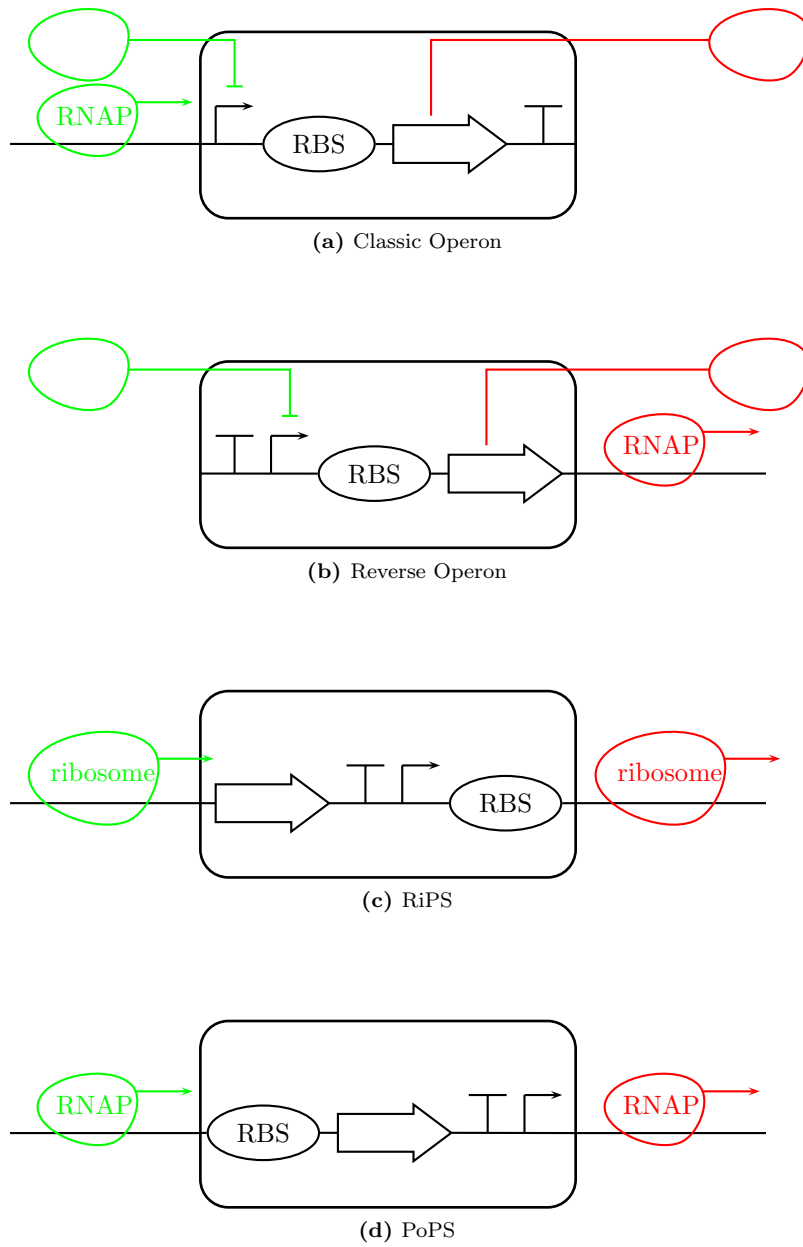


Figure 5: The choice of module boundaries in the four models lead to different input and output signals. The inputs come in from the left into the module and are shown in green. The outputs are shown in red leaving the module on the right. RNAP represents RNA polymerase in the process of transcribing DNA.

ondary input signal for the operon module is an often neglected “hidden” transcription into the device. If there’s leaky transcription from upstream of the device, then it can enter the module and impact the module’s operation.

The reverse operon model is similar to the classic model except that the terminator at the end has been placed at the beginning. It is also a protein inverter with its primary input and output being the repressor proteins. Instead of having transcription potentially leaking into the module, transcription can now leak out. For our purposes, this model can be treated identically to the classic operon model, and, so, we will lump them together and refer to them as the operon models.

When we consider an existing system like the cascaded inverters of Figure 4, the choice between the operon model or one of the other models is not too important as the same physical system results. However, if we have a collection of modules and need to design and build a system with those modules, then operon-based devices have a significant drawback. Putting aside the relatively minor drawback of having probably undesired extra transcription input and output signals, the major drawback is that the inputs and outputs for operon-type devices are non-universal proteins specific for the device.

