ACS SyntheticBiology

Research Article

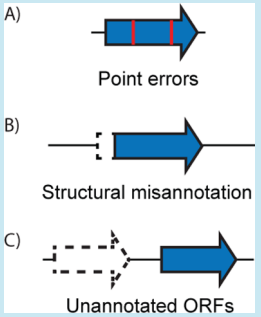
pubs.acs.org/synthbio

Engineered DNA Sequence Syntax Inspector

Timothy Hwei-Chung Hsiau and J. Christopher Anderson*

Bioengineering Department, University of California, Berkeley, California 94720, United States

ABSTRACT: DNAs encoding polypeptides often contain design errors that cause experiments to prematurely fail. One class of design errors is incorrect or missing elements in the DNA, here termed syntax errors. We have identified three major causes of syntax errors: point mutations from sequencing or manual data entry, gene structure misannotation, and unintended open reading frames (ORFs). The Engineered DNA Sequence Syntax Inspector (EDSSI) is an online bioinformatics pipeline that checks for syntax errors through three steps. First, ORF prediction in input DNA sequences is done by GeneMark; next, homologous sequences are retrieved by BLAST, and finally, syntax errors in the protein sequence are predicted by using the SIFT algorithm. We show that the EDSSI is able to identify previously published examples of syntactical errors and also show that our indel addition to the SIFT program is 97% accurate on a test set of *Escherichia coli* proteins. The EDSSI is available at http://andersonlab.qb3.berkeley.edu/Software/EDSSI/.



KEYWORDS: *synthetic biology, BioCAD, debugging*

Designed DNAs encoding polypeptides often contain design errors that cause experiments to prematurely fail; to address this concern, we have developed a computational tool that detects likely syntactic design errors in a genetic construct. DNA sequences are typically designed from prior knowledge of biological phenomena, but implementation of novel biological functions is error-prone.[1] Although computational design tools such as GenoCAD[25] and Eugene[22] exist to aid in the design of genetic circuits based on known rules, experimental failure is still common. Discovery of the cause of the error requires some experimental "debugging", and typically only after many alternative hypotheses have been ruled out is the problem traced to incorrect or missing features in the designed DNA.

Previously, we have reviewed the many challenges that genetic engineers face.[1] One class of challenges is incorrect or missing elements in the DNA, here termed syntax errors. These errors can be predicted beforehand and corrected in the design stage. However, in many experiments today, these errors are discovered during the debugging stage after an experiment has failed.

Syntax errors that occur in polypeptide-encoding DNAs result in a nonfunctional protein or cause unintended interactions in the early, proof-of-concept stages of a project. Such a result can then be interpreted as a total failure of the experiment rather than an artifact caused by syntax errors. We have identified three major sources of syntax errors and the corresponding manifestations: (1) sequencing errors in the primary data that lead to point mutations or truncations, (2) wrong gene structure annotations, typically of the gene start site that lead to truncated proteins, and (3) unanticipated open reading frames (ORFs), which give rise to unintended polypeptides. These syntax errors are shown in Figure 1. We give examples for these three types of syntax errors and present an analysis pipeline that aids in the identification of such errors.

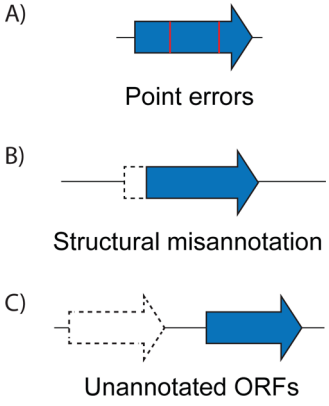


Figure 1. Causes of syntactical error in genetic designs. (A) Point errors can result from erroneous sequencing or manual data entry. (B) Structural misannotation caused by late start sites can result in N-terminal truncations. (C) Unannotated ORFs can result in the expression of unintended genes.

Point Errors from Sequencing or Manual Data Entry.

Sequencing errors from Sanger technology in the 1990s were estimated at 0.1%.[2] Although next-generation sequencing can compensate for higher error rates in individual reads by using information from overlapping reads, finished contigs were nevertheless estimated to have an error rate of 0.33% as of 2009.[3] Additionally, error rates are unequally distributed across sequenced genomes and fluctuate on the basis of both local sequence composition and the specific sequencing technology employed. Sequencing errors can cause nonsynonymous mutations and truncations of a gene by introducing erroneous start or stop codons. Additionally, manual sequence editing has the potential to introduce this type of error and other types.

