Contents

Chapter 1

Computational Methods

The goal of this study was to predict the complex phenotypes of individual cells in terms individual molecules and their interactions. The primary challenges to building a unified whole-cell model of a single cell are three-fold: (1) *complexity*, processes relevant to cellular behavior are diverse and span a wide range of length and time scales, (2) *heterogeneity*, cellular networks have heterogeneous mathematical structures and are typically investigated using heterogeneous experimental methods, and (3) *sparsity*, little quantitative data rigorously describing single cell physiology is available. We felt that the flexibility of a hybrid model allowed us to best meet our goal while navigating these challenges. This chapter discusses the mathematical foundation and construction of the *M. genitalium* whole-cell model, as well as model fitting and validation. Appendix A provides further discussion of the implementation, execution, and testing of the whole-cell model.

1.1 Whole-Cell Simulation Algorithm

Overall, the whole-cell model is similar to a system of ordinary differential equations (ODEs) where the 16 cellular states are analogous to the state variables and the 28 cellular processes are analogous to the differential equations. Therefore, the whole-cell model is simulated using an algorithm comparable to those used to numerically integrate ODEs, such as the Runge-Kutta 4th order method[928]. Algorithm S1 summarizes the whole-cell simulation algorithm. First, the cell state variables are initialized. Second, the temporal evolution of the cell state is calculated on a 1 s time scale by repeatedly allocating the cell state variables among the processes, executing each of the cellular process sub-models, and updating the values of the cell states. Finally, the simulation terminates when either the cell divides, or the time reaches a predefined maximum value. The principal differences between the whole-cell model algorithm and numerical ODE integration methods are (1) the whole-cell model "equations" are grouped into 28 processes, (2) the whole-cell model "variables" are grouped into 16 cellular states, and (3) the state variables must be allocated among the processes at each time step to satisfy the sub-model independence assumption.

1.2 Cellular Processes

Because biological systems are modular, cells can be modeled by (1) dividing cells into functional processes, (2) independently modeling each process on a short time scale, and (3) integrating process sub-models at longer time scales. We divided *M. genitalium* into the 28 functional processes illustrated in Figure 1 of the accompanying manuscript, and modeled each process independently on a 1 s time scale using different mathematics and different experimental data. The sub-models spanned six areas of cell biology: (1) transport and metabolism, (2) DNA replication and maintenance, (3) RNA synthesis and maturation, (4) protein synthesis and maturation, (5) cytokinesis, and (6) host interaction. Sub-models were implemented as separate classes. See Chapter 3 for further discussion of each sub-model.

Algorithm S1 | Whole-cell dynamic simulation algorithm.

```
Construct whole-cell simulation objects using the KnowledgeBase classes
Computationally align processes and fit parameters
Identify initial conditions variance control parameters using Algorithm S2
Initialize cell state using Algorithm S3 and the fit values of the cell state variance control parameters
repeat
    Increment the time by 1 s
    Set the external conditions based on Table S3F and Table S3H
    Allocate shared resources:
    foreach metabolite $i$ in compartment $j$ do
        foreach process $k$ do  Calculate the demand, $d_{ijk}$, of process $k$ for metabolite $i$ in compartment $j$
        Divide the total count, $m_{ij}$, of metabolite $i$ in compartment $j$ into temporary dedicated pools, $m_{ijk}$, for each
        process proportional to demand, $m_{ijk} \leftarrow m_{ij} \frac{d_{ijk}}{\sum_k d_{ijk}}$
    Compute temporal evolution:
    foreach process $i$ do
        Retrieve the current values of cell state variables and the counts of metabolites allocated to process $i$
        Compute the contribution of process $i$ to the temporal evolution of the cell state
        Update the values of the cell state variables
until cell divided or time $> 1.5\times$ average mass doubling time
```

1.3 Cellular Process Integration

Cell States

We integrated the sub-models in three steps. First, we structurally integrated the process sub-models by linking their common inputs and outputs. However, rather than directly linking these inputs and outputs, we mapped the inputs and outputs of each sub-model onto 16 state variables which together represent the complete configuration of the modeled cell: (1) metabolite, RNA, and protein copy numbers, (2) metabolic reaction fluxes, (3) nascent DNA, RNA, and protein polymers, (4) molecular machines, (5) cell mass, volume, and shape, (6) the external environment including the host urogenital epithelium, and (7) time. Each cellular state variable was implemented as a separate class. See Chapter 2 for further discussion of the mathematics and computational implementation of each state variable.

Shared Resource Allocation

Second, to satisfy our sub-model independence assumption, at each time step we computationally allocated common sub-model inputs. At each simulation time step, prior to the evaluation of the sub-models, we estimated the metabolite resources required by each process and divided the total pool of each metabolite among the processes proportional to demand. Resource requirements were estimated by calculating the expected metabolite consumption of each process conditioned on the current cell configuration and an infinite metabolite supply. Algorithm S1 outlines the shared metabolite allocation algorithm. Because macromolecules and enzymatic capacity are less heavily shared by the cellular processes, we chose not to implement similar procedures for these sub-model inputs.

Process Alignment & Parameter Fitting

Third, because the 28 cellular processes were trained using different experimental data obtained by different investigators under different conditions using different techniques and different model organisms, we refined the values of the sub-model parameters to make the processes mutually consistent. This was necessary for example, because amino acid production by the `Metabolism` process, which was trained using the observed amino acid composition of *M. gallisepticum* reported by Morowitz *et al.*[870], conflicted with the amino acid requirements of the `Translation` process, which was trained using the observed *Mycoplasma* genetic code[53,168,607], the reported *M. pneumoniae* mRNA expression[569], and the N-end rule[586]. Specifically, Morowitz *et al.* reported that cysteine was only present in trace amounts, whereas the combination of the genetic code and observed mRNA expression is consistent with low, but not insignificant cysteine incorporation.

First, we manually identified constraints among groups of parameters spread across multiple processes. Among these constraints, we identified equality constraints between the cell chemical composition used to train the flux-balance analysis (FBA) metabolic objective and the expected metabolite requirements of the cellular processes. We also identified inequality constraints among the kinetic rate, expression, and required enzymatic capacity of each enzyme which describe the minimum expression of each gene consistent with all 28 sub-models.

Second, we computationally identified a set of parameter values which (1) satisfy all of these constraints and (2) deviate minimally from their experimentally observed values. Initially, we attempted to rigorously formulate this problem as a non-linear constrained optimization problem, and identify the parameter values which minimize the sum of squared differences between the adjusted and observed parameter values among all sets of parameters which satisfy the constraints. However, we were unable to find a feasible solution, much less the globally optimal solution, to this optimization problem. Instead, we focused on identifying a mutually consistent set of parameter values, and developed a heuristic procedure that uses the constraints to calculate a consistent set of parameter values from a subset of the parameters. This procedure primarily adjusted the gene expression. The adjusted gene expression correlated highly with that observed by Weiner *et al.*[569] ($R^2 = 0.68$, see Figure S1A). Section A.7.8 and the `FitConstants` class provide further discussion of the implementation of the parameter refinement procedure.

1.4 Initial Conditions

Cell theory[919] states that all cells are created from old cells, or, more mathematically, that on the time scale of a single-generation, mother and daughter cells are statistically identical. This principle relates the initial and final cell state distributions of the whole-cell model, providing a rigorous way to define the initial cell state distribution in terms of the dynamic model.

We applied this principle in seven steps (see Algorithm S2). First, we developed a method that calculates the expectation value of each cell state variable. Second, we approximated the distribution of each cell state variable by a standard, well-behaved statistical distribution, and set the mean of each distribution to its calculated value. For example, we assumed that the copy number of each RNA and protein species is multinomially distributed, and that the total cell mass is normally distributed. Third, we parameterized the variance of the distribution of each cell state variable and initially set the variance of each distribution to zero. Fourth, we developed the procedure outlined in Algorithm S3 to sample these distributions and set the initial value of each state variable. Fifth, we simulated a population of wild type cells using this cell state initialization procedure and calculated the variance of the final distribution of each cell state variable. Sixth, we set the initial variance of each state variable to its calculated final variance. Finally, we repeated steps five and six until convergence.

Algorithm S2 | Initial conditions identification algorithm.

```
Initialize the initial cell state variance control parameters: $\sigma_m \leftarrow 0$, $\eta_r \leftarrow 0$, $\eta_p \leftarrow 0$
repeat
    Simulate the life cycle of a population of wild type cells using Algorithm S3 to initialize the value of each cell state variable
    Randomly segregate the cellular content into two daughter cells
    Calculate the variances of the total cell mass, RNA copy number, and protein copy number states
    Set the values of the initial distribution control parameters of each state equal to that of the final distribution
    $\sigma_m \leftarrow$ standard deviation of the final cell mass distribution
    $\eta_r \leftarrow \sigma_r^2/N_r$, where $N_r$ and $\sigma_r^2$ are the mean and variance of the final RNA copy number distribution
    $\eta_p \leftarrow \sigma_p^2/N_p$, where $N_p$ and $\sigma_p^2$ are the mean and variance of the final RNA copy number distribution
until the initial variance control parameters ($\sigma_m$, $\eta_r$, and $\eta_p$) converge
```

Algorithm S3 | Cell state initialization procedure.

```
Input: $\sigma_m \leftarrow$ standard deviation of the initial total cell mass distribution
Input: $\eta_r, \eta_p \leftarrow$ RNA and protein copy number distribution initial variance control parameters
Input: $f_r, f_p \leftarrow$ reconstructed fractional cell RNA and protein composition
Input: $e_r, e_p \leftarrow$ expected relative expression of each RNA and protein species
Input: $w_r, w_p \leftarrow$ molecular weight of each RNA and protein species
Input: $N_i\left(f_i, e_i, w_i\right) \leftarrow f_i m / \left(e_i' * w_i\right)$ total initial RNA ($i$ =r) and protein ($i$ =p) copy number functions
Set time $\leftarrow$ 0 s
Set values of external stimuli and metabolites according to Table S3F and S3H
Set total cell mass, $m \leftarrow\sim N\left(\mu, \sigma_m\right)$, and calculate cell volume and shape
Set the metabolite counts according to the total cell mass and reconstructed cell composition (see Table S3I)
Initialize the Chromosome state with one methylated chromosome; decrement dNMP counts to maintain cell mass
Set mature RNA copy numbers according to multinomialRand($\eta_r N_r$, $e_r$)$/\eta_r$; decrement NMP counts
Set mature protein monomer copy numbers according to multinomialRand($\eta_p N_p$, $e_p$)$/\eta_p$; decrement amino acids
Form macromolecules by calculating the steady-state of the Macromolecular Complexation process
Set the RNA Polymerase and Transcript states to a steady-state of the Transcription sub-model
Set the Ribosome and Polypeptide states to a steady-state of the Translation sub-model
Set the FtsZ state to a steady-state of the FtsZ Polymerization sub-model with no septal rings
Set the growth rate and metabolic reaction fluxes to a steady-state of the Metabolism sub-model
Set the Host state to a steady-state of the Host Interaction sub-model
Set the chromosome protein occupancy to a steady-state the chromosome-interacting sub-models
```

1.5 Reconstruction

The *M. genitalium* whole-cell model was based on a detailed reconstruction of *M. genitalium* physiology developed from over 900 primary sources, reviews, books, and databases. First, we reconstructed the organization of the *M. genitalium* chromosome including the locations of each gene, transcription unit, promoter, and protein binding site primarily based on studies by Weiner *et al.*[411], Güell *et al.*[418], and the CMR genome annotation[168]. We also reconstructed the affinity of RNA polymerase for each promoter based on the reported expression[569] and half-life[602] of each RNA species.

Second, we functionally annotated each gene beginning with the CMR[168] annotation. We annotated genes with additional information from the BioCyc[6], KEGG[113], NCBI[61], and UniProt[96] genome annotations. To fill gaps in the reconstructed organism, such as observed reactions without reported enzyme catalysts, and to maximize the scope of the model, we also expanded and refined each gene's annotation using primary research articles and reviews identified by systematically searching PubMed and Google Scholar for each gene and homologs of each gene. Table S3J lists all functional annotations assigned beyond the CMR annotation. Additionally, we curated the reported essentiality[193] of each gene product.

Next, we curated the structure of each gene product, including the sequence of each protein, the post-transcriptional and post-translational processing and modification of RNA and protein, the signal sequence and localization of each protein, the DNA footprint of each DNA-binding protein, the chaperones and prosthetic groups required to fold each protein, the subunit composition of each protein and ribonucleoprotein complex, and the disulfide bonds of each protein and complex.

After annotating each gene, we categorized the genes into 28 cellular processes. We curated the chemical reactions of each cellular process with particular emphasis on reactions needed to interface the processes. For example, we added several metabolic reactions to provide the metabolites required for RNA modification. We also added metabolic reactions to catabolize modified nucleotides produced by the degradation of modified RNA. We reconstructed the stoichiometry and catalysis of each chemical reaction based on the databases BioCyc[6] and KEGG[113], a *M. genitalium* FBA metabolic developed by Suthers *et al.*[610], and hundreds of additional primary research articles. We reconstructed the kinetics of each reaction primarily based on the databases BRENDA[570] and SABIO-RK[100].