For example, in Figure 6(a), we have three operon-type device modules with different proteins for inputs and outputs. If we wish to connect these modules to form an inverter cascade, there is only one possible way to join them together. The module which contains the input regulatory region A must come after the module which has the corresponding output protein A , and a similar restriction holds for joining the third module. The inherent problem in the modular design of these devices is that the inputs and outputs are all in units of different proteins. A device requires as its input the protein that regulates its promoter and similarly, the output is whatever protein is made from the coding sequence that the device contains.

We could try to make the modules generic by standardizing the input and output protein by having all modules take protein A as input and make protein B as output. This would allow each module in this collection to substitute for any other module, which is

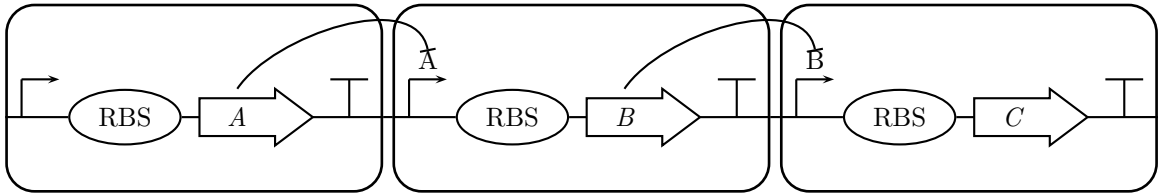
good, but would not allow connection of these modules together. To be able to arbitrarily compose devices together, the input and output must be the *same* so that the output of any module is compatible with the input of any other module. If we make the input and output the same protein as shown in Figure 6(b), we *are able* to connect an arbitrary number of modules together. However, all the modules end up being identical and due to the module crosstalk resulting from the modules having the same inputs and outputs, the connections would be mostly meaningless. Thus, such a device family is useless in designing biological systems.

Using mRNA species as signals instead of proteins with the same operon-like devices has the identical problem of having different inputs and outputs for each module. Using protein or mRNA levels as signals is convenient because both are familiar and methods exist for measuring the levels of mRNA and proteins in cells and cell extracts. However, in building up biological systems from individual modules, the natural operon models are not sufficient.

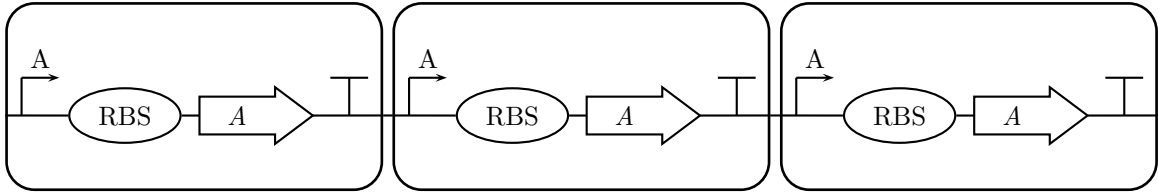
We can solve this problem of device-device connectivity by choosing different device boundaries. Both of the remaining two models, the RiPS (ribosomes per second) and PoPS (RNA polymerases per second) models, are sufficient to allow generic device interconnections. In this section, we will focus on the PoPS model and we will discuss RiPS in the protein layer section.

The generic PoPS device shown in Figure 5(d) consists of a RBS, the repressor coding sequence, the terminator, followed by the region regulated by the device repressor. Both the input and output to the device is the number of RNA polymerases per second flowing into or out of the device. Unlike the operon models, these transcription events are now a desired signal, rather than an unwanted side-effect. When input transcription is high (high PoPS), the RBS and repressor is transcribed and translated which leads to repression of output transcription (low PoPS). Similarly, low input PoPS leads to high output PoPS. Thus, the PoPS devices are inverters at the RNA level.

Figure 7 shows three different PoPS devices. Although these PoPS modules use the same number and



(a) These three device modules, structured in the operon form, can only be connected together in the one way shown here. Other connections are not valid as each module has different proteins for its input and output. For example, there is no way to directly connect the leftmost module with the rightmost module as the output of one is the protein *A* and the input of the other is the protein *B*.



(b) Standardizing on the input and output does not work for these modules. If the input and output are all standardized to be protein *A*, then all modules become identical, and are of little use in building multi-device systems.

Figure 6: Operon-like modules cannot be connected in a modular fashion. Note that there are alternate physical ways of joining these modules together, but in each of these alternates, the signal connections will be identical to that shown here, forming the same device network.