Received: October 27, 2013
Published: December 23, 2013

Real World Example. Engineers refactored a *Klebsiella* nitrogen fixation gene cluster to remove unwanted regulation by synthesizing sequences derived from NCBI entry X13303.1.[4] The synthesized genes were nonfunctional when tested by knockout complementation, and the failures were traced back to nonsynonymous mutations due to erroneous sequencing data in the original submission. Identifying the problem and correcting it by resequencing the source DNA consumed 3 months.[5]

Gene Structure Misannotation. While genome annotation has rapidly developed, predicted gene structures are still imperfect, and many erroneous entries exist in the databases. As one example, automated gene annotation software misannotates at least 10% of prokaryotic gene start sites.[6,7] Similarity-based analyses of genome sequences have identified gene-calling errors as high as 15%.[8] The gold standard for gene start site identification is experimental validation by N-terminal sequencing, which is sparse and not collected in a central database. While there have been efforts to improve annotation of gene start sites,[9] many of the entries in nr, NCBI's nonredundant protein database, still have erroneous annotations. Gene prediction in eukaryotes is nontrivial as the software must also accurately predict introns.[10] However, programs such as AUGUSTUS can now incorporate "hints" from diverse experimental sources such as RNA-seq, genomic conservation, or tandem mass spectrometry to improve predictions.[24] In general, gene structure misannotation can also happen when users manually infer the incorrect gene structure.

Real World Example. The *invF* gene was used in a design for genetic logic gates in *Escherichia coli*; however, because of an incorrect annotation, the synthesized ORF was truncated.[11] This error is particularly common when refactoring overlapping ORFs or when dealing with ORFs that have many methionines near the start (e.g., β-lactamase).

Unintended ORFs. Overlapping ORFs have been found in all domains of life. On average, 27% of genes in prokaryotic genomes are involved in at least one overlap [Lillo], and internal or partial ORFs can occur when sequences are copied from their native context. During transfer of a target ORF to a new context, annotation of the overlapping ORF may be forgotten or discarded. Additionally, ORFs expressed outside of their native context can contain unintended translational signals, such as a ribosomal binding site, that lead to production of truncated protein products.[13]

Real World Example. A chimeric gene composed of rabbit structural capsid protein VP60 fused to cholera toxin B subunit was not stable in *E. coli* hosts.[14] Constructing frame-shift mutations versions of the gene did not alleviate plasmid instability. The true cause of instability was found to be due to the use of nonstandard codons, which resulted in a translational start signal in a different frame and expression of a leucine-rich ORF. The leucine-rich polypeptide was hypothesized to insert into the membrane and was shown to be the cause of toxicity.

Errors in designed protein-coding sequences are predictable and preventable and cause unnecessary experimental delays. Thus, there is a need for software tools that decrease risk of failure by identifying potential errors in a genetic design. In this article, we report the Engineered DNA Sequence Syntax Inspector (EDSSI), a new tool that uses sequence conservation to identify primary sequence syntax errors in the user's protein-coding sequence. We focus on protein-coding syntax for genes from any source being expressed in bacteria, to facilitate the common practice of placing existing protein-coding sequences under engineered transcriptional regulation. Additionally, protein-coding syntax is much better understood than non-protein-coding syntax and poses a more tractable problem. In our tool, users input a DNA sequence, protein-coding regions are detected, and a homology-based approach is used to predict errors. Users can then view the syntax error analysis on the results page. By quickly allowing syntax errors to be considered or discarded as a hypothesis in troubleshooting experiments, this tool allows a more rational design of protein-coding sequences.

RESULTS AND DISCUSSION

Here we present the EDSSI, a sequence analysis tool that inspects input DNA for potential syntax errors in the protein-coding sequences when expressed in a bacterial context. By combining gene prediction, homologue retrieval, protein sequence alignment, and mutational analysis software, the EDSSI predicts one type of genetic design error. The EDSSI is available at http://andersonlab.qb3.berkeley.edu/Software/EDSSI, and an API service is available at http://andersonlab.qb3.berkeley.edu/Software/EDSSI/api.html. The source code is available at https://github.com/hsiaut/EDSSI under the BSD license.