We reconstructed the *M. genitalium* metabolome based on the substrates and products of the reconstructed chemical reactions, the observed chemical compositions of *M. gallisepticum*[870] and *E. coli*[393], and the reported chemical composition of SP-4 *Mycoplasma* growth medium[754–759]. We reconstructed the structure of each metabolite based on several metabolomic databases including PubChem[587]. We reconstructed the protein regulatory properties of each metabolite primarily based on DrugBank[847].

Finally, we calculated the empirical formula, molecular weight, and several other physical properties of each metabolite, RNA, protein, and macromolecular complex using the `KnowledgeBase` classes, ChemAxon Marvin[594], and ExPASy ProtParam[585].

Because *M. genitalium* is not well-studied, the *M. genitalium* reconstruction was primarily based on studies of *M. genitalium* homologs identified by bi-directional best BLAST. Where possible, the *M. genitalium* reconstruction was based on closely related organisms.

Table S3A-S3S define the reconstructed *M. genitalium* organism including the structure of every metabolite; the sequence of every RNA and protein; and the stoichiometry, kinetics, and catalysis of every reaction. Table S3T-S3BK describe the how the reconstructed organism was developed, including detailed notes on how the value of each reconstructed property was derived by consensus of all available experimental observations and computational predictions. List S1 lists the principal sources of the *M. genitalium* reconstruction; Table S3S provides a complete list of all the sources of the reconstruction. Table S2B-S2C list the computationally refined values of the reconstructed cellular composition and gene expression. See Chapter 2 for further discussion of the reconstruction of each cell variable. See Chapter 3 for further discussion of the reconstruction of each cellular process sub-model. See Section 1.3 for further discussion of modeling fitting and computational refinement of the reconstruction.

List S1. Primary sources of the *M. genitalium* reconstruction.

Data source	Content
Bernstein *et al.*, 2002[602]	mRNA half-lives
BioCyc[6]	Genome annotation, metabolic reactions
BRENDA[570]	Reaction kinetics
CMR[168]	Genome annotation
Deuerling *et al.*, 2003[388]	Chaperone substrates
DrugBank[847]	Antibiotics
Eisen *et al.*, 1999[891]	DNA repair
Endo *et al.*, 2007[391]	Chaperone substrates
Feist *et al.*, 2007[558]	Metabolic reactions
Glass *et al.*, 2006[193]	Gene essentiality
Güell *et al.*, 2009[418]	Transcription unit structure
Gupta *et al.*, 2007[280]	N-terminal methionine cleavage
KEGG[113]	Genome annotation, orthology
Kerner *et al.*, 2005[389]	Chaperone substrates
Krause *et al.*, 2004[409]	Terminal organelle assembly
Lindahl *et al.*, 2000[462]	DNA damage
Morowitz *et al.*, 1962[870]	Cell chemical composition
NCBI Gene[61,777]	Genome annotation
Neidhardt *et al.*, 1990[393]	Cell chemical composition
Peil, 2009[105]	RNA modification
PubChem[587]	Metabolite structures
SABIO-RK[100]	Reaction kinetics
Solabia[754–759]	Media chemical composition
Suthers *et al.*, 2009[610]	Metabolic reactions
UniProt[96]	Genome annotation
Weiner *et al.*, 2000[411]	Promoters
Weiner *et al.*, 2003[569]	mRNA expression

1.6 Experimental Validation

The *M. genitalium* whole-cell model was validated by comparing the model's predictions to three experimental datasets: (1) the essentiality of each *M. genitalium* gene reported by Glass *et al.*[193], (2) the measured growth rates of 12 non-essential *M. genitalium* single-gene disruption strains, and (3) the cytosolic concentrations of 39 *E. coli* metabolites reported by Bennett *et al.*[392] and curated by Sundararaj *et al.*[394]. The *M. genitalium* whole-cell model also reproduces several experimental measurements which were used to train the model including the published cellular composition of *M. gallisepticum*[870], the measured RNA composition of *E. coli*[393], the reported *M. pneumoniae* mRNA expression[569], the observed *E. coli* mRNA half-lives[602], and the measured growth rate of wild type *M. genitalium*.

To validate the model against the observed gene essentiality and the observed disruption strain growth rates, we first simulated the individual disruption of each gene. We ran 5 simulations of each single-gene disruption strain by (1) randomly initializing the cell state using Algorithm S3, (2) deleting the *in silico* gene, and (3) calculating the temporal evolution of the cell state for the first generation post-disruption. Gene disruption was implemented in two steps: (1) we modeled insertion of a transposon of length zero which reduces the stability of the terminal products of the deleted gene, and set the half-life of the RNA and protein products of the deleted gene to zero; and (2) to more quickly highlight altered phenotypes, we deleted all RNA and protein products of the deleted gene. Next, we calculated the mean predicted mass doubling time, cell cycle length, terminal organelle protein mass, and damaged protein copy number of each disruption strain.

Third, we calculated the mean growth rate of each single-gene disruption strain at successive generations post-disruption. Rather than simulating the complete dynamics of successive generations, which was infeasible due to the significant computational cost of each simulation, we predicted only the growth rate of each disruption strain at successive generations post-disruption by initializing simulations using a modified version of the method described above. (1) We initialized cells using the wild type cell initialization method. (2) We deleted the *in silico* gene as previously described. (3) To simulate the long-term effects of the gene disruption and dilution resulting from cellular growth and division, we reduced the copy numbers of macromolecules which are normally synthesized by the deleted gene product. (4) We calculated the growth rate using the `Metabolism` process.

Fourth, we classified each *in silico* single-gene disruption strain as (quasi-)essential if the predicted first generation cell cycle length was significantly ($P \leq 0.01$) less than that of wild type *in silico M. genitalium*, if the terminal organelle was protein mass was significantly ($P \leq 0.01$) less than that of wild type *in silico M. genitalium*, if the predicted damaged protein copy number was significantly ($P \leq 0.01$) greater than that of wild type *in silico M. genitalium*, or if the growth rate declined over successive generations. We found that the model reproduces the observed gene essentiality with 79% accuracy. Figure 6B of the accompanying manuscript illustrates these distinct disruption strain phenotypes. Figure S2 illustrates the distribution of growth rates among wild type *M. genitalium* and the quasi-essential and essential single-gene disruption strains. Table S2G lists the predicted growth rate of each disruption strain.

Fifth, we compared the experimentally observed and predicted growth rates of 12 non-essential single-gene disruption strains (see Figure 7A of the accompanying manuscript and Table S1), and found that the model correctly predicts the measured growth rates of 67% of the disruption strains.

To validate the model against the Bennett *et al.*[392] and Sundararaj *et al.*[394] datasets, we calculated the mean concentration of each cytosolic metabolite in a population of 128 wild type cells. Figure 2E of the accompanying manuscript and Table S2E compare the predicted and measured concentrations of 39 cytosolic metabolites, illustrating that 70% of the model's predictions are statistically consistent with the Sundararaj *et al.* dataset. The model doesn't reproduce the Bennett *et al.* dataset, and interestingly, there is significant disagreement between the Bennett *et al.* and Sundararaj *et al.* datasets.

1.7 Computational Implementation

The whole-cell model was implemented in `MATLAB`, and consisted primarily of the main `Simulation` class and one class for each cellular state and process. The computational correctness of the whole-cell model algorithm was validated using unit testing. The *M. genitalium* reconstruction was stored using a modified version of the BioWarehouse schema[918] in a MySQL relational database. The knowledge base was viewed and edited using a web-interface implemented in PHP. Several `KnowledgeBase` classes represented the knowledge base in `MATLAB`. Appendix A provides further discussion of the whole-cell model architecture and it's computational validation.

1.8 Computational Simulation & Analysis

We used the whole-cell model to simulate 192 wild type cells and 3,011 single-gene deletants. All simulations were performed with `MATLAB` R2010b on a 128 core Linux cluster. The predicted dynamics of each cell was logged at each time point and subsequently analyzed using `MATLAB`. Appendix A provides further discussion of the execution, logging, and analysis of the whole-cell model.

Chapter 2

Cellular State Methods

The whole-cell model used 16 state variables to represent the instantaneous configuration of *M. genitalium* and integrate the 28 modeled cellular processes. The 16 state variables represented seven areas of cellular physiology: (1) copy numbers of metabolites, RNA and proteins, (2) metabolic reaction fluxes, (3) nascent DNA, RNA, and protein polymers, (4) molecular machines, (5) cell-level properties, including mass, volume and shape, (6) nascent polymers of DNA, RNA and protein, and (7) time. This chapter provides detailed discussions of the mathematics and computational implementation of each state variable.

The `Metabolite`, `Rna`, `Protein Monomer`, and `Protein Complex` states represented the copy number of each metabolite, RNA, protein monomer, and macromolecular complex. The complement of metabolite species was indirectly reconstructed by reconstructing the chemical reactions of each cellular process. The RNA complement was primarily reconstructed from the *M. genitalium* genomic annotation[168], the experimentally defined operon structure of *M. pneumoniae*[418], and the reported complement of *E. coli* RNA polycistronic cleavages and non-coding RNA modifications. The protein complement was primarily reconstructed from the predicted localization and signal sequence of each protein gene product (see Table S3AM-S3AO), the observed chaperone interactions of *E. coli*[388,389] and *B. subtilis*[391], the observed complement of *M. genitalium* and *M. pneumoniae* protein modifications[94,277,282,283,672], the reported N-terminal methionine cleavage of *Shewanella oneidensis* MR-1[280], and the inferred subunit composition of each macromolecular complex (see Table S3AS).

The `Metabolic Reaction` state recorded the predicted flux of each metabolic reaction. The *M. genitalium* metabolic network was reconstructed as described in Section 3.10.

The `Chromosome`, `Transcript`, `Polypeptide`, and `FtsZ Ring` states represented the configurations of the chromosome, nascent RNA and protein polymers, and FtsZ septal ring. The `Chromosome` state represented the polymerization, protein occupancy, and modification status of the chromosomes. The *M. genitalium* chromosome was sequenced by Fraser *et al.*[57]. Protein binding sites and DNA footprints were reconstructed from the primary literature and several databases (see Table S3M and S3N). DNA modification sites were predicted based on the reported DNA motif of the MunI methylase[96]. The `Transcript` and `Polypeptide` states represented the sequence of each nascent RNA transcript and polypeptide. The sequence of each RNA and protein species was reconstructed as described in Chapter 3. The `FtsZ Ring` state represented the configuration of the FtsZ septal ring. The structure of the FtsZ septal ring was reconstructed based on the Li *et al.* iterative pinching model[611].

The `RNA Polymerase` and `Ribosome` states represented the detailed configuration of each RNA polymerase and ribosome. The `RNA Polymerase` state represented the status – free, non-promoter bound, promoter-bound, or actively transcribing – of each RNA polymerase molecule, and the chromosomal location and direction of each DNA-bound RNA polymerase. The affinity of RNA polymerase for each promoter was primarily reconstructed from the observed expression of each RNA gene product[569] and the observed half-life of each *E. coli* mRNA[602]. The genetic code was reconstructed based on that of *M. pneumoniae*[53].

The Mass state represented the total cell mass. The Geometry state represented the cell shape and volume. The average cell size, density, and mass were reconstructed from studies by Baldwin *et al.*[553], Bray[554], and Zhao *et al.*[555].

The Metabolite, Stimulus, Host, and Geometry states represented the configuration of the external environment, including the extracellular copy number of each metabolite which was reconstructed as described in Section 3.10. The Stimulus state represented the temperature, the fluxes of six types of radiation, and the status of three common experimental stress conditions. Table S3F describes the reconstruction of the Stimulus state. The Host state represented four properties of the host human urogenital epithelium: (1) *M. genitalium* adherence, (2) activation of Toll-like receptors 1, 2, and 6, (3) activation of the host transcriptional regulator NF-κB, and (4) activation of the host inflammatory response. The Geometry state represented the volume of the extracellular environment.

Finally, the Time state represented the time elapsed from the beginning of the simulation.

2.1 Chromosome

Biology

Chromosomes encode the structure and function of every RNA and protein, and thereby control cellular behavior. Due to their critical function and large size, cells dedicate considerable resources to chromosome replication, maintenance, and compaction. This state represents the polymerization, winding, modification, protein occupancy, and (de)catenation status of the chromosome(s).