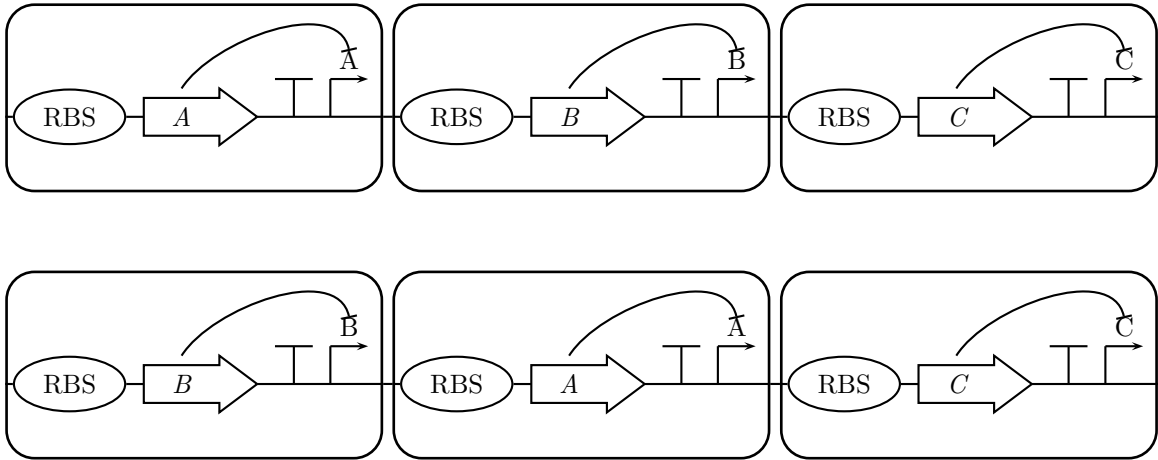


Figure 7: Unlike the modules in Figure 6, the module connections represent both a physical and abstract connection between modules. Each of the ways of joining these 3 modules to each other represents a different device network. 2 of the possible 6 ways of joining these 3 modules are shown.

type of components as used in the operon devices in Figure 6(a), the PoPS modules can be arbitrary composed together as all inputs and outputs are in identical units. This huge advantage over the operon model comes from the fact that each device encapsulates the repressor and the region it regulates inside the device, making it independent of other modules. Note that we are ignoring potential issues of undesired crosstalk with other modules and related problems here.

Unlike in the operon models, the physical connections between PoPS devices are meaningful. By making the connection meaningful, we limit ourselves in our physical construction (two modules cannot be connected on separate plasmids), but it also allows arbitrary module complexity as the module connection is *isolated by the physical connection*. Whereas in the operon model, there was unwanted crosstalk if we made all device inputs and outputs the same protein, in the PoPS model, there is no such crosstalk. The RNA polymerase transcribing signal, unlike a floating protein, has the polymerase on a piece of DNA going in one direction. The DNA acts as a “wire” ensuring the RNA polymerase only goes to the next module.

PoPS is perhaps one of the most useful general signals for several reasons. First, the signal allows functional module connections at the lowest common layer that supports functional standards well. As mentioned previously, modular functional standards for the lower DNA and the materials layer do not really exist. DNA and materials are structurally useful but are less useful for functionally joining devices together. Functional standards for layers above the RNA layer could also be useful, but are less general, as some devices may not use those layers. As transcription is the basis for all higher layers, PoPS is potentially the most widely-usable common signal carrier.

Importantly, PoPS is device-independent. The flow of RNA polymerases along a DNA specifies nothing about the particular sequence of DNA. For the operon-devices, when we standardized on a single protein for input and output, all the devices became identical. In PoPS-devices, many different devices can exist with a standard PoPS input and output. RNA polymerases is a general information carrier in

the same way that the flow of electrons carries information in the electrical concept of “current.” Using RNA as a signal carrier also meshes well with the natural role of RNA as an intermediate in gene expression. The process of transcription can be highly regulated and controlled. In contrast, if we imagine using “DNA polymerases per second,” the process of replication by DNA polymerase is not known to be as controllable or manipulatable and hence not as useful as a potential signal carrier.

3.5 Layer 2: RNA

Description The RNA layer contains RNA parts.

Standards The RNA standard is given by its structure: the ribose sugar, the phosphodiester backbone, and the nitrogenous base. As the layer is defined based on this natural standard, it is unlikely a significantly different standard can be used.

Interfaces Modules with RNA layer interfaces include RNA processing enzymes and ribozymes. For example, a module could have an output of a RNA ribozyme that targets and cleaves a certain RNA sequence. This ribozyme can then be used as the input for another module.

3.6 Interlayer: Protein

Description This interlayer produces proteins from RNA.

Standards The standard RNA-protein interface found in nature is translation of RNA into protein using the universal genetic code. However, modified genetic codes could also serve as alternate standards as we have the capability of engineering the genetic code either for inserting new amino acids or to reshuffle codons using existing amino acids [1, 6, 20]. Other relevant interface parameters include the speed and location of translation activity. For example, the ribosome binding site (RBS) that determines the translation initiation point can be changed [24].