Performance. The EDSSI analysis pipeline fits well into common design workflows. The two BLASTs are the main bottlenecks of the pipeline, so results for both are cached to improve performance for commonly queried sequences. The protein BLAST searches are also conducted in parallel to accelerate performance. We timed the analyses for 20 *E. coli* genomic loci and found that on average each kilobase takes 3 min to run.

Sequence Inspector Reports. The EDSSI generates a report that contains a graphical representation of the input sequence. Detected ORFs are drawn as arrows and are labeled by a text annotation. The ORF labels can be toggled on and off by a button at the top right of the display. ORFs are color-coded by the level of evidence: ORFs with exact database matches are colored green, ORFs with database hits but no exact match orange, and ORFs with no database hits black. Errors in the protein sequence are drawn as red vertical bars.

Each ORF links to its corresponding multiple-sequence alignment. The sequence of the predicted ORF, or the input sequence, is given as the first sequence in the alignment. Subsequent protein sequences are ordered by similarity. The entire alignment is generated by JavaScript and also can be dynamically resized.

Predicted errors in the input sequence are depicted as color-coded vertical bars in the multiple-sequence alignment, with more likely errors encoded with a deeper shade of red. The amino acid characters are also color-coded to facilitate visual comparison.

Several files from each analysis are available for download. The complete output data are available as a JSON download to facilitate interoperability with automated genetic design software. The aligned protein sequences are available in a FASTA file. The conserved domain and homologue searches produce links to relevant, indexed abstracts in PubMed that users can read. The analysis pipeline also produces an annotated GenBank file that can be read by popular DNA editors such as ApE, LaserGene, etc.

A literature search was conducted to find and evaluate underlying software for the EDSSI. Genemark.hmm was found