Reconstruction

Structure and Organization

The reconstructed *M. genitalium* chromosome contains 525 genes based on the Comprehensive Microbial Resource (CMR) genomic annotation[168] (see Table S3J) organized into 335 transcription units based on the *M. pneumoniae* operon organization experimentally defined by Güell *et al.*[418] and the *M. genitalium* promoters computationally predicted by Weiner *et al.*[411] (see Table S3U). The reconstructed genome also contains 17 transcriptional regulatory elements based on 15 studies and databases[96,110,112,186,196,418–420,433–438,505] (see Table S3P), 2,283 DnaA binding sites computationally identified based on the consensus binding motif reported by Grimwade *et al.*[744] and Speck *et al.*[745] (see Table S3L), 760 MunI restriction/modification (R/M) sites computationally identified based on the reported MunI binding motif[96] (see DNA Repair process), and 19 short tandem repeats identified by Ma *et al.*[183] and Washio *et al.*[184] (see Table S3V). Additionally, the location of the replication origin was reconstructed based on studies by Lobry[880], Jensen *et al.*[517], and others[521,848], and the predicted DnaA box sites. Finally, the reconstructed steady-state superhelicity is based on the observed equilibrium helical repeat length of plasmid DNA[879] and the observed superhelicity of *E. coli* DNA[749].

Protein Binding

The binding motif and footprint of each DNA-binding protein was reconstructed from several experimental studies[85,461,517,521,709,710,712,717,719–726,728,735–737,743,748] and the databases 3D-Footprint[696], NDB[697], and ProNIT[703]. DNA-bound protein displacement reactions were assembled describing which proteins each protein species is able to displace from the chromosome (see Table S3O). Functionally, displacement of DNA-bound proteins enables proteins to access the chromosome and fulfill their chromosomal replication or maintenance role. The affinity of RNA polymerase for each promoter was first reconstructed from the observed RNA composition of *E. coli*[393] (see Table S3U), the expression of each *M. genitalium* mRNA reported by Weiner *et al.*[569] (see Table S3W), the half-life of each *E. coli* mRNA reported by Bernstein *et al.*[602] (see S3Y, the observed amino acid composition of *M. gallisepticum*[870], and the *Mycoplasma* genetic code[53,168,607] (see Table S3X). Subsequently, the affinity of RNA polymerase for each promoter was fit to match the additional data used to train the 28 modeled cellular processes (see Section 1.3).

Computational Representation

This state represents the polymerization, winding, modification, and protein occupancy of each nucleotide of each strand of each copy of the *M. genitalium* chromosome, and the (de)catenation status of the two sister chromosomes following replication. List S2 summarizes the mathematical representation of the *M. genitalium* chromosome(s) including the size and type of each variable.

Mathematically, each quantity except winding, base and sugar-phosphate modification, protein occupancy and catenation, is represented as a 3-dimensional Boolean tensor. Winding is represented as a 3-dimensional real tensor which indicates the linking number density of each nucleotide. Base and sugar modification are represented as 4-dimensional tensors which indicate the chemical identity $l = \{1..M\}$ of each nucleotide, where $M = 722$ is the number of distinct metabolite species represented by the `Metabolite` state. Protein monomer and complex occupancy are represented as 4-dimensional tensors which indicate the identity, $l = \{1..B^m\}$ and $l = \{1..B^c\}$ respectively, of the protein bound at each nucleotide, where $B^m = 482$ and $B^c = 201$ are the numbers of distinct protein monomer and complex species represented by the `Protein Monomer` and `Protein Complex` states. (De)catenation is represented as a Boolean scalar.

List S2. Mathematical representation of nucleotide $i = \{1..L\}$ **of strand** $j = \{1..2\}$ **of chromosome copy** $k = \{1..2\}$.

Physical Property	Symbol	Size	Type
Polymerization	p_{ijk}	$L \times 2 \times 2$	Boolean
Winding	w_{ijk}	$L \times 2 \times 2$	Real
Modification			
Gap site	m^g_{ijk}	$L \times 2 \times 2$	Boolean
Abasic site	m^a_{ijk}	$L \times 2 \times 2$	Boolean
Sugar-phosphate	m^p_{ijkl}	$L \times 2 \times 2 \times M$	Boolean
Base	m^b_{ijkl}	$L \times 2 \times 2 \times M$	Boolean
Intrastrand cross link	m^c_{ijk}	$L \times 2 \times 2$	Boolean
Strand break	m^s_{ijk}	$L \times 2 \times 2$	Boolean
Holliday junction	m^h_{ijk}	$L \times 2 \times 2$	Boolean
Protein occupancy			
Monomer	b^m_{ijkl}	$L \times 2 \times 2 \times B^m$	Boolean
Complex	b^c_{ijkl}	$L \times 2 \times 2 \times B^c$	Boolean
Catenation	s	1×1	Boolean

Integration

The `Metabolite` state describes the identity of each modified base and sugar-phosphate. The `Protein Monomer` and `Protein Complex` states represent the total DNA-bound copy number of each protein monomer and macromolecular complex. For computational efficiency, the `RNA Polymerase` state redundantly represents the chromosomal location of each bound RNA polymerase. The mass of the chromosome(s), including all modifications and all DNA-bound proteins, is included in the cell mass calculated by the `Mass` state.

Ten processes access and modify the `Chromosome` state. The `Replication Initiation` process models DnaA DNA-binding and formation of the oriC DnaA complex which promotes replication initiation. The `Replication` process models bidirectional DNA polymerization from the oriC, continuously on the leading strand and discontinuously on the lagging strand, and Okazaki fragment ligation. The `Chromosome Segregation` process models sister chromosome decatenation following successful chromosome replication. The `Cytokinesis` process models cell division following successful decatenation. The `Chromosome Condensation` and `DNA Supercoiling` processes model SMC- and supercoiling-mediated chromosome compaction. The `DNA Damage` process models stochastic and radiation- and chemically-induced DNA damage. The `DNA Repair` process models three DNA repair pathways – base excision repair, nucleotide excision repair, and homologous recombination. The `DNA Repair` process also models methylation and restriction of MunI (MG184) restriction/modification sites. Finally, the `Transcriptional Regulation` and `Transcription` processes model the binding of transcription factors and RNA polymerases to promoters and the synthesis of RNA.

Initial Conditions
First, the `Chromosome` state is initialized with one supercoiled and fully methylated chromosome,

$$
\begin{aligned}
p_{ij1} &= 1 \;\forall\; i, j \\
p_{ij2} &= 0 \;\forall\; i, j \\
w_{ij1} &= (1+\sigma_{ss})\,\frac{L}{h} \;\forall\; i,\; j \\
w_{ij2} &= 0 \;\forall\; i, j \\
m^{g}_{ijk} &= 0 \;\forall\; i, j, k \\
m^{a}_{ijk} &= 0 \;\forall\; i, j, k \\
m^{p}_{ijk} &= 0 \;\forall\; i, j, k \\
m^{b}_{ij1} &= \begin{cases} 1 & i \in \text{MunI methylation sites},\; \forall\; j \\ 0 & \text{otherwise} \end{cases} \\
m^{b}_{ij2} &= 0 \;\forall\; i, j \\
m^{c}_{ijk} &= 0 \;\forall\; i, j, k \\
m^{s}_{ijk} &= 0 \;\forall\; i, j, k \\
m^{h}_{ijk} &= 0 \;\forall\; i, j, k \\
s &= 0,
\end{aligned}
$$

where $L = 580076\,\text{nt}$ is the length of the *M. genitalium* chromosome, $\sigma_{ss} = -0.06$ is the observed bacterial steady-state superhelicity[749], and $h = 10.5\ \text{nt}\,\text{lk}^{-1}$ is the observed equilibrium helical repeat length[879].

Second, the `Protein Monomer` and `Protein Complex` states initialize the total copy number of each protein species. Finally, the `Chromosome Condensation`, `DNA Supercoiling`, `Replication Initiation`, `Transcriptional Regulation`, and `Transcription` processes initialize the protein occupancy of the chromosome.

Fitting
The chemical composition of *M. genitalium* and the objective of the flux-balance analysis metabolic model were fit to match the mass and dNMP composition of the *M. genitalium* chromosome (see Section 1.3 and 2.5). The affinity of RNA polymerase for each promoter was fit to provide the gene products required by the 28 modeled processes to reproduce the observed 9 h *M. genitalium* mass doubling time (see Section 1.3). The oriC DnaA DNA-binding cooperativity was fit to match our intuition for the duration of the replication initiation cell cycle phase (see Section 3.19). The expression and activity of DNA gyrase and topoisomerse I were balanced to match the observed steady state superhelical density (see Section 3.6).

2.2 FtsZ Ring

Biology

`Cytokinesis` is the division of a cell into two daughter cells. *M. genitalium* contains a protein called FtsZ that assembles into long filaments that are implicated in cell pinching. These filaments bind to the membrane at the midline of the cell[217]. FtsZ is a GTPase, and when the GTPs bound in a membrane-bound FtsZ filament hydrolyze to GDP, the filaments bend constricting the membrane[421]. We use a geometric model of iterative filament bending modified from a model proposed by Li *et al.*[611]. See the `Cytokinesis` process and Schematic S6, for more details about this model. In summary, the FtsZ ring at the midline of the cell can exist in many states of various numbers of filaments in the bent and straight configurations. The purpose of this state class is to keep track of the state of the FtsZ ring across timesteps.

Reconstruction

In our model, the FtsZ ring is represented as a polygon of FtsZ filaments. As the circumference of the midline of the cell (C) decreases with time, the number of edges in the inscribed FtsZ polygon decreases. The FtsZ filaments are of a fixed length (l) of 40 nm[217].

Computational Representation

The number of edges in the polygon (E) is determined by the maximum number of edges of length l that can be inscribed in the given cell circumference. (The use of a fixed filament length is a simplification. See the `Cytokinesis` and `FtsZ Polymerization` processes for more details about the assumptions used.) This state class computes the number of edges in the FtsZ polygon using the inscribed polygon formula and the current cell midline diameter (d):

$$E = \frac{\pi}{\arcsin(\frac{l}{d})} \tag{S1}$$

The filaments may not perfectly inscribe the cell circumference, and so we round down, such that E only accounts for full filaments in the FtsZ ring.

The `Cytokinesis` process determines how many straight and/or bent filaments reside at each edge of the FtsZ polygon. At each edge, any of the following occupancies are possible:

- 1 straight filament
- 2 straight filaments
- 1 bent filament
- 2 bent filaments
- 1 bent filament and 1 straight filament
- 1 bent filament and 2 straight filaments

The `FtsZ Ring` state class keeps track of the filament occupancy of each edge. As the filament occupancy in a timestep is highly dependent on the occupancy at the previous timestep, the tracking of the state of the FtsZ ring is extremely important in the time evolution of `Cytokinesis`.

Integration

The `FtsZ Ring` state class reads the diameter of the cell's midline (d) from the `Geometry` state class.

The `Cytokinesis` process class reads the number of FtsZ ring edges (E) and the filament occupancy at each edge from the `FtsZ Ring` state class. The `Cytokinesis` process class updates the filament occupancy at each edge, and this is stored in the `FtsZ Ring` state class.

Initial Conditions

An FtsZ ring is not assembled at the beginning of the simulation. Therefore, no initialization steps are required for this state class.

2.3 Geometry

Biology

While *M. genitalium* divides by binary fission similarly to other bacterial species, its non-uniform shape, lack of cell wall, and lack of complete division machinery make its growth and division distinctive from most other bacteria. *M. genitalium* has a flask/pear-like shape, with a protruding adhesion structure called the terminal organelle. This flask-like shape is rather fluid due to the lack of a cell wall[872].

Reconstruction

Since the growth and division of *M. genitalium* are not well understood, we chose to model the cell as varying from a spherical to short rod shape. The cell is modeled as a cylinder with two hemispherical caps, and growth is modeled in the cylinder length. Once cell pinching commences at the midline of the cell, the shape and size of a "septum region" is also modeled. The `Geometry` state class calculates and keeps track of the physical shape of the cell including its width, length, volume, and surface area. This state also keeps track of the progress of cytokinesis, and the completion of cytokinesis is the trigger to end the entire simulation.

Computational Representation

The `Geometry` state uses a set of geometric equations to calculate the shape of the cell, given the width (w), density (ρ), and cytokinesis progress of the cell. The state houses the calculations of cell length (l), volume (V), surface area (SA), and progress of cell pinching. The geometric representation requires us to assume that the cell density and cell width are constant across the cell cycle. Various sources have indicated that the volumetric density of a cell remains constant overtime[874,875]. The cell density used is that of *E. coli*, which has been estimated as 1100 g/L[553]. Several sources have described the width of a rod-shaped bacterial cell as remaining approximately constant across the cell cycle and across cell divisions[873]. The cell width is calculated based on the initial cell mass (m_0) and density and the assumption that the cell is a sphere. The initial cell mass was fit to result in the measured *M. genitalium* cell width of 200 nm[225]. The general geometric equations to represent the shape of a cell are inspired by Domach *et al.*[305]. These geometric equations use the fixed width and fixed cell density assumptions to calculate all the other aspects of the cell geometry.