Interfaces Translational devices that depend on regulation of translation fall into this interlayer. Analogous to PoPS for the RNA interlayer, RiPS (ribosomes per second) can be defined as the rate of ribosomes translating at a particular point of RNA. RiPS has many of the same advantages of PoPS, in particular being a device-independent functional signal. Ribosome flow is independent of the particular RNA being translated or the protein being made.

3.7 Layer 3: Protein

Description The protein layer defines the use of proteins in modules.

Standards The protein layer is defined by the standard peptide bond and general amino acid structure.

Interfaces Protein interaction parts and devices fall use this layer. Just as proteins are an extremely diverse group of molecules in nature, a large number of potential protein standards could exist. Protein structural standards could include protein dimerization interfaces that allow compatible proteins to dimerize and protein scaffold standards that allow protein pieces to be assembled together. Making proteins structurally modular so they can be assembled together is not an easy task, but not unthinkable as our knowledge of proteins increase.

A functional standard signal that could be used is the rate of enzymatic activity, for example, protein activity per second (PaPS). However, due to the large number of types of enzymatic activities, it is difficult to create one standard that encompasses them all, as a phosphorylation event is quite different from a methylation event. But, for example, phosphorylations per second could be used as a device-independent signal carrier for a family of kinases.

Protein layer functional signals such as enzymatic rates could have faster response times compared with transcription-based signals. However, proteins are currently more difficult to engineer than nucleic acids. There will undoubtedly be different situations in which different signals will be advantageous.

3.8 Interlayer: Cell

Description This interlayer specifies how cell internals are targeted to locations inside and outside the cell. For example, molecules can be exported outside of the cell, targeted to the membrane, or localized to a location such as the nucleus.

Standards Tags known to target molecules to specific locations can be standardized. For example, standardizing a protein export tag makes it easier to design such proteins.

Interfaces A module with an interface at the cell interlayer is a part that targets another molecule to a location in the cell. For example, the phospholocator can target a protein to the nucleus or cytoplasm depending on phosphorylation state.

3.9 Layer 4: Cell

Description The cell layer provides operation at the level of whole cells, such as communicating with other cells via cell-cell signaling and processing signals from the environment. Population dynamics such as the variability among cells are also included within this layer.

Standards The definition of the cell includes the cell membrane and other organelle structures within the cell. Also, standards could define the maximum expected noise or variability among cells in a population.

Interfaces Interfaces at the cell layer include cell-cell signaling molecules allowing compatible cells to talk with each other. For example, cells that send a quorum signaling molecule or cells that detect and process signals sent from another cell can have module interfaces at the cell layer. Interfaces also exist for transferring information across the cell membrane, primarily using proteins such as membrane transporters and signal transduction mechanisms.

3.10 Interlayer: Chassis

Description The cell-chassis interface defines how to package designed parts into a chassis. This interface is analogous to the computer motherboard form factor specifications, which define the interface for inserting a motherboard into a case with power supply. Any standards-compatible motherboard can be inserted into any compatible case with expected operation. Similarly, a cellular chassis is equivalent to the computer case and power supply and should specify the interface for inserting a system into it. Common biological chassis are plasmids and cell strains. A plasmid chassis usually has a multiple cloning site as an interface for allowing the insertion of compatible parts into the plasmid.

Standards As chassis are designed to be heavily reused, it is extremely important to have a well-characterized chassis interlayer. However, this interlayer is probably the most difficult to engineer a clean abstraction boundary. One interface standard needed is matching the demand from the lower layers for molecules with the supply capability of the chassis. At the same time, the chassis and components using the chassis should have minimal effects on each other. As biological systems are intrinsically deeply connected, a component of a chassis, such as a protein in a cellular chassis, can easily unintentionally influence an included component and vice-versa.

Although defining a clean chassis interface and building a strong abstraction barrier is not trivial, efforts are underway for constructing a standard interface for cellular host chassis [7]. By using orthogonal RNA polymerases and ribosomes for the cell chassis and the embedded system, the two transcription and translation systems can be decoupled providing engineered insulation. In addition, the chassis ability to supply the demands of the system in terms of materials, machinery, and energy can be characterized. Other chassis interlayer standards are also needed. Just as there are multiple form factors for computer cases, developing multiple standards for integrating into cellular chassis should prove most useful.

Interfaces An example module with an interface at this interlayer would be a module that produces extra ribosomes, thus increasing the protein production capacity of the chassis.