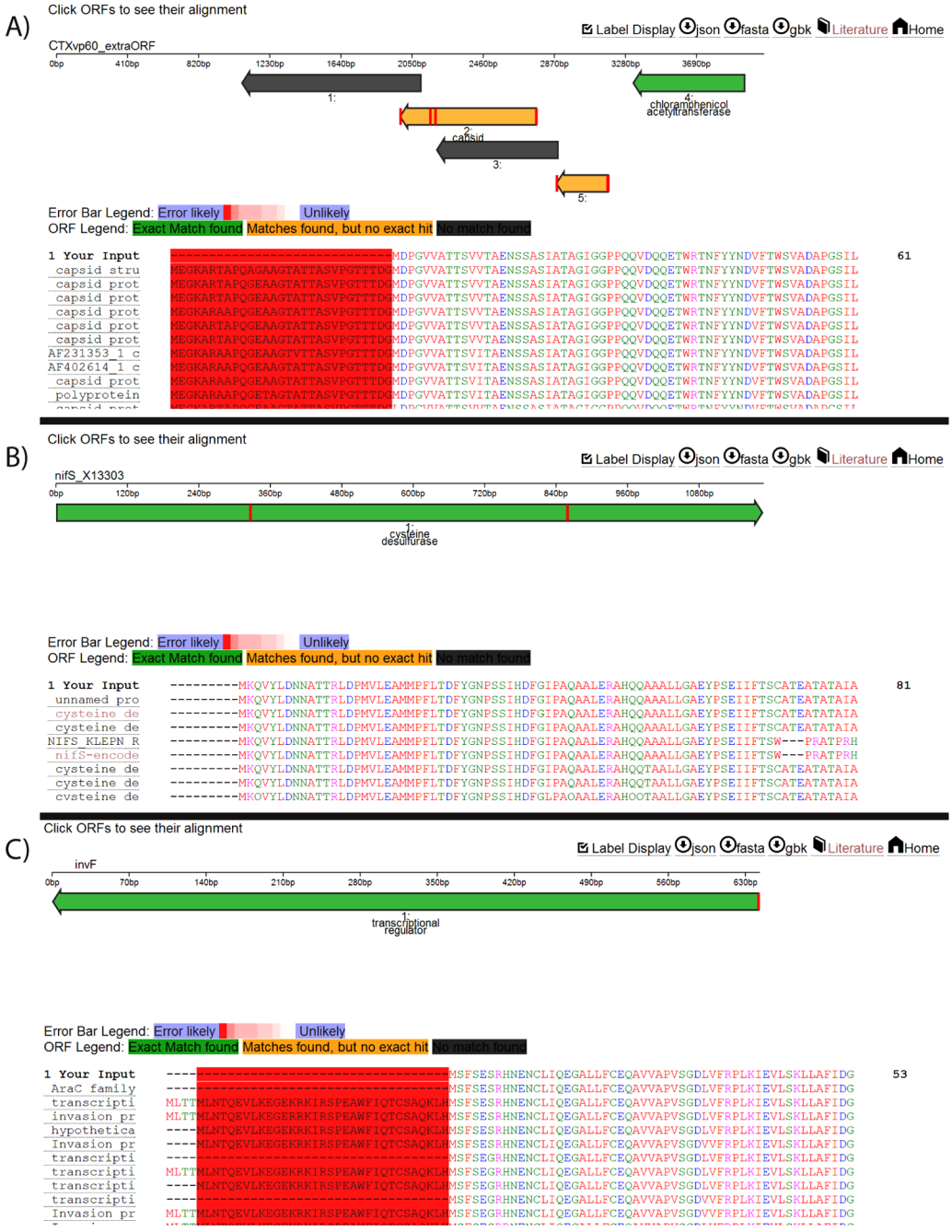


Figure 2. Examples of the results page. (A) Extraneous ORFs detected in the pCTXVP60 example. The gray ORFs are the leucine-rich ORFs with no homologues, while the two fusion partners are shown as orange ORFs. (B) Point mutations detected in GenBank entry X13303 used in the study[4] are displayed as vertical red bars on the ORF. (C) Analysis results of the truncated *invF* gene used in the study.[11] The truncation is shown as a vertical red bar at the 5′ end of the ORF and also highlighted in the multiple-sequence alignment.

to correctly predict 93.5% of experimentally verified genes across a wide range of bacteria and compares favorably with other gene callers.[21] We found Genemark.hmm to perform well on our plasmid-sized inputs, while some of the other software required more than 100 kb as a minimal input size because they were designed primarily for annotating genomes. SIFT has been benchmarked against a large set (thousands of substitution mutants) of functional data from nearly complete mutagenesis of LacI, HIV protease, and T4 lysozyme. SIFT has false positive rates of 20% and false negative rates of 31%.[18] Finally, to test our indel addition to the SIFT analysis workflow, we selected ORFs from the EcoGene Verified Protein Starts set (922 genes)[26] containing an internal methionine, generated the truncated genes, and ran them through the EDSSI. Truncated sequences may result in a false negative analysis if there are sufficient shorter, erroneous protein sequence entries in NCBI nr. However, the EDSSI was able to predict 97% of the truncated protein sequences.

Three Illustrative Examples. To demonstrate the utility of the EDSSI, we examined the three published examples of syntax errors discussed above. We first syntax-checked synthetic construct pCTXvp60.[14] The leucine-rich ORFs in pCTXvp60, including the toxicity causing ORF238, were correctly identified by the analysis. These artificial ORFs have no homologues and

dx.doi.org/10.1021/sb400176e | *ACS Synth. Biol.* 2014, 3, 91−96

were therefore not identified by the conserved domain search. As expected, artificial fusion protein CTXvp60, which lacks a bacterial RBS, was detected as two separate ORFs as seen in Figure 2A. However, knowing about the spurious ORF238 would likely have aided in troubleshooting the unexpected but observed plasmid toxicity in *E. coli*.

We next examined the X13303.1 *nif* cluster from ref 4, and the EDSSI predicted 13 errors. The *nifS* gene is shown in Figure 2B as an example of how these point mutations are displayed. During *nif* cluster resequencing, 18 nonsynonymous mutations were found, of which eight agree with the ones found by the EDSSI. In comparison, the two homologous *nif* clusters found had only six predicted errors in the same 22 kb region. Knowing about the predicted errors in the *nif* cluster sequence would likely have aided in the debugging of the initial failed experiment. The EDSSI also successfully identified the truncation in the published *invF* construct (Figure 2C). While some database entries share the same truncated translation start site, the majority of entries have the genuine start site. Knowledge of those entries would have alerted the designers to the potential truncation.