Cell Geometry Calculations

The mass (m) and density (ρ) enable calculation of the volume at all time points:

$$V = \frac{m}{\rho} \tag{S2}$$

Before cell pinching starts

The cell is modeled as a cylinder (length: l_c, diameter: w) with two hemispherical caps (diameter: w). We use the cell volume to calculate the cell length and surface area:

$$\begin{aligned}
V &= \overbrace{\frac{1}{6}\pi w^3}^{\text{2 hemispheres}} + \overbrace{\frac{1}{4}\pi w^2 l_c}^{\text{cylinder}} && \text{(S3)}\\
l &= \overbrace{w}^{\text{2 hemispheres}} + \overbrace{l_c}^{\text{cylinder}} && \text{(S4)}\\
&= w + \frac{4}{\pi w^2}\left(V - \frac{1}{6}\pi w^3\right) && \text{(S5)}\\
SA &= \overbrace{\pi w^2}^{\text{2 hemispheres}} + \overbrace{\pi w l_c}^{\text{cylinder}} && \text{(S6)}
\end{aligned}$$

After cell pinching starts

Once cell pinching commences at the midline of the cell, there is a "septum region" as well (Schematic S1). Here the cylinder length (l_c), is the combined length of the two cylinders. The septum length (s) is the length from the cell midpoint to the edge of the septum region. This septum length is calculated from the "pinched diameter" property that is calculated in the `Cytokinesis` Process. Each half of the septum volume is calculated as the area of two quarter circles (radius: s) and one rectangle (width: s, height: $w - 2s$), integrated around the midline cylindrically (see Schematic S2). We use the cell volume to calculate the cell length and surface area:

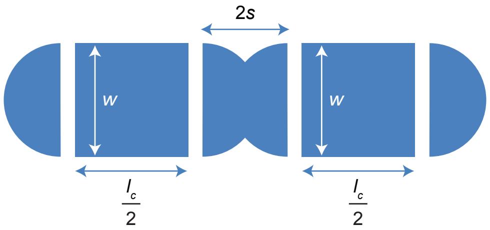
$$V = \overbrace{\frac{1}{6}\pi w^3}^{\text{2 hemispheres}} + \overbrace{\frac{1}{4}\pi w^2 l_c}^{\text{2 cylinders}} + \overbrace{\frac{s\pi}{2}\left(\left(8s^2 - 4sw + w^2\right) + s\pi\left(w - 2s\right) - \frac{4}{3}s^2\right)}^{\text{septum region}} \tag{S7}$$

$$l = \overbrace{w}^{\text{2 hemispheres}} + \overbrace{l_c}^{\text{2 cylinders}} + \overbrace{2s}^{\text{septum region}} \tag{S8}$$

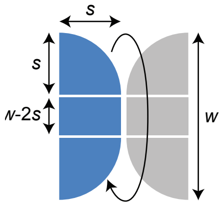
$$= w + \frac{4}{\pi w^2}\left(V - \frac{1}{6}\pi w^3 - \left(\frac{s\pi}{2}\left(\left(8s^2 - 4sw + w^2\right) + s\pi\left(w - 2s\right) - \frac{4}{3}s^2\right)\right)\right) + 2s \tag{S9}$$

$$SA = \overbrace{\pi w^2}^{\text{2 hemispheres}} + \overbrace{\pi w l_c}^{\text{2 cylinders}} + \overbrace{4\pi s\left(w - s\right)}^{\text{septum region}} \tag{S10}$$

Once the pinched diameter is zero, this state records that the cell has divided, and this determines the end point of the simulation.



Schematic S1. Model representation of cell geometry.



Schematic S2. Representation of volume of septum region.

Integration

The Geometry state class obtains the initial mass of the cell (to determine the cell width) from the Mass state class. It also obtains the mass of the cell from the Mass state class at all time points to determine the cell volume.

The Metabolism, Protein Activation, Replication Initiation, and FtsZ Polymerization process classes all obtain the cell volume from the Geometry state class. The Cytokinesis process class reads the pinched diameter from the Geometry state class, and sends back the updated pinched diameter.

Initial Conditions

The initial cell volume is calculated based on the initial cell mass (m_0) and density (ρ). The initial cell width is calculated from the volume and the assumption that the cell is a sphere. All of the other parameters describing cell geometry are evaluated from the mass, volume, density, and width.

2.4 Host

Biology

As discussed in Section 3.8, the *M. genitalium* terminal organelle is believed to mediate attachment to the human urogenital epithelium and enable its parasitic lifestyle. Upon attachment, lipoproteins are believed to activate host Toll-like receptors (TLRs) 1, 2, and 6, the host transcriptional regulator NF-κB, and ultimately the host inflammatory response. This state represents the instantaneous configuration of the human host. The `Host Interaction` process describes the host response to *M. genitalium*.

Reconstruction

Section 3.8 describes the reconstruction of the interaction of *M. genitalium* with its human host.

Computational Representation

This state uses six Boolean variables to represent the instantaneous configuration of the human host:

- Adherent – true if *M. genitalium* is attached to its human host and false otherwise.
- TLR 1 activation – true if TLR receptor 1 is active and false otherwise.
- TLR 2 activation – true if TLR receptor 2 is active and false otherwise.
- TLR 6 activation – true if TLR receptor 6 is active and false otherwise.
- NF-κB activation – true if NF-κB is active and false otherwise.
- Inflammatory response activation – true if the host inflammatory response is active and false otherwise.

Integration
None of the other 15 states directly interact with the `Host` state. The `Host Interaction` process completely determines the initial conditions and temporal dynamics of all six Boolean variables represented by this state.

Initial Conditions
As discussed in Section 3.8, this state is initialized to the steady-state of the host-interaction dynamic model.

2.5 Mass

Biology

Cells are composed primarily of water, DNA, RNA, and protein enclosed in a bilipid membrane. The `Metabolite`, `Chromosome`, `Rna`, `Protein Monomer`, and `Protein Complex` states represent the detailed molecular composition of *M. genitalium*. The `Mass` state calculates the total cell mass from those states. The total cell mass is used as a proxy for membrane surface area in several processes which are based on mathematical models originally developed outside our integrated whole-cell modeling framework without calculations of the membrane surface area.

Reconstruction

The mass and chemical composition of *M. genitalium* were reconstructed based on an extensive review of the primary literature (see Table S3I, S3AR-S3BE) and fit to match the 28 modeled cellular processes. First, the average initial *M. genitalium* cell mass (13.1 fg) and dry mass (3.93 fg) were self-consistently calculated assuming a spherical geometry with 200 nm diameter[555], 1.1 g ml^{-1} density[553], and 70% water composition by mass[554]. Note this is comparable to the 18.9 fg *M. genitalium* cell mass reported by Morowitz[869]. Second, the molecular composition of the dry mass was hierarchically reconstructed:

1. The *M. genitalium* dry mass was divided into eight classes – DNA/dNMPs, RNA/NMPs, protein/amino acids, lipids, carbohydrates, polyamines, vitamins & cofactors, and ions.
2. The fractional mass (see Table S3AS) and molecular composition (see Table S3AU-S3BE) of each class was reconstructed from the literature.
3. The complete molecular composition of *M. genitalium* was assembled (see Table S3I and S3AT).

4. The dry mass composition was fit to match the 28 modeled cellular processes as described below and in Section 1.3.

Computational Representation

This state calculates the total cell mass. The total cell mass is equal to the sum of that represented by the `Chromosome`, `Rna`, `Protein Monomer`, `Protein Complex`, `Transcript`, `Polypeptide`, and `Metabolite` states.

Integration

This state interacts with the `Geometry` state which calculates the cell shape, surface area, and volume from the calculated cell mass and observed density. None of the processes directly modify the `Mass` state. The `Metabolism` process uses the cell mass to bound the rates of exchange reactions. The `Replication Initiation` process models DnaA-ADP reactivation to DnaA-ATP as a function of membrane lipid mass. The `Metabolism` and `Replication Initiation` processes use the cell mass rather than the cell volume because these processes are based on mathematical models that were originally developed outside our whole-cell modeling framework without more detailed calculations of the membrane surface area.

Initial Conditions

The cell mass is hierarchically initialized. First, the total cell mass is initialized according to a normal distribution with mean 13.1 fg and standard deviation 0.66 fg. The mean initial cell mass was set to the reconstructed value. The variance was fit to match that of the predicted final cell mass following cell division. Second, the total cell mass is divided among individual metabolites according to the reconstructed chemical composition (see Section 2.7). Third, the `Chromosome`, `Rna`, `Protein Monomer`, `Protein Complex`, `Transcript`, and `Polypeptide` states are initialized as functions of the metabolite copy numbers, and the metabolite copy numbers are decremented to maintain the initial cell mass.

Fitting

The mean initial cell mass was fit to the reconstructed value. The variance was fit to match that of the predicted final cell mass following cell division.

Cellular composition is the balance of the production and usage of molecules by cellular processes. Consequently, it was necessary to reconcile the reconstructed cellular composition with that predicted by the 28 modeled cellular processes. See Section 1.3 for further discussion. First, the DNA dry mass fraction, including both chromosomal DNA and free dNTP, was increased 322% to be consistent with the predicted mass of the chromosome and the free dNTP pool. The predicted chromosome mass accounted for the observed chromosome sequence[182], predicted modifications (see `DNA Repair` process), and predicted time-average copy number (see `Replication`, `Replication Initiation`, and `Metabolism` processes). The predicted free dNTP pool accounted for the free dNTPs needed support the observed 9 h *M. genitalium* mass doubling time. Second, the RNA dry mass fraction, including both RNA and free NTPs, was increased 39% to be consistent with the amounts of m-, r-, s-, and tRNA needed to support the observed 9 h mass doubling time. The NMP composition of the RNA fraction was set equal to that predicted by the expression, sequence, and modification of each RNA species (see `Transcription`, `RNA Processing`, and `RNA Modification` processes). Third, the vitamin & cofactor and ion dry mass fractions were increased to match the amounts of vitamins, cofactors, and ions needed for protein folding and catalysis (see `Protein Folding` process). Finally, the protein dry mass fraction, including both proteins and free amino acids, was decreased 23% to offset the increased DNA and RNA mass fractions. The amino acid composition of the protein fraction was set equal to that predicted by the expression, sequence, and modification of each protein species (see `Translation`, `Protein Processing I`, `Protein Processing II`, and `Protein Modification` processes).

2.6 Metabolic Reaction

Biology

Cells grow by importing nutrients from the external environment and using those nutrients to construct macromolecules. The `Metabolism` process models the dynamics of the 645 transport and chemical reactions which provide the metabolic building blocks required for macromolecular synthesis and drive cellular growth. Table S3O lists the reconstructed transport and chemical reactions.

Reconstruction

The *M. genitalium* metabolic network was reconstructed based on extensive review of the primary literature, several databases, and the metabolic demands of all of the processes. See Section 3.10 for further discussion.

Computational Representation

This state records the instantaneous flux of each metabolic reaction in reactions per second as a floating point vector. This state also records the instantaneous cellular growth rate predicted by the `Metabolism` process in cells per second as a floating point scalar.

Integration
None of the other 15 states directly interact with the `Metabolic Reaction` state. The `Metabolism` process calculates the flux of each metabolic reaction and the total cellular growth rate using flux-balance analysis (FBA). The `Metabolism` process uses the predicted fluxes to update the copy number of each metabolite.

Initial Conditions
The `Metabolism` process initializes the flux of each metabolic reaction and the total cellular growth rate to a steady-state of the FBA model.

Fitting
The FBA objective was set to the reconstructed *M. genitalium* chemical composition (see `Mass` and `Metabolite` states and `Metabolism` process). The cellular growth rate was fit to match the observed 9 h *M. genitalium* mass doubling time using a modified version of minimization of metabolic adjustment (MOMA)[876] (see Section 1.3 and `Metabolism` process).

2.7 Metabolite

Biology

The *M. genitalium* model accounts for the dynamics of 722 distinct metabolites (see Table S3G) which serve many important functions in over 1,100 chemical reactions across three compartments – cytosol, membrane, and extracellular space. First, cells use nucleic and amino acids to synthesize DNA, RNA, and protein. Cells also use ions and other prosthetic groups to stabilize macromolecules. Second, cells use small molecule bonds and gradients to store energy and drive cellular processes. In particular, cells drive many energetically unfavorable reactions through hydrolysis of the high energy intermediates ATP and GTP. Third, cells use coenzyme functional moieties to facilitate chemical catalysis. Additionally, cells use small molecules such as Ca^{2+}, ATP, and (p)ppGpp for communication and regulation. Finally, cells use small molecule antibiotics to defend against predators and attack prey.

Reconstruction

The *M. genitalium* metabolite complement was indirectly reconstructed by reconstructing the chemical reactions of each of the 28 modeled cellular processes. Table S3G lists the 722 reconstructed metabolites.

Metabolite Physical Properties

The empirical formula and structure of each metabolite was curated based on an extensive review of the primary literature including two genome-scale metabolic models of *M. genitalium*[610] and *E. coli*[558], and the databases BioCyc[6], ChEBI[593], Delta Mass[589], FindMod[590], KEGG[113], LIPID MAPS[588], Modomics[267], PubChem[587], RESID[592], and UniMod[591]. The molecular weight, van der Waals volume, pI, logP, and logD of each metabolite was computed using ChemAxon Marvin[594]. Table S3G lists the physical properties of each metabolite.