3.11 Layer 5: Chassis

Description The chassis layer provides the power supply and scaffolding for running and handling biological parts. The primary purpose of the chassis layer is to allow reuse of biological parts implementing “housekeeping” functions. The chassis should include commonly reused behavior, such as the ability to replicate autonomously. Examples of chassis include plasmids and cellular strains. For example, *E. coli* and yeast provide different chassis environments, but provide similar chassis functions for an integrated system.

Standards Chassis layer standards could define standard plasmids for propagating DNA, standard cell lines, and standard methods for characterizing these chassis components. For example, there are a multitude of *E. coli* strains, making it difficult to ensure systems are run consistently in the same conditions. The MIT Registry of Standard Biological Parts [22] is working to define standard plasmids and cell strains. An ideal cell chassis should be sequenced and fully annotated. Not only should plasmids and strains be standardized, they need to be characterized for use as chassis. Useful information would likely include such things as the amount of free nucleotides, amino acids, polymerases, and ribosomes that are available, the doubling time, the mutation rate, plasmid copy number, plasmid stability, and general metabolic load capacity.

Interfaces The chassis with its attached components can be considered as the complete physical package of the system being built. A chassis structural interface is needed for humans to handle it. Chassis interfaces include packaging or format of distribution such as tubes or 96-well plates.

3.12 Interlayer: Environment

Description The environment interlayer is how a chassis is activated by being inserted into its operating environment.

Standards The standard for inserting chassis into a particular environment is usually simple. For example, inoculating liquid media from a glycerol stock using a sterile loop or spreading cells on an agar petri dish are common lab methods.

Interfaces This interface includes how the chassis system affects the environment. A system can influence the environment, such as by depleting nutrients or by releasing waste molecules into the environment.

3.13 Layer 6: Environment

Description The environment layer defines the environment in which a system is expected to operate. Examples include: growth in continuous culture conditions at 37°C; growth in the ocean; or media lacking a certain amino acid.

Standards Different environments, such as minimal versus rich media, can allow for different cell growth rates, impacting system operation. Having standardized environments allows for replicating lab conditions across experiments and experimenters. Standards in the environment layer would specify standard operating conditions, such as defining batch or continuous culture, growth temperature, solid or liquid media, and minimal or rich media. For example, one standard media could be EZ rich defined medium, a variation of Neidhardt Supplemented MOPS Defined Medium [14,21]. Using a defined medium is one step in providing standard operating conditions for system operation, but many other variables also need to be specified for complete standardization of the environment conditions.

Interfaces Changes in the environment define the desired inputs and outputs for a system allowing both control of a system and obtaining information from a system. Examples include: responding to red light;

detecting chemical X at a certain concentration; or having an output green fluorescence. Standards here should specify such things as wavelengths, concentrations, and transfer curves. For example, excitation and emission wavelengths for measuring GFP would be a useful standard.

Note that the same signal in different systems can be treated either in this layer or in a lower layer. For example, a cell signaling molecule added by the user as input into the system falls in the environment interface. The same molecule used by a population of cells to communicate with each other would fall in the cell layer and if a cell gave off the molecule as a waste product into the environment, it could fall into the environment interlayer.

3.14 Interlayer: Application

Description The application interlayer specifies the requirements of the application layer in a form implementable by the lower layers.

Standards Specification standards are important in making sure specifications are consistent and implemented correctly. Standard specifications facilitate implementation reuse. There are many general specification areas that would benefit from standardization. For example, stability, noise, dynamic response, and safety are several types of specifications.

Stability specifications describe the desired long-term operational stability or instability of the system. The categories for stability standards include defining stability and robustness of operation over time, defining instability for elimination from the environment, and defining the conditions for evolution. The stability of a system can be specified in terms of either the physical (i.e. genetic) or functional (i.e. phenotype) stability. Parameters could include the half-life of the physical parts and overall functional stability. For example, a minimal risk of release standard could specify that a system will be undetectable in the environment after a specified period of time. Another standard could specify that a system is functionally robust in that the application specifications will be maintained even with a certain number of DNA mutations. Other examples include: the system self-

deconstructs within 10 doublings or 5 hours; the system will evolve to adapt to its environment; or the system maintains functionality stably over 1 week, after which the operation is undefined.

Noise is another category of an application specification. There are cases we want to use noise in our system [?, 26] and some means of specifying the desired noise is necessary. Even if we do not want noise, biological systems are inherently noisy [11], so specifications may specify acceptable noise margins and variability.

Dynamic response specifications provide characteristics such as speed of operation, timings, and thresholds for the system. Standards are especially needed for characterizing parameters like these. In addition, methods for tweaking parameters to match the desired specifications are necessary.

Safety standards could ensure that a system is not accidentally released outside of a defined environment. Similarly, standards for harmlessness and non-pathogenicity are an important category of standards to help cement the public's trust towards synthetically designed organisms.