Failed experiments are common in genetic engineering, and there is a need for software tools that provide suggestions for debugging these experiments. One common source of error is the subtleties in the DNA syntax of the tested constructs. Current practice relies on an implicit assumption that annotations and sequence databases are correct. However, as we have found in the published examples, those annotations can mislead the engineer and can cause simple experiments to fail. Via the creation of better tools for syntax checking and semantic verification, such experiments will have a lower chance of failure.

We used modern-day computer code editors as inspiration when we created the EDSSI. Modern-day computer code editors can find syntax errors or warnings before runtime, allowing a faster debugging cycle. Similarly, our sequence inspector will allow for upfront handling of errors or can provide hypotheses for failed experiments. We envision that the EDSSI could be useful to "sanity check" designed constructs before experimental effort is spent, and as a hypothesis-generation method when troubleshooting failed experiments.

Though there is no formal theory for how each side chain position contributes to overall protein function, statistical approaches for predicting deleterious mutations can provide a means of prediction. By using the statistical techniques pioneered by programs like SIFT, our EDSSI output correlates with expert human analysis for the three published examples and our synthetic test sequences. However, in the *nif* gene cluster example, the sequence inspector did not identify all of the nonsynonymous mutations found by resequencing. This disparity is due to a false negative of the SIFT program, or some of the mutated positions are tolerated. Empirical testing on sequences substituted with each mutation for the desired nitrogen fixation function could differentiate between the two interpretations. The synthetic benchmarks suggest that the ORF prediction and gap scoring algorithms can be used for pre-experimental error prediction, while the amino acid substitution scoring may be useful for hypothesis generation in postexperiment debugging.

The EDSSI can be improved by inclusion of more data. Analyses of rapidly evolving genes, such as endonucleases, or unique gene sequences will return many errors because the sequence inspector performs poorly when given few data points. As more strains and individuals are sequenced, the number of homologues for any given protein can be expected to increase. Also, more proteomic data will allow precise prediction of translation start sites. Integrating protein structure, when available, into the analysis could also improve predictions of effects of amino acid substitutions such as in the PolyPhen prediction pipeline.[19]

For the biological engineer, the ability to rule out certain designs before fabrication will have an important role in allowing complicated designs.[1,20] Already, with the right information, software systems can check the validity of designed logic gates[22] and metabolic pathways.[23] By addressing one common aspect of failure in genetic engineering, this tool will help move the practice closer to rational design.

METHODS

The sequence inspector workflow is illustrated in Figure 3. The workflow consists of the following steps: sequence submission, gene prediction, homologue search, alignment, scoring, and display.

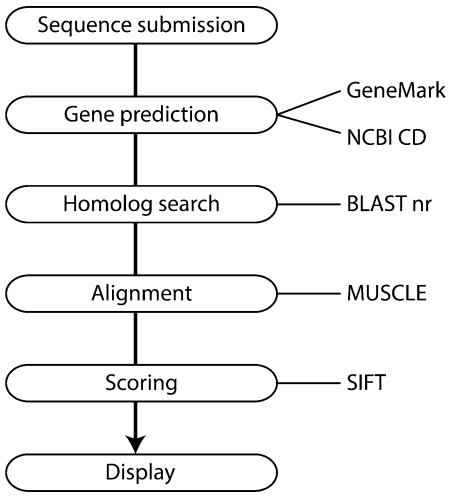


Figure 3. Data analysis workflow. The user submits DNA sequences through an online interface, which are then run through gene prediction, homologue search, alignment, and scoring. Associated programs or algorithms for each stage are shown.

Bioinformatics Workflow. The sequence inspector predicts genes in input DNA by using GeneMark.hmm and NCBI's Conserved Domain search (CD-search). The HMM framework of GeneMark.hmm uses the statistical patterns of nucleotides encoding proteins to predict likely genes. Predictions of translational start sites are further improved by incorporating a model of the ribosome binding site (RBS). CD-search identifies nucleotide regions matching protein family profiles.[15] The protein family profile match regions are then extended to the closest start and stop codon for a minimal gene prediction. The predicted genes from the two approaches are then merged if they overlap and are in the same frame.