Metabolite Regulatory Properties

The regulatory properties of several antibiotics were reconstructed from the primary literature and the database DrugBank[847]. See `Protein Activation` process for further discussion.

Cellular Chemical Composition

The molecular composition of *M. genitalium* was reconstructed based on an extensive review of the primary literature. See Section 2.5 for further discussion.

Extracellular Medium Chemical Composition

M. genitalium was cultured in complex *Spiroplasma* medium #4 (SP-4 medium)[878] for all experiments presented in this study. Accordingly, the chemical composition of the *in silico* external environment modeled that of SP-4 medium. Because the chemical composition of SP-4 medium is undefined, the composition of *in silico* medium was reconstructed based on the characterized composition of each SP-4 medium component[754–762] (see Table S3BJ), and supplemented with additional metabolites to support *in silico* growth. Addition of supplemental metabolites was guided by the `Metabolism` process and by the *M. pneumoniae* minimal medium defined by Yus *et al.*[614]. Table S3BI lists the composition of the *in silico* medium.

Computational Representation

This state represents the copy number of each metabolite in each of 3 compartments – cytosol, membrane, and extracellular space – as an integer matrix.

Integration

At each time step of the simulation, the `Simulation` object simulates an *M. genitalium* cell culture incubator which maintains the extracellular partial pressures of carbon dioxide and oxygen by setting the dissolved extracellular copy numbers of these gases. Table S3BI lists the simulated partial pressures of carbon dioxide and oxygen. To satisfy the assumption that each process is independent on the time scale of the 1 s simulation time step, and because many metabolite species participate in multiple processes, the `Simulation` object also allocates shared metabolites among processes at each time step. See Section 1.3 for further discussion.

Cytosolic- and membrane-localized metabolites are included in the cell mass calculated by the `Mass` state. The `Metabolism` process models the import of extracellular nutrients into the cytosol and membrane, the conversion of those metabolites into the precursors of DNA, RNA, protein, and lipids, and the export of byproducts into the extracellular space. The exchange rate of each metabolite of the flux-balance analysis metabolic model is limited by its extracellular copy number. 23 additional processes access and modify the `Metabolite` state, primarily drawing cytosolic metabolites to support macromolecule synthesis. Four processes do not directly interact with the `Metabolite` state: `Host Interaction`, `Macromolecular Complexation`, `Terminal Organelle Assembly`, and `Transcriptional Regulation`.

Initial Conditions

After the `Mass` state initializes the total cell mass, the `Metabolite` state initializes the total cytosolic and membrane copy numbers of each metabolite according to the calculated chemical composition of *M. genitalium* (see Section 2.7 and 2.7 and Table S3I). Additionally, after the `Geometry` state initializes the volume of the extracellular compartment, the `Metabolite` state initializes the extracellular copy number of each metabolite according to the reconstructed medium composition (see Section 2.7 and 2.7 and Table S3H). Following `Metabolite` state initialization, the `Chromosome`, `Rna`, `Protein Monomer`, `Protein Complex`, `Tran-`

`script`, and `Polypeptide` states initialize the copy number of each macromolecule species, and decrement the metabolite copy numbers to maintain the initial cell mass.

Fitting

The chemical compositions of *M. genitalium* and the extracellular medium were initially reconstructed based on the experimentally characterized compositions of *M. genitalium* and SP-4 medium. The cell and medium composition were both supplemented to support the metabolic demands of the 28 modeled cellular processes. See Section 1.3, 2.5, and 2.7 for further discussion.

2.8 Polypeptide

Biology

The `Polypeptide` state class keeps track of the progress of translation. In our model, binding of free ribosomes to mature mRNAs is determined by mRNA availability. Once bound, a polypeptide is sythesized at a rate of up to 16 amino acids per second[564]. Therefore, polypeptides may take multiple 1 second timesteps to be completed. The `Polypeptide` state class holds information about which mRNAs are ribosome bound, as well as ribosomal progress of translating a transcript across timesteps.

For reasons such as limited resources, a ribosome may stall resulting in an incomplete polypeptide. In such cases, a proteolysis tag (a short peptide added to the end of a nascent polypeptide) may be synthesized to mark the ribosome for release and the aborted polypeptide for degradation. This state also houses the progress of proteolysis tag synthesis and the amino acid sequences of the incomplete polypeptides that are to be degraded.

Reconstruction

The `Polypeptide` state class serves as a “support system” for translation, providing it with infomation that is required to translate polypeptides.

The state holds fixed information such as the length, tRNA sequence, and amino acid sequence of every *M. genitalium* monomer and the length, molecular weight, tRNA sequence, and amino acid sequence of every possible *M. genitalium* proteolysis tag.

Computational Representation

As the simulation progresses, this state class holds transient information such as the ribosome-bound mRNAs, the lengths of polypeptides that are being translated, and the sequences of aborted polypeptides.

The state can calculate information from the transient properties such as the counts of each amino acid in each polypeptide and the length of aborted sequences. Lastly, this state is able to calculate the weight of all nascent polypeptides as the sum of the weights of the amino acid components.

Integration

The `Translation` process class reads all of the fixed parameters describing polypeptides and proteolysis tags. It both reads and updates the time evolving parameters describing polypeptides and proteolysis tags. The `Protein Decay` process class reads the sequence and length of each aborted polypeptide.

Initial Conditions

The simulation begins in a state in which ribosomes are already bound to mRNAs and in the process of elongating. Each growing polypeptide is accounted for in the `Polypeptide` state class. No proteolysis tags exist at the start of the simulation.

2.9 Protein Complex

Biology

Protein complexes are used by almost all of the process classes in the system to perform their respective functions. A protein complex could be as small as transketolase, a dimer, or as large as pyruvate dehydrogenase that has 192 monomeric subunits. A complex may contain a mixture of protein and RNA subunits (e.g. ribosome), metabolites (e.g. DnaA protein bound to ATP), or ions (e.g. oxidized form of the protein thioredoxin). A complex can also exist in various locations within a cell including in the cytoplasm, membrane, terminal organelle, and bound to the chromosome. The `Protein Complex` state class holds information about complex composition, as well as the counts of each complex in the cell.

Reconstruction

The `Protein Complex` state class holds fixed parameters including the complex subunit composition, molecular weights, amino acid composition, half-lives, and localization. Some essential functions require a certain threshold abundance of particular protein complexes, and so this state class also holds the minimum expression of each complex.

Computational Representation

As complexes are produced by the Macromolecular Complexation process and other processes in the model, their counts are stored in this state. This state can also compute the dry weight of each protein complex in the cell.

Integration

List S3. Connections between the `Protein Complex` state class and other processes in the cell.

Connected Processes	Read from `Protein Complex`	Written to `Protein Complex`
`Macromolecular Complexation`	▪ Counts of protein complexes ▪ Complex subunit composition	▪ Updated counts of protein complexes
`Ribosome Assembly`	▪ Counts of ribosomes	▪ Updated counts of ribosomes
`Replication`	▪ Counts of replication complexes	▪ Updated counts of replication complexes
`Replication Initiation`	▪ Counts of DnaA complexes	▪ Updated counts of DnaA complexes
`Protein Decay`	▪ Counts of protein complexes ▪ Complex half lives ▪ Complex amino acid composition	▪ Updated counts of protein complexes
`Transcription`	▪ Minimum average expression	
Various other processes	▪ Protein complex counts to determine maximal enzyme activity	

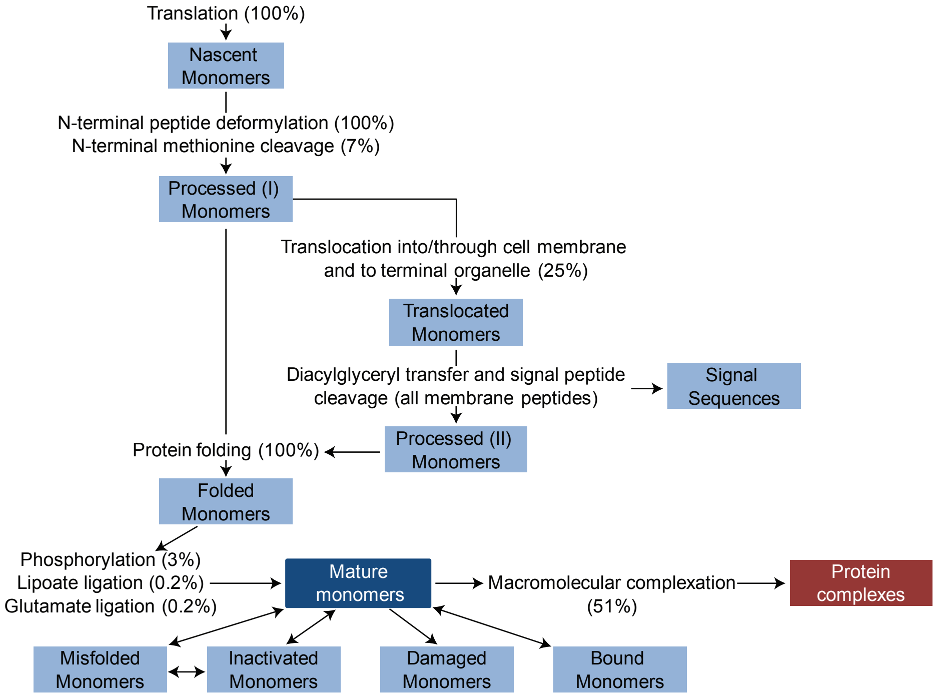
Initial Conditions

The system is initialized with a set of proteins. The determination of which proteins are expressed and at what quantities is determined randomly based on expected protein expression and expected total protein mass fraction.

2.10 Protein Monomer

Biology

Protein Monomers are the direct result of successful translation events. Upon `Translation`, a monomer undergoes various steps towards maturation including deformylation, translocation, folding, and phosphorylation. As a result, a monomer can exist in many forms (nascent, processed (I), translocated, processed (II), folded, and mature) as it moves through the maturation pipeline (See Schematic S3).



Schematic S3. Protein monomer forms diagrammed in the context of the maturity pipeline.

In addition to maturation, various processes can render a monomer non-functional, resulting in the misfolded, inactive, or damaged forms. Further, a mature monomer may be freely floating in the cytoplasm or bound to another molecule in the cell. For example, a translation factor may be bound to a mRNA, or a topoisomerase may be bound to a chromosome. The quantities of monomers in each of the maturation phases, functional forms, and non-functional forms are stored in this state class. Note that some functional enzymes are a complex of monomers, and these complexes are stored in a separate state class called `Protein Complex`.

Reconstruction

The main purpose of the `Protein Monomer` state class is to hold the counts and attributes of the monomeric species in the system.

This state class holds important information describing protein monomers such as their molecular weights, amino acids composition, length (in amino acids), and half-lives. A subset of monomers are translocated to different compartments in our model such as the membrane or terminal organelle. The `Protein Monomer` state class contains information about where each monomer localizes.

The `Protein Monomer` state class also houses information that is important for the fitting and initialization of the model. Some essential functions require a certain abundance of particular proteins. For example, the Cell Division process may require at least some threshold abundance of the division protein FtsZ for cell division to ever be possible. The sytem is then fit such that in an unperturbed state, at least this threshold amount of FtsZ is produced. This state stores the minimum expression of each monomer.

The current version of this model involves a stochastic generation of the initial protein abundances in the cell. The initial abundance of certain monomers is more rigid than others for the maintainace of a stable system that can grow and divide. The `Protein Monomer` state class stores the degree of variation we allow in the initial abundances of each monomer.

Computational Representation

The `Protein Monomer` state class holds the counts of the monomeric species in each of the forms shown in blue in Schematic S3.

Integration

List S4. Connections between the `ProteinMonomer` state class and other processes in the cell.

Connected Processes	Read from `Protein Monomer`	Written to `Protein Monomer`
`Translation`	▪ Counts of nascent monomers	▪ Updated counts of nascent monomers
`Protein Processing I`	▪ Counts of nascent and processed (I) monomers	▪ Updated counts of nascent and processed (I) monomers
`Protein Translocation`	▪ Counts of processed (I) and translocated monomers	▪ Updated counts of processed (I) and translocated monomers
`Protein Processing II`	▪ Counts of translocated and processed (II) monomers and signal sequences	▪ Updated counts of translocated and processed (II) monomers and signal sequences
`Protein Folding`	▪ Counts of processed (II) and folded monomers	▪ Updated counts of processed (II) and folded monomers
`Protein Modification`	▪ Counts of folded and mature monomers	▪ Updated counts of folded and mature monomers
`Macromolecular Complexation`	▪ Counts of mature monomers	▪ Updated counts of mature monomers
`Protein Decay`	▪ Counts of monomers ▪ Monomer half lives ▪ Monomer amino acid composition	▪ Updated counts of monomers ▪ Assignment of damaged monomers ▪ Assignment of misfolded monomers
`Protein Activation`	▪ Counts of mature and inactivated monomers	▪ Updated counts of mature and inactivated monomers
Various other processes	▪ Counts of mature and bound monomers ▪ Protein monomer counts to determine maximal enzyme activity	▪ Updated counts of mature and bound monomers

Initial Conditions
The system is initialized with a set of proteins. The determination of which proteins are expressed and at what quantities is determined randomly based on expected protein expression and expected total protein mass fraction.