Interfaces Possible interfaces to this layer could be such things as methods for tuning the behavior of a system. For example, standard interfaces for modulating thresholds or noise margins would be extremely helpful.

3.15 Layer 7: Application

Description The application layer is the top layer and defines the functionality that the system designer and users want. Examples include: detect chemical X and break it down; an 8-bit counter; or a cell cycle counter.

Standards Application standards specify system families that perform a particular task. An example application specification would be a cell cycle counter for *E. coli* that is stable for 100 doublings and emits purple light on every fifth division. Note that the specification includes some required implementation details (such as use of *E. coli*), yet leaves many un-

specified details (such as which particular strain), allowing multiple implementations at the lower layers. The responsibility for fulfilling the application specifications are spread out over the lower layers.

Interfaces The application interface allows for hooking applications together and for using the system in a desired manner. For example, consider pattern forming bacteria [4]. The application interface to this system involves placing bacteria in certain locations to get desired patterns.

4 Module Families

Another class of standards involves the definition of compatible module families. A module family standard consists of a set of standards to which all modules within the family conform. One potentially useful module family type is a logic family. A logic family can be used to connect modules to perform logic operations. An example from electronics is the transistor-transistor logic (TTL) family standard. Some standards within the TTL family include using a 5 volt power supply, defining a “low” signal between 0 and 0.8 volts and a “high” signal as between 2 and 5 volts, and using particular types of transistors. Biological logic families could include such classes as transcription-based logic, post-transcription logic, translation logic, or post-translational logic.

As one example, consider a transcription-based logic family of inverters engineered around protein repressors as transcriptional regulators. All devices in this family would have one PoPS input and one PoPS output with the output being a logical inversion of the input. The low and high signals are defined by specified PoPS levels at the RNA interlayer. It would use standard molecules such as nucleotides at the materials layer. A standard for joining the modules together at the DNA layer would enable the physical assembly of the modules in the family. In addition, application layer standards such as noise margins could be specified. A module family defined like this would enable all modules within the family to be treated similarly for a particular application such as logic.

5 Discussion

5.1 Related Work

There have been other proposed abstraction hierarchies for synthetic biology [2, 13]. However, these approaches categorize modules into groups such as parts, devices, systems, genes, reactions, pathways, cells, and cultures. The approach taken in the biological layer abstraction model is complementary to previous approaches. Instead of a hierarchy for module types, the layer model focuses on a hierarchy for module interfaces.

5.2 Abstraction Paradigms

Abstraction paradigms from other engineering domains can be applied to biological systems in order to manage complexity. The ideas of objects and inheritance of properties are used in many fields. One goal of object abstraction is to allow objects to work seamlessly together by separating the abstract interface from its concrete implementation. The biological layer model aims to provide standard abstract interfaces to make it possible to develop interoperable biological objects (i.e. parts).

Abstract objects can work together in two primary types of relationships. One kind of object relationship is that of inheritance, also known as the *is-a* relationship. Object 1 *is-a* object 2 when object 1 extends the functionality of object 2 while still being the type of object 2. For example, a “student” *is-a* “person.” A “person” object could define things common to a person such as having a head, arms, and legs. A “student” is a “person” and has all the characteristics of a person but has additional characteristics such as someone who sometimes studies, often goes to parties, etc. Similarly, a “graduate student” *is-a* “student.” A graduate student is a student with additional properties such as doing research, and it also automatically inherits all the properties of a “person.”

Another way two objects can be connected is the *containment* or more weakly the *uses* relationship. This type of relationship is also called the *has-a* relationship from the notion that object 1 has a version of

object 2. In the weaker form, object 1 only makes use of object 2. Unlike the *uses* relationship, the *contains* relationship implies a close inter-dependent relationship between the contained and containing object, such that neither object would exist in the same form without the other. For example, a “university” *has-a* “graduate student” and neither a university nor its graduate students would functionally exist without the other.

The parts, devices, and systems nomenclature provides one crude categorization of objects in terms of their conceptual use. This abstraction hierarchy is built upon inheritance or *is-a* relationships. A system *is-a* device and a device *is-a* part. Perhaps additional layers can be usefully inserted into this hierarchy in the future, but these three types appears to be sufficient currently. We could also imagine an alternative formulation of parts, devices, and systems using containment relationships. If devices are envisioned as containing parts, then a different type of abstraction hierarchy is created. However, the advantage of using inheritance relationships is that it clearly specifies that a device has all the properties of a part and a system has all the properties of a device. A definition using containment would not be specific enough to be useful. If a device is something that contains parts and devices and parts are distinctly different types, then it is unclear when something transforms from being a part to a device.