The sequence inspector next searches for and retrieves homologous protein sequences, aligns the sequences, and scores the input sequence for syntax errors. To find protein homologues, for each predicted ORF, the program uses a BLAST search against the nonredundant (nr) protein database to find closely related genes. Full-length protein sequences identified by the BLAST search are retrieved, and the

homologues are aligned using the MUSCLE aligner.[16] The alignment is scored using the SIFT algorithm.[17] In brief, amino acid distributions at each column are used to calculate a normalized probability that the observed residue is correct. Aligned columns with more variation are more likely to tolerate substitutions than highly conserved positions. The standard cutoff of 0.05 was used. Because SIFT ignores gaps in the input alignment, we added a custom scoring function for the gapped positions that uses a simple weighted vote.

Results are passed as a JavaScript Object Notation (JSON) file and displayed using the JavaScript visualization library RaphaelJS and a JavaScript multiple-sequence aligner viewer developed in house. We use the ELink functionality provided by NCBI to retrieve publications relevant to each protein BLAST hit. Results from the analysis are output as an independent JSON file, which is read and displayed by the HTML/JS viewer.

Workflow Technical Details. Genemark.hmm version 2.8a is run with the *E. coli* model and the -r option, which uses an RBS model for start codon prediction. All other prediction options were kept as default. The conserved domain search was performed with the rpstblastn binary included in the BLAST+ package from the NCBI. Rpstblastn is run with an *e* value of 1×10^{-50}. Outputs from Genemark.hmm and rpstblastn are parsed by Python scripts to generate gene predictions for the input DNA. Python scripts were developed in house and have no dependencies.

After gene prediction, protein sequences are individually queried against the nr database using BLASTp, and an *e* value of 1×10^{-50} and up to 50 homologues are then retrieved. MUSCLE is then called with all default options. To retain the input order of the sequences, the stable.py script supplied with MUSCLE is used to reorder the alignment. Because the standalone SIFT binary does not accept gaps in the aligned FASTA input, any alignment columns with a gap in the reference are removed. Processed alignments are then analyzed using the info_on_seqs SIFT binary (SIFT version 5.0.3). Program default settings performed well on our test cases. Changing the settings resulted in no change in the output or worse performance (ORFs not called or errors not detected). BLAST settings of up to 50 homologues were chosen to limit excessive analysis times.

AUTHOR INFORMATION

Corresponding Author

*E-mail: jcanderson@berkeley.edu.

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

We thank Saurabh Srivastava, Oscar Westesson, Josh Kittleson, David Chen, and Zachary Russ for helpful discussions. We acknowledge the following funding sources: The Synthetic Biology Engineering Research Center (SynBERC, 0540879), National Science Foundation Graduate Research (CBET-1151220), and Department of Defense National Defense Science and Engineering Graduate Fellowship N66001-12-C-4204 to T.H.-C.H.

REFERENCES

(1) Kittleson, J. T., Wu, G. C., and Anderson, J. C. (2012) Successes and failures in modular genetic engineering. *Curr. Opin. Chem. Biol. 16* (3−4), 329−336.

(2) Bork, P., and Bairoch, A. (1996) Go hunting in sequence databases but watch out for the traps. *Trends Genet. 12.10*, 425.

(3) Farrer, R. A., Kemen, E., Jones, J. D., and Studholme, D. J. (2008) De novo assembly of the *Pseudomonas syringae* pv. *syringae* B728a genome using Illumina/Solexa short sequence reads. *FEMS Microbiol. Lett. 291.1*, 103−111.

(4) Temme, K., Zhao, D., and Voigt, C. A. (2012) Refactoring the nitrogen fixation gene cluster from *Klebsiella oxytoca*. *Proc. Natl. Acad. Sci. U.S.A. 109.18*, 7085−7090.

(5) Temme, K., personal communication.

(6) Pati, A., Ivanova, N. N., Mikhailova, N., Ovchinnikova, G., Hooper, S. D., Lykidis, A., and Kyrpides, N. C. (2010) GenePRIMP: A gene prediction improvement pipeline for prokaryotic genomes. *Nat. Methods 7*, 455−457.

(7) Poptsova, M. S., and Gogarten, J. P. (2010) Using comparative genome analysis to identify problems in annotated microbial genomes. *Microbiology 156*, 1909−1917.