2.11 Ribosome

Biology

Composition
Ribosomes are large ribonucleoproteins which synthesize polypeptides. The *M. genitalium* 70S ribosome is composed of two subunits – the 30S and 50S ribosomal subunits – which assemble on mRNA with assistance from initiation factors 1-3 (MG173, MG142, MG196). The 30S subunit is composed of 1 RNA and 20 protein monomer subunits. The 50S subunit is composed of 2 RNA and 32 protein monomer subunits. The 30S and 50S ribosomal subunits are believed to assemble in stereotyped patterns[660,661], and six GTPases – EngA (MG329), EngB (MG335), Era (MG387), Obg (MG384), RbfA (MG143), and RbgA (MG442) – have been associated with ribosomal subunit assembly[58,102,104,105,182,222,223]. The exact functions of the six GTPases are unknown.

Translation
Following 70S ribosome assembly, ribosomes synthesize amino acid polymers according to nucleic acid templates specified by mRNA and decoded by tRNA. Upon reaching the stop codon UAG, ribosome release factor (MG258) binds to the 70S ribosome, recognizes the stop codon, hydrolyzes the peptidyl-tRNA bond, and releases the terminal tRNA. Finally, elongation factor G (MG089) and initiation factor 3 dissociate the 30S and 50S ribosomal subunits, the mRNA, and the ribosome release factor. See Section 3.27 for further discussion of translation.

Stalled Ribosome Response
Upon prolonged translational stalling, tmRNA can displace both ribosome-bound tRNA and mRNA, leading to the generation of chimeric polypeptides containing an N-terminal mRNA-coded domain and a C-terminal tmRNA-coded degradation domain, or SsrA tag. Following translation termination, the protein degradation machinery recognizes the SsrA tag and degrades the chimeric polypeptide. See Section 3.27 and 3.12 for further discussion.

Computational Representation

The `Ribosome` state represents (1) the status – actively translating or stalled – of each 70S ribosome, (2) the mRNA, or tmRNA in the case of stalled ribosomes, species each 70S ribosome is bound to and translating, and (3) the position, in codons, of each 70S ribosome from the start codon. The status, bound (t)mRNA, and (t)mRNA position of each ribosome are implemented as scalar integer variables.

Integration
The `Polypeptide` state represents the sequence of each nascent polypeptide. The `Protein Complex` state represents the copy numbers of free 30S and 50S ribosomal subunits, and of mRNA-bound 70S ribosomes. The `Rna` state represents the copy number of each mRNA species and of tmRNA.

Three processes – `Translation`, `RNA Decay`, and `Protein Decay` – access and modify the `Ribosome` state. The `Translation` process models the formation of 70S ribosomes on mRNA, polypeptide synthesis catalyzed by 70S ribosomes, and 70S ribosome disassembly following translation termination. The `Translation` process also models ribosome stalling, tmRNA substitution, and SsrA degradation tag synthesis. Computationally, the `Translation` process predicts the state of each 70S ribosome, the mRNA or tmRNA species each 70S ribosome is bound to, and the elongation rate of each nascent polypeptide.

mRNA and tmRNA degradation events modeled by the `RNA Decay` process trigger early 70S ribosome disassembly resulting in incomplete polypeptides. Similarly, 70S ribosome degradation events modeled by the `Protein Decay` process result in incomplete polypeptides. Both processes decrement the copy number of mRNA-bound 70S ribosomes represented by the `Protein Complex` and `Ribosome` states.

The `Ribosome Assembly` process models the assembly of 30S and 50S ribosomal subunits from rRNA and ribosomal protein monomers. The `Protein Activation` process models the effect of antibiotics on the catalytic activity of 30S and 50S ribosomal particles.

Initial Conditions
After the `Rna` and `Protein Monomer` states initialize the total copy number of each RNA and protein monomer species and the `Ribosome Assembly` process initializes the total copy numbers of the 30S and 50S ribosomal subunits, the `Translation` process initializes the `Ribosome` and `Polypeptide` states. As described in Algorithm S4, the `Translation` process forms 70S ribosomes equal to the minimum of the 30S and 50S subunit copy numbers, and randomly positions each 70S ribosome on mRNA weighted by the copy number of each mRNA.

Algorithm S4 | `Ribosome` **and** `Polypeptide` **state initialization.**

```
Free 70S ribosomes ← min(free 30S, 50S ribosomal particles)
Decrement the copy numbers of free 30S and 50S ribosomal particles
foreach 70S ribosome $i$ do
    Select the mRNA of 70S ribosome $i$ weighted by the product of mRNA copy number and length
    Set the bound mRNA species of 70S ribosome $i$
    Select the position of 70S ribosome $i$ along the bound mRNA with uniform probability
    Set the status of 70S ribosome $i$ to actively translating
    Decrement the copy number of free 70S ribosomes
    Increment the copy number of bound 70S ribosomes
    Set the sequence of the nascent polypeptide corresponding to 70S ribosome $i$
```

2.12 RNA

Biology

Transcription leads to the production of nascent RNA that, once mature, may be used in various cell functions. mRNAs serve as a template for protein synthesis, rRNAs are a part of the ribosome structure, tRNAs carry amino acids to growing polypeptides, and sRNAs act as regulators. tmRNAs come into action when ribosomes stall on an mRNA during translation, helping recycle the stalled ribosome and adding a proteolysis tag to the incomplete polypeptide.

Different nascent RNAs undergo various steps towards maturation, including cleavage from polycistronic (transcription unit) to single RNA form, methylation, and thiolation (see Schematic S4). Maturation is carried out by the `RNA Processing` and `RNA Modification` process classes. There is an additional step for tRNAs which are coupled with amino acids by the tRNA Aminoacylation process.

In addition to the various forms an RNA can take in the maturation pathway, an RNA can also transition between various non-functional and functional forms. RNAs may exist with proteins in a macromolecular complex, and when the complex is marked for decay by the `Protein Decay` process class, its RNA subunits are marked as damaged in the `Protein Monomer` state class. Damaged RNAs are degraded by the `RNA Decay` process class. Further, a mature RNA may be freely floating in the cytoplasm or bound to another molecule in the cell. For example, an mRNA may be bound to a ribosome. This is accounted for in the the Bound RNA form. The quantities of RNA in each of these immature/mature and functional/non-functional forms is stored in this state class (see Schematic S4).

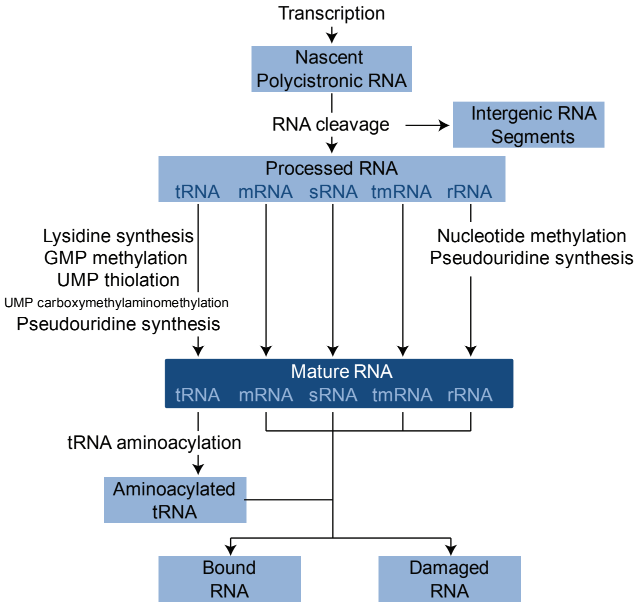
Reconstruction

This state class holds information about RNAs such as their weights, compositions, lengths, and localizations. It also holds important parameters for the maturation pathway of RNA, such as a number of matrices that map one form of RNA to another, and the gene composition of transcription units. Other parameters such as expected RNA half-lives and RNA weight fractions are included in this class, and are used to determine the RNA Polymerase-DNA binding parameters associated with gene expression.

Computational Representation

The `Protein Monomer` state class contains the number of each RNA species in each RNA form.

Other properties are calculated by this state class. The RNA decay rates are calculated from the experimentally measured half lives. The actual decay rates and RNA expression values used in the model are fit from the experimentally measure values such that the cell will double its mass and successfully divide by the end of the cell cycle.



Schematic S4. RNA forms diagrammed in the context of RNA maturation.

Integration

List S5. Connections between `ProteinMonomer` state and other processes in the cell.

Connected Processes	Read from `Rna`	Written to `Rna`
`Transcription`	▪ Counts of nascent RNAs	▪ Updated counts of nascent RNAs
`RNA Processing`	▪ Counts of nascent, processed, and intergenic RNAs	▪ Updated counts of nascent, processed, and intergenic RNAs
`RNA Modification`	▪ Counts of processed and mature RNA	▪ Updated counts of processed and mature RNA
`tRNA Aminoacylation`	▪ Counts of mature and aminoacylated tRNA	▪ Updated counts of mature and aminoacylated tRNA
`RNA Decay`	▪ Counts of mature, misfolded, and damaged RNA	▪ Updated counts of mature, misfolded, and damaged RNA
Various Processes	▪ Counts of mature and bound RNAs	▪ Updated counts of mature and bound RNAs

Initial Conditions

The system is initialized with a set of RNAs. The determination of which RNAs are expressed and at what quantities is determined randomly based on expected RNA expression and expected total RNA mass fraction.

2.13 RNA Polymerase

Biology

RNA polymerases are protein complexes that bind to gene promoters on the chromosomes and mediate the synthesis of RNA transcripts. This state class helps keep track of the conditions and chromosomal locations of RNA polymerases in the cell.

Reconstruction

An RNA polymerase in our model can exist in one of four conditions:

1. Free (not bound to DNA)
2. Non-specifically bound (bound to the DNA, but not to a specific gene promoter and not transcribing)
3. Specifically bound (bound to a specific gene promoter)
4. Actively transcribing (moving along a gene to produce an RNA)

This state class stores properties relating to RNA polymerases such as the expected probabilities of a polymerase being in the above 4 conditions.

Computational Representation

As the simulation progresses, this state class stores the condition of each RNA polymerase in the simulated cell. Transition between the conditions may involve RNA polymerase association and dissociation from the cell's chromosome(s). This state class holds the precise positions on the chromosome(s) where polymerases are bound. The `RNA Polymerase` state class also handles basic accounting. For example, upon an RNA polymerase decay event, a free polymerase is decremented. Further, this class records the premature release of RNA polymerases from the chromosome(s) and passes information about the aborted transcript to the `Transcript` state class.

Further, the `Transcription` process class requires the RNA Polymerase-promoter binding probability for each transcription unit. These probabilities may vary during the cell cycle due to the effects of other processes in the system, such as Transcriptional Regulation or DNA supercoiling. The fold changes to the base binding probabilities incurred by these other processes is stored in the `RNA Polymerase` state class.

This state class can also calculate the total number of RNA polymerases in each condition.

Integration

The `Chromosome` state class updates the chromosomal positions of RNA polymerases in the `RNA Polymerase` state class. The `Transcript` state class reads the aborted RNA sequences from and writes the updated aborted sequences to the `RNA Polymerase` state class.

The `Transcription` process class reads the RNA polymerase conditions and progress from and updates the RNA polymerase conditions and progress to the `RNA Polymerase` state class. The `Transcription` process class also reads the RNA polymerase condition transition probabilities and RNA polymerase binding probabilities from the `RNA Polymerase` state class. The `Transcriptional Regulation` and `DNA Supercoiling` process classes record the fold change in RNA polymerase binding probabilities to the `RNA Polymerase` state class.

Initial Conditions

NA polymerases are initialized as follows:

1. Each RNA polymerase is randomly assigned (with replacement) to one of the actively transcribing, specifically bound, non-specifically bound, or free states weighted by the expected occupancy of each state
2. Actively transcribing and specifically bound polymerases are randomly assigned to transcription units weighted by the transcription unit binding probabilities (P_{tu}).
3. Actively transcribing polymerases are randomly assigned to positions within the assigned segment of their assigned transcription unit (positions near the segment border are not allowed to prevent polymerases from being too close to each other) with uniform probability.
4. Non-specifically bound polymerases are randomly assigned to an accessible region on the chromosome.

2.14 Stimulus

Biology

Cells live in complex environments where they are exposed to physical and chemical stresses including antibiotics, radiation, heat and cold. Furthermore, manipulation of the external environment is a powerful tool for discovering new biology. This state represents the status of 10 properties of the external environment.

Reconstruction

Table S3F describes the reconstruction of the values of these 10 properties of the external environment.