The biological layer model is a more fine-grained hierarchy built upon containment or *has-a* relationships. The application layer *has-a* I/O layer which *has-a* environment layer all the way down to the materials layer. The layers do not form an *is-a* relationship. Higher layers are not a subclass of lower layers. For example, the protein layer is distinctly different from the RNA layer. On the other hand, the protein layer uses or contains the RNA layer. The benefit of defining a containment hierarchy is that it allows us to design complex systems by composing parts together. Unlike the previous inheritance hierarchy, the layers were chosen to clearly delineate the boundaries where interfaces can be chosen.

Inheritance can be viewed as layering a different view or adding new functionality to an existing object, without making a copy of the object. It allows

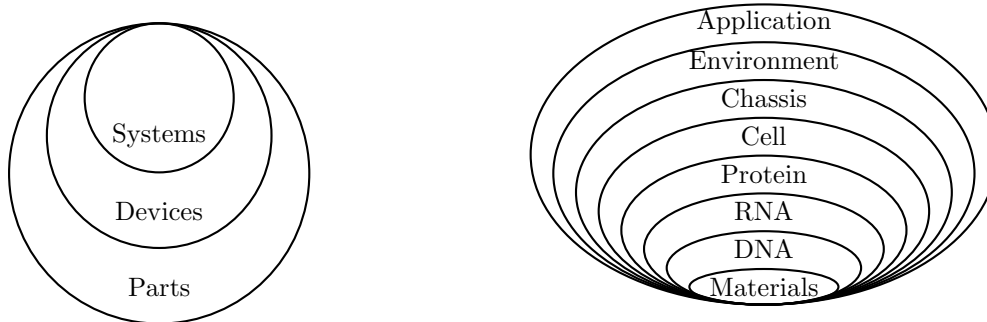


Figure 8: A view of two abstraction hierarchies. The parts/devices/systems hierarchy is based on inheritance relationships and the layer model uses containment relationships.

abstraction duplication in a way not possible with a containment relationship. For example, a single part could be treated as different devices depending on how we wish to view the inputs, outputs, and signals. The containment hierarchy does not provide for this polymorphic view of a part. One piece of RNA could be used by multiple protein layers but the RNA will always make the same protein. Containment, on the other hand, provides for a way of grouping heterogeneous objects together. One protein layer can contain multiple RNAs that each make different proteins.

Thus, we see that both the *is-a* and *has-a* biological hierarchies can coexist in a semi-orthogonal manner. Parts and devices can each occur at many different layers in the model. Figure 8 shows a schematic view of the two hierarchies. In the two models, the abstraction components are represented differently. In the *is-a* hierarchy, a “system” is a subclass of both devices and parts so is represented with the smallest domain. In a sense, “parts” are more abstract than “systems” as systems perform concrete tasks whereas parts are not necessarily useful by themselves. On the other hand, in the *has-a* layer model, the application layer, which intuitively has a similar role as a system, instead is the largest component containing everything else within it. In this usage, the application layer is the most abstract and general. These hierarchies highlight an interesting duality. A designed system can be considered as having a well-specified concrete function and, at the same time, be considered the most abstract for specifying only desirable

behavior with the implementation details hidden.

5.3 Conclusion

The biological layer model provides an extensible abstraction framework for designing complex biological systems. One assumption in the layer model is that the topology is a stack, rather than another graph structure. In some cases, it may not be obvious that the layers are stacked on top of one another. For example, the materials and application layers can interact with all layers. However, the stack model provides conceptual simplicity, allowing for abstracting different concerns into layers.

Each layer in the biological layer model provides external interfaces for a part to be connected with other parts and interfaces up and down the hierarchy in building up more complex parts. By standardizing the interfaces with functional and structural standards, such as PoPS and BioBricks, we can create a standards hierarchy allowing biological parts to interoperate. With abstractions and standards for defining how modules connect together, we can begin to assemble a collection of useful modules [22]. With standard biological parts that work in standard chassis in standard environments, we can increase our capacity for engineering complex systems.

Although natural systems unlikely use the abstractions presented here, an intelligent designer can benefit from adapting engineering principles in constructing many component biological systems. The useful-

ness and practical feasibility of the biological layer abstraction model remains to be demonstrated. Experimental work on characterizing and constructing parts at different layers is proceeding in many different labs. The abstraction framework presented here is the beginning of an attempt to bring together the work of many people. Items to be addressed include demonstrating that meaningful module interfaces can be defined for biological systems, specifying accepted standards at each layer and interlayer, and specifying standards for characterizing module interfaces.