(8) Kellis, M., Patterson, N., Endrizzi, M., Birren, B., and Lander, E. S. (2003) Sequencing and comparison of yeast species to identify genes and regulatory elements. *Nature 423*, 241−254.

(9) Wall, M. E., Raghavan, S., Cohn, J. D., and Dunbar, J. (2011) Genome Majority Vote Improves Gene Predictions. *PLoS Comput. Biol. 7*, e1002284.

(10) Stanke, M., Steinkamp, R., Waack, S., and Morgenstern, B. (2004) AUGUSTUS: A web server for gene finding in eukaryotes. *Nucleic Acids Res. 32*, W309−W312.

(11) Moon, T. S., Lou, C., Tamsir, A., Stanton, B. C., and Voigt, C. A. (2012) Genetic programs constructed from layered logic gates in single cells. *Nature 491*, 249−253.

(12) Lillo, F., and Krakauer, D. C. (2007) A statistical analysis of the three-fold evolution of genomic compression through frame overlaps in prokaryotes. *Biol. Direct 2*, 22.

(13) Lee, H., Whitaker, W., and Dueber, J. E. (2013) Avoidance of Truncated Proteins from Unintended Ribosome Binding Sites within Heterologous Protein Coding Sequences, manuscript in preparation.

(14) Umenhoffer, K., Fehér, T., Balikó, G., Ayaydin, F., Pósfai, J., Blattner, F. R., and Pósfai, G. (2010) Reduced evolvability of *Escherichia coli* MDS42, an IS-less cellular chassis for molecular and synthetic biology applications. *Microb. Cell Fact. 9*, 38.

(15) Powell, B. C., and Hutchison, C. A. (2006) Similarity-based gene detection: Using COGs to find evolutionarily-conserved ORFs. *BMC Bioinf. 7*, 31.

(16) Edgar, R. C. (2004) MUSCLE: Multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res. 32*, 1792−1797.

(17) Ng, P. C., and Henikoff, S. (2001) Predicting deleterious amino acid substitutions. *Genome Res. 11*, 863−874.

(18) Ng, P. C., and Henikoff, S. (2006) Predicting the effects of amino acid substitutions on protein function. *Annu. Rev. Genomics Hum. Genet. 7*, 61−80.

(19) Adzhubei, I. A., Schmidt, S., Peshkin, L., Ramensky, V. E., Gerasimova, A., Bork, P., Kondrashov, A. S., and Sunyaev, S. R. (2010) A method and server for predicting damaging missense mutations. *Nat. Methods 7*, 248−249.

(20) Purnick, P. E., and Weiss, R. (2009) The second wave of synthetic biology: From modules to systems. *Nat. Rev. Mol. Cell Biol. 10*, 410−422.

(21) Hyatt, D., Chen, G. L., Locascio, P. F., Land, M. L., Larimer, F. W., and Hauser, L. J. (2010) Prodigal: Prokaryotic gene recognition and translation initiation site identification. *BMC Bioinf. 11*, 119.

(22) Bilitchenko, L., Liu, A., Cheung, S., Weeding, E., Xia, B., Leguia, M., Anderson, J. C., and Densmore, D. (2011) Eugene: A domain specific language for specifying and constraining synthetic biological parts, devices, and systems. *PLoS One 6*, e18882.

(23) Hatzimanikatis, V., Li, C., Ionita, J. A., Henry, C. S., Jankowski, M. D., and Broadbelt, L. J. (2005) Exploring the diversity of complex metabolic networks. *Bioinformatics 21*, 1603−1609.

(24) Mortazavi, A., Schwarz, E. M., Williams, B., Schaeffer, L., Antoshechkin, I., Wold, B. J., and Sternberg, P. W. (2010) Scaffolding

a *Caenorhabditis* nematode genome with RNA-seq. *Genome Res. 20*, 1740−1747.
(25) Czar, M. J., Cai, Y., and Peccoud, J. (2009) Writing DNA with GenoCAD. *Nucleic Acids Res. 37*, W40−W47.
(26) Zhou, J., and Rudd, K. E. (2013) EcoGene 3.0. *Nucleic Acids Res. 41*, D613−D624.