Computational Representation

Specifically, this state represents the temperature in °C, six types of radiation in $\mathrm{W\,m^{-2}}$, $\mathrm{Gy\,s^{-1}}$, or $\mathrm{m^{-2}\,s^{-1}\,sr^{-1}}$, and three Boolean-valued stress conditions. Computationally, each property is implemented as a floating point scalar. Table S3F lists the values of these 10 properties used throughout each simulation.

Integration
This state does not directly interact with any of the other 15 states. The `Geometry`, `Metabolite`, `Protein Monomer`, `Protein Complex`, and `Host` states represent additional properties of the extracellular environment. The `Geometry` state represents the volume of the external environment. The `Metabolite` state represents the copy numbers of extracellular metabolites including antibiotics. The `Protein Monomer` and `Protein Complex` states represent the copy numbers of extracellularly localized proteins. The `Host` state represents six properties of the host urogenital epithelium.

Three processes – `Protein Activation`, `DNA Damage`, and `Metabolism` – access the `Stimulus` state. The `Protein Activation` process regulates the activity of four transcription factors as functions of temperature and three stress conditions (see Table S3Q). The `DNA Damage` process models the rates of several types of DNA damage as functions of radiation (see Table S3O). The `Metabolism` process models the generation of hydroxyl radicals from water and γ-radiation (see Table S3O). None of the processes modify the 10 properties of the external environment.

Initial Conditions
For this study the 10 properties of the external environment were initialized to the values listed in Table S3F, and were not modified during the simulation.

2.15 Time

Biology

The physical, chemical, and biological processes relevant to cell physiology span a wide range of time scales. To limit the scope of this study, we modeled *M. genitalium* on a 1 s time scale and averaged out the effects of faster processes. *M. genitalium* can be approximated as a well-mixed system at this time scale.

Computational Representation

This state represents the time elapsed from the start of the simulation in seconds as a single integer variable.

Integration
The `Time` state directly interacts with only three parts of the simulation:

- The `Simulation` linearly advances time in 1 s increments (See Section 1.1).
- If the cell has not yet divided, each simulation terminates at a predetermined maximum time of 50,000 s, approximately equal to 150% of the observed mean *M. genitalium* mass doubling time (See Section 1.1).

- The concentrations of extracellular metabolites and the values of extracellular stimuli depend on time (see Section 2.7 and 2.14).

None of the processes or other states depend directly on the Time state.

Initial Conditions
The Time state is initialized to zero and the other 15 states are initialized to the beginning of the cell cycle. Consequently, the Time state also represents the time elapsed from the beginning of the cell cycle.

2.16 Transcript

Biology

During Transcription, an RNA polymerase binds to a gene promoter and facilitates the synthesis of an RNA transcript. This state stores all of the information that pertains to nascent RNA transcripts and aids in the time evolution of Transcription.

Reconstruction

There are multiple fixed properties relating to the gene templates of transcripts that are essential for transcription and stored in the Transcript state. For each transcription unit, we store the 5' coordinate of the gene on the genome. This is the template for the 1st nucleotide in a growing transcript. In addition to this, we store the direction in which the template is read, towards or away from the origin. We also store the sequence and length of each transcription unit.

Computational Representation

Transcription proceeds at a maximal rate of 50 nucleotides per second[562,563]. Therefore, it may take several timesteps to synthesize a transcript, and the progress of the RNA polymerase along the transcription unit must be stored. This state stores the transcription unit to which each RNA polymerase is bound, including which chromosome it is bound to and the progress of the RNA polymerase along the transcription unit.

An RNA polymerase may stall in the Transcription process or be knocked off of a gene by another protein. In these cases, an incomplete transcript may result, and will be targeted for degradation by the RNA Decay process. This state holds the sequence of each aborted transcript, such that the nucleotides can correctly be accounted for upon degradation.

Multiple RNA polymerases may be bound to a given transcription unit, and this state can calculate the total number of polymerases on each unit. Finally, the Transcript state also calculates the dry weight of the nascent transcripts.

Integration
The Transcript state class reads about the existence of an aborted sequence from the RNA Polymerase state class. Upon RNA polymerase displacement, the Transcript state class updates the Transcript state in the Chromosome state class. It also records the start sites and directions of transcription units in the Chromosome state class.

The Transcription process class uses all of the fixed properties and time evolving properties housed in the Transcript state class.

Initial Conditions
An RNA polymerase may be initialized in the actively transcribing state. For all such polymerases, the growing transcript is accounted for in the Transcript state class.

Chapter 3

Cellular Process Methods

The whole-cell model is composed of 28 sub-models which span six major areas of cellular physiology: (1) transport and metabolism, (2) DNA replication and maintenance, (3) RNA synthesis and maturation, (4) protein synthesis and maturation, (5) cytokinesis, and (6) host interaction. The sub-models were modeled using different mathematics and trained using different experimental data. Each of the 28 functional processes, or cellular processes, represents a group of chemical reactions which transform chemical substrates into products using enzyme catalysts. Computationally, the inputs and outputs of each sub-model are the copy numbers of metabolites and macromolecules; the configurations of RNA, protein and DNA polymers; and the catalytic capacity and configurations of the enzymes which participate in each sub-model. This chapter provides detailed discussions of the mathematics and computational implementation of each cellular process.

Metabolism

The `Metabolism` process modeled the import of nutrients from the external environment and their conversion primarily into the metabolic building blocks required by the 27 other processes for macromolecule synthesis. Therefore, the `Metabolism` process served as the primary interface between the external environment and the 27 other processes, providing the nutrients required for each of the other processes and recycling and/or exporting the metabolic byproducts of the other processes. The `Metabolism` process was reconstructed primarily based on DNA sequence homology of *M. genitalium* to *E. coli* and a comprehensive metabolic model of *E. coli*[558]. The `Metabolism` process was modeled using FBA and trained using the observed growth rate of *M. genitalium* and the observed chemical compositions of *M. gallisepticum*[870] and *E. coli*[393]. The composition of the *in silico* growth medium was reconstructed based on the reported chemical composition of the individual components of *Mycoplasma* SP-4 medium, with additional metabolites added to support sustained *in silico* growth.

RNA Synthesis & Degradation

RNA synthesis and maturation was modeled by four processes: `Transcription`, `RNA Processing`, `RNA Modification`, and `tRNA Aminoacylation`. `Transcription` modeled the state – free, non-promoter bound, promoter-bound, or actively transcribing – of each RNA polymerase, transcription initiation at specific promoters, transcription elongation and NTP allocation among nascent transcripts, and transcription termination. The state of each RNA polymerase was modeled as a Markov chain and trained with the observed occupancies of each of the four modeled states[775]. Transcription initiation was modeled as a stochastic process and trained using the reconstructed expression and decay rates of each transcription unit[569,602]. The transcription unit organization of the chromosome was reconstructed from the observed operon organization of *M. pneumoniae*[418]. Because transcription termination is not well-characterized, termination was modeled as a deterministic process which proceeds to completion within the 1 s simulation time step if there is least one copy of each of the characterized transcription termination factors.

Transcriptional Regulation modeled the fold change effects of transcriptional regulators on the affinity of RNA polymerase for specific promoters. The Transcriptional Regulation sub-model was reconstructed using the database DBTBS[419]. Transcriptional Regulation was modeled as a stochastic process and trained using reported fold change effects.

Following transcription, non-coding transcripts are cleaved, modified, and aminoacylated. The RNA Processing sub-model modeled the cleavage of polycistronic non-coding RNA into individual non-coding RNA gene products. The RNA Processing sub-model was reconstructed primarily based on reported *E. coli* RNA cleavages[39,105,649] and the complement of RNA processing enzymes contained in the *M. genitalium* genome. The RNA Modification process modeled the modification of specific bases of specific non-coding RNAs and was reconstructed based on the observed complement of *E. coli* RNA modifications[45,47,55,56,64,65,105,560,651,885]. The tRNA Aminoacylation sub-model modeled the aminoacylation of tRNAs according to the observed *Mycoplasma* genetic code[53,168,607]. RNA Processing, RNA Modification, and tRNA Aminoacylation were modeled motivated by mass-action kinetics and parameterized using the observed kinetics of the RNA cleavage and modification enzymes.

RNA transcripts are hydrolytically cleaved into 10-30 nucleotide fragments by ribonuclease R, which in turn are further degraded into individual NMPs. The RNA Decay sub-model modeled the decay of each RNA species as a first-order Poisson process with rate parameters equal to the observed decay rates of *E. coli* RNA[602].

Protein Synthesis & Degradation

Protein synthesis and maturation was modeled by nine processes: Translation, Protein Processing I, Protein Translocation, Protein Processing II, Protein Folding, Protein Modification, Macromolecular Complexation, Ribosome Assembly, and Terminal Organelle Assembly. Translation modeled the assembly of 70S ribosomes on mRNA Shine-Dalgarno sequences, the polymerization of amino acids into polypeptides, and the allocation of aminoacylated tRNAs among the multiple active ribosomes. The Translation sub-model also modeled the role of tmRNA in the termination of stalled ribosomes as a rare stochastic event.

Following translation, polypeptides are deformylated, cleaved, translocated, folded, and modified before forming macromolecular complexes. Protein Processing I modeled the first steps in protein maturation: polypeptide deformylation and N-terminal methionine cleavage. Protein Translocation modeled integral membrane, lipoprotein, and secreted protein translocation into the membrane by the SecA translocase. Protein Processing II modeled lipoprotein diacylglyceryl transfer, and lipoprotein and secreted protein signal sequence cleavage. Protein Folding modeled the folding of polypeptides into compact three-dimensional structures. Protein Modification modeled the covalent modification of specific amino acids. Macromolecular Complexation modeled the formation of protein and ribonucleoprotein complexes. Ribosome Assembly modeled the role of several GTPases in the assembly of 30S and 50S ribosomal particles. Terminal Organelle Assembly modeled the observed hierarchical assembly of the protein content of the *M. genitalium* terminal attachment organelle.

The eight post-translational processing sub-models were reconstructed based on specific N-terminal methionine cleavages observed in *Shewanella oneidensis* MR-1[280], the predicted localization and signal sequence of each protein monomer (see Table S3AM-S3AO), the observed chaperone interactions of *E. coli*[388,389] and *B. subtilis*[391], the observed complement of *M. genitalium* and *M. pneumoniae* protein modifications[94,277,282,283,672], and the inferred subunit composition of each macromolecular complex (see Table S3AS). The eight post-translational processing sub-models were modeled motivated by mass-action kinetics and were parameterized using the reported kinetics of the post-translational processing enzymes.

The Protein Decay sub-model modeled the hydrolytic cleavage of proteins into 10-20 amino-acid-long fragments by protease La, as well as the cleavage of aborted polypeptides by protease FtsH. Similar to RNA Decay, Protein Decay was modeled as a first-order Poisson process. The half-life of each protein was predicted using the N-end rule[586]. Additionally, the Protein Decay sub-model modeled protein misfolding and ClpB chaperone-mediated refolding. Protein misfolding was modeled as a stochastic process. Protein refolding was modeled as a deterministic process governed by a single Boolean rule.

The `Protein Activation` process modeled the chemical regulation of protein activity. The `Protein Activation` process was reconstructed primarily based on the database DrugBank[847]. Because the exact mechanisms of protein chemical regulation are poorly characterized, `Protein Activation` was modeled as a Boolean network.

Chromosomal Replication & Maintenance

Seven processes modeled chromosomal replication, damage, and maintenance: `Replication Initiation`, `Replication`, `Chromosome Segregation`, `Chromosome Condensation`, `DNA Supercoiling`, `DNA Damage`, and `DNA Repair`. The `Replication Initiation` process modeled the recruitment of DNA polymerase to the oriC by the formation of a large DnaA complex at the R1-5 functional oriC DnaA boxes. `Replication Initiation` was reconstructed using the reported DnaA DNA-binding motif, and implemented similar to the *E. coli* replication initiation model developed by Messer[924].

The `Replication` process modeled bidirectional DNA replication, dissolution of the oriC DnaA complex, single stranded binding protein (SSB) binding to exposed single-stranded DNA, and Okazaki fragment ligation. Allocation of dNTPs among elongating DNA polymerases was modeled similarly to `Transcription` and `Translation`.

The `Chromosome Segregation` process modeled daughter chromosome segregation following chromosomal replication. Because `Chromosome Segregation` is poorly characterized, `Chromosome Segregation` was modeled as a single event governed by a simple Boolean rule.

`Chromosome Condensation` and `DNA Supercoiling` modeled DNA compaction mediated by structural maintenance of chromosome (SMC) proteins and topoisomerases. `Chromosome Condensation` was reconstructed based on the reported DNA density of SMC proteins[517] and SMC DNA-binding was modeled as a stochastic process. `DNA Supercoiling` was modeled as the balance of the competing winding and unwinding effects of topoisomerase I and DNA gyrase, and parameterized by the observed kinetics of topoisomerase I and DNA gyrase[502,750,922].