References

- [1] J. Christopher Anderson, Ning Wu, Stephen W. Santoro, Vishva Lakshman, David S. King, and Peter G. Schultz. An expanded genetic code with a functional quadruplet codon. *Proc. Natl. Acad. Sci. USA*, 101(20):7566–7571, May 2004.
- [2] Ernesto Andrianantoandro, Subhayu Basu, David K Karig, and Ron Weiss. Synthetic biology: new engineering rules for an emerging discipline. *Molecular Systems Biology*, 2(1), 2006.
- [3] J. D. Bain, Christopher Switzer, A. Richard Chamberlin, and Steven A. Benner. Ribosome-mediated incorporation of a non-standard amino acid into a peptide through expansion of the genetic code. *Nature*, 356:537–539, April 1992.
- [4] Subhayu Basu, Yoram Gerchman, Cynthia H. Collins, Frances H. Arnold, and Ron Weiss. A synthetic multicellular system for programmed pattern formation. *Nature*, 434:1130–1134, April 2005.
- [5] Steven A. Benner. Understanding nucleic acids using synthetic chemistry. *Accounts of Chemical Research*, 37(10):784–797, October 2004.
- [6] Mohua Bose, Dan Groff, Jianming Xie, Eric Brustad, and Peter G. Schultz. The incorporation of a photoisomerizable amino acid into proteins in *E. coli*. *Journal of the American Chemical Society*, 128(2):388–389, 2006.
- [7] Barry Canton. Engineering the interface between cellular chassis and integrated biological systems. MIT Ph.D. Thesis Proposal <http://hdl.handle.net/1721.1/19813>, June 2005.
- [8] Austin Che. BioBricks++: Simplifying assembly of standard DNA components. <http://austinche.name/docs/bbpps.pdf>.
- [9] Austin Che. Fluorescence assay for polymerase arrival rates. Master’s thesis, MIT, 2003.
- [10] Michael B. Elowitz and Stanislas Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–338, January 2000.
- [11] Michael B. Elowitz, Arnold J. Levine, Eric D. Siggia, and Peter S. Swain. Stochastic gene expression in a single cell. *Science*, 297(5584):1183–1186, August 2002.
- [12] Drew Endy. Adventures in synthetic biology. *Nature*, November 2005.
- [13] Drew Endy. Foundations for engineering biology. *Nature*, 438(7067):449–453, November 2005.
- [14] EZ rich defined medium. http://openwetware.org/wiki/Neidhardt_EZ_Rich_Defined.
- [15] Dan Ferber. Microbes made to order. *Science*, 303(5655):158–161, January 2004.
- [16] Timothy S. Gardner, Charles R. Cantor, and James J. Collins. Construction of a genetic toggle switch in *Escherichia coli*. *Nature*, 403(6767):339–342, January 2000.
- [17] ISO. Information technology – Open Systems Interconnection – basic reference model: The basic model. ISO/IEC 7498-1, 1994.
- [18] Thomas F. Knight. Idempotent vector design for standard assembly of biobricks. Technical report, MIT, 2003.
- [19] Haibo Liu, Jianmin Gao, Stephen R. Lynch, Y. David Saito, Lystranne Maynard, and Eric T. Kool. A four-base paired genetic helix with expanded size. *Science*, 302(5646):868–871, October 2003.
- [20] Ryan A. Mehl, J. Christopher Anderson, Stephen W. Santoro, Lei Wang, Andrew B. Martin, David S. King, David M. Horn, and Peter G. Schultz. Generation of a bacterium with a 21 amino acid genetic code. *JACS*, 125(4):935–939, 2003.

- [21] Frederick C. Neidhardt, Philip L. Bloch, and David F. Smith. Culture medium for enterobacteria. *Journal of Bacteriology*, 119(3):736–747, September 1974.
- [22] MIT Registry of Standard Biological Prats. <http://parts.mit.edu/>.
- [23] Ira Phillips and Pamela Silver. A new biobrick assembly strategy designed for facile protein engineering. Technical report, Harvard Medical, 2006.
- [24] Oliver Rackham and Jason W Chin. A network of orthogonal ribosome · mRNA pairs. *Nature Chemical Biology*, 1(3):159–166, August 2005.
- [25] Christopher Y. Switzer, Simon E. Moroney, and Steven A. Benner. Enzymic recognition of the base pair between isocytidine and isoguanosine. *Biochemistry*, 32(39):10489–10496, 1993.
- [26] Lingchong You, III Cox, Robert Sidney, Ron Weiss, and Frances H. Arnold. Programmed population control by cell-cell communication and regulated killing. *Nature*, 428(6985):868–871, April 2004.