`DNA Damage` modeled spontaneous and chemically- and radiation-induced DNA damage as a stochastic process parameterized by the reported efficiencies of DNA damage. `DNA Repair` modeled four modes of DNA repair – direct damage repair (DDR), base excision repair (BER), nucleotide excision repair (NER), and homologous recombination double strand break repair (HR-DSBR) – as well as the *M. genitalium* MunI restriction/methylation (R/M) system. DDR, BER, NER, and HR-DSBR were modeled as stochastic processes parameterized by their observed kinetic rates. R/M was reconstructed based on the reported MunI DNA-binding motif and modeled as a stochastic process.

Cytokinesis

Following chromosomal replication and segregation, *M. genitalium* divides by binary fission. According to the Li *et al.* hypothesis[611], *M. genitalium* divides by iteratively pinching its the septal membrane using the filamentous protein FtsZ. The `FtsZ Polymerization` process modeled the formation of septal FtsZ rings. `FtsZ Polymerization` is implemented using the ordinary differential equations described by Surovtsev *et al.*[164]. The `Cytokinesis` process modeled the iterative contraction of successively smaller FtsZ septal rings. The `Cytokinesis` sub-model was parameterized using the observed rate of FtsZ-GTP hydrolysis[421].

Cell Cycle

Three *M. genitalium* cell cycle phases were modeled: (1) pre-replication, or replication initiation, (2) replication, and (3) cytokinesis. All simulations reported in this study were initialized to the beginning of the replication initiation phase. Starting at the beginning on each simulation, the `Replication Initiation` process modeled the formation of the oriC DnaA complex, ultimately resulting in the recruitment of DNA polymerase to the replication origin and the start of DNA replication. Following DNA polymerase recruitment to the replication origin, the `Replication` process modeled DNA replication. After chromosomal replication the Chromosome Segregation process modeled daughter chromosomal segregation. Finally, the FtsZ Polymerization and Cytokinesis processes modeled the formation and contraction of FtsZ septal rings, ultimately resulting in cell division.

Host Interaction

The Host Interaction process modeled the interaction of *M. genitalium* with its host human urogenital epithelium. The Host Interaction process was reconstructed based on the observed composition of the protein content of the *M. genitalium* terminal organelle[408,409] and the reported *M. genitalium*-host interactions[88,176,194,313,842,844]. Because the interaction of *M. genitalium* with its human host is poorly characterized, Host Interaction was implemented as a Boolean model.

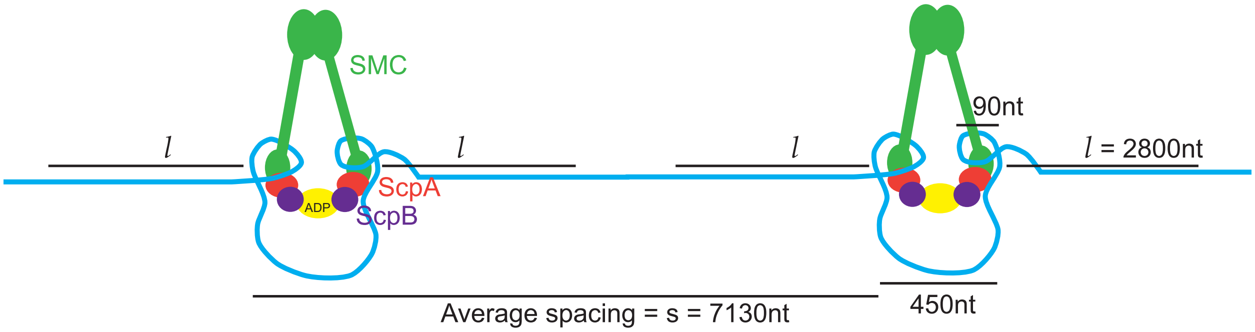
3.1 Chromosome Condensation

Biology

Bacterial chromosomes are compacted 10^4-fold in volume in order to physically fit inside of a cell[921]. Compaction is also necessary to support various cellular processes including chromosome replication, cell division, and chromosome segregation to opposing poles of the cell[921]. Bacteria employ several mechanisms to compact DNA including "clamping" of the DNA by structural maintenance of chromosome (SMC) proteins, DNA supercoiling, macromolecular crowding, and charge neutralization[921]. *M. genitalium* has the machinery to perform all of these levels of DNA compaction, and we explicitly model DNA Supercoiling in a separate process class and DNA clamping by SMC complexes here in the Chromosome Condensation process class.

Reconstruction

SMC complexes consist of an SMC core protein (Smc: MG298) and segregation and condensation proteins A and B (ScpA: MG213, ScpB: MG214). Together, the SMC complex is a "V" shaped structure (with a head and two legs) that induces positive supercoils in double stranded DNA[848]. The complexes are believed to work with a lock and key mechanism in which first DNA is looped around the legs of the SMC complex, and then an ATP is bound between the two tails to lock the SMC complex in place. The complexes bind and clamp the DNA causing many loops in the DNA and DNA compaction. The loops around each leg occupy 90bp. A loop of about 450bp forms between the two SMC complex legs[517,521]. Further, it has been inferred that there is about one SMC complex bound for every 7130bp of DNA (See Schematic S5)[517]. All of the parameters used in the Chromosome Condensation process class are described in List S6.



Schematic S5. SMC complex occupation of the DNA.

List S6. Fixed parameters used in the `Chromosome Condensation` process class.

Parameters	Value	Symbol	Source
Average separation of SMC complexes on the chromosome(s)	7130 nt	s	[517]
SMC complex threshold spacing	2800 nt	l	Fit
SMC DNA footprint	630 nt		[517]
SMC arm DNA footprint	90 nt		[517]
SMC inter-arm loop length	450 nt		[517]

Parameter Assignment
Given that the DNA footprint of an SMC complex is much less than its average spacing, if we simply applied a $1/s$ probability of each base being bound at each timestep, over time given an SMC abundance we would see SMCs bound at intervals much less than s. the SMC complex threshold spacing parameter (l) prevents this phenomenon by inhibiting SMC complex binding where the gap between existing SMC complexes on the DNA is already much less than the desired spacing. Biologically, this parameter represents the notion that SMC complex binding is not only determined by SMC complex abundance, but also the writhe and physical properties of the DNA that prevent SMC complex to bind too close to each other. This parameter was fit such that the average SMC complex spacing in the model is 7130 nt. The fit of this parameter was assessed by various tests to assure an average SMC spacing of 7130 nt.

Computational Representation

This module models the contribution of SMC proteins to chromosome condensation. SMC-related DNA condensing could be considered along with the effects of topoisomerases, but we model the effect of the topoisomerases by tracking the effect of each topoisomerase activity on the DNA linking number (in the DNA supercoiling process). Since SMCs do not act by strand-passing events (the causing of a nick in the DNA, enabling two double stranded DNA regions to pass through each other, and the re-ligation of the DNA), and since we do not know the exact effect of SMC complex activity on the DNA linking number, we consider SMC condensation as compacting the DNA at a different level and model it independently of the DNA linking number calculations in the DNA Supercoiling process. Macromolecular crowding and charge neutralization are not presently modeled.

SMCs are bound at an average spacing (s) of 7130 bases[517]. Since it is unknown whether SMC complexes bind to specific DNA motifs, at each timestep, we bind SMC complexes to random positions on the chromosomes. The binding is weighed by calculated probabilities (P) of a free SMC complex binding to each accessible double-stranded DNA base. The probability distribution is calculated as a step function: bases within a threshold distance (l) from another SMC complex have a zero probability of being bound, and bases beyond a threshold distance, l, from other SMC complexes bind at a probability of $1/s$. SMC complex disassociation from the DNA occurs upon interaction with other DNA binding proteins and is handled by the `Chromosome` state class.

Integration
The `Chromosome Condensation` process class reads from and writes to the `Chromosome` state class. It reads in the regions of DNA that SMC complexes can bind to, and writes back the positions of SMC complexes on the chromosome(s).

Initial Conditions
Before the start of the simulation, we iteratively run the SMC complex binding calculations described above, to bind SMC complexes to the first chromosome until a steady state is reached, that is no more SMC complexes can be bound.

Dynamic Computation
At each timestep, we perform the following algorithm:

1. Calculate the maximum number of SMC complexes that can bind in the given timestep.

$$\text{SMCBindingLimit} = \min\left(\text{Number of free SMC complexes}, \text{Number of availableATPmolecules}\right)$$

2. Bind SMCs
   (a) Query the Chromosome state to determine the free bases where SMC complexes can bind. Accessible bases are $> l$ bases away from any DNA bound SMC complexes.
   (b) Given that the probability of binding each accessible base is,

$$P_{\text{Binding Accessible Base}} = \frac{1}{s - 2l}$$

   randomly choose sites to bind based on this probability up to the SMCBindingLimit
   (c) Form an SMC-ATP complex, and hydrolyze the ATP to ADP.
      i. Decrement a free SMC
      ii. Increment and decrement free metabolites for ATP hydrolysis
   (d) Bind the SMC-ADP complex to chromosome (facilitated by the `Chromosome` state)

3.2 Chromosome Segregation

Biology

DNA replication produces two chromosomes that are catenated, or connected like links of a chain. Before cell division can take place, these chromosomes need to migrate to opposing sides of the cell and be decatenated. Chromosome segregation in *M. genitalium* is not well understood, but is believed to result from both entropic factors and enzymatic activity[849,850].

Reconstruction

The `Chromosome Segregation` process class assumes that the chromosomes entropically segregate during replication as described in Bloom *et al.* and Jun *et al.*[849,850]. That is, it is unfavorable for the two copies of replicated DNA to exist too close to each other, thus as the replication fork moves, the replicated DNA starts to migrate towards the poles. Enzymatic segregation activity is also modeled. The five chromosome segregation proteins are a nucleotide binding domain (CobQ/CobB/MinD/ParA: MG470), an MraZ protein (MraZ: MG221), a GTP binding protein (Era: MG387), a GTPase (Obg: MG384), and topoisomerase IV (ParE: MG203, ParC: MG204). The other segregation protein, FtsZ is accounted for in the `Cytokinesis` process class. These proteins assist in the chromosomal migration towards the cell poles and decatenate the chromosomes following the completion of replication. However, there is no detailed understanding of the function of these proteins, and much fewer proteins are present in *M. genitalium* than that required for the detailed mechanisms for segregation described for other bacterial species. Further, the specific kinetics, and metabolic costs of these proteins are not known.

Note that topoisomerase IV is also used in the `DNA Supercoiling` process class, as it is known to relieve coils that form just downstream of the replication loops. Here it performs a decatenation function, the unlinking of the two chromosomes.

All of the parameters used in the `Chromosome Segregation` process class are described in List S7.

List S7. Fixed parameters used in the `Chromosome Segregation` process class.

Parameter	Value	Symbol	Source
GTP cost of chromosome segregation	1 GTP	E_{seg}	Value unknown
Superhelical density tolerance	± 0.1	σ_{tol}	Value unknown
Equilibrium superhelical density	-0.06	σ	[922]

Parameter Assignment

The energetic cost of segregation has not been experimentally characterized. The energetic cost is set to a nominal value of 1 GTP per segregation event. The superhelical density tolerance is also unknown, so the allowed range of DNA superhelical density was set to be quite wide.

Computational Representation

This process models enzymatically catalyzed sister chromosome separation and decatenation. Entropically driven sister chromosome separation is not well understood and is not presently modeled.

Because the molecular biology of chromosome segregation is not well understood, we have chosen to implement this process as a simple Boolean process: the chromosomes are regarded as segregated immediately after the following four conditions are met:

- The chromosome is replicated
- The chromosomes are properly supercoiled within a given tolerance of superhelical density
- There is at least one free and functional molecule of each of the five segregation proteins, and
- There are enough available GTP molecules

Note that Glass *et al.* gene essentiality study suggests that the *cobQ*/*cobB*/*minD*/*parA* gene is non-essential, but we model this gene as essential because we don't know its specific function and how the other segregation proteins compensate in its absence.

Integration

List S8. State classes connected to the `Chromosome Segregation` process class.

Connected States	Read from state	Written to state
`Chromosome`	▪ Is chromosome replicated? ▪ Are chromosome superhelical densities within tolerance? ▪ Are chromosomes segregated?	▪ Chromosomes are segregated

Initial Conditions
The chromosome state is initialized with 1 chromosome with superhelical density within the acceptable tolerance.

Dynamic Computation
At each timestep, the following conditional statement is performed: If

- The chromosome is replicated
- The superhelical density is in the range: $\sigma \pm \sigma_{\text{tol}}$
- There is at least one free and functional molecule of each of the 5 segregation proteins
- Available GTP $\geq E_{seg}$

Then, mark the chromosome as segregated.

3.3 Cytokinesis

Biology

Cytokinesis, the division of the cell into two separate daughter cells, is the final step in the cell cycle. The major step of cytokinesis is the pinching of the cell membrane in a certain location (typically the midline of the cell) until the membrane is separated and division is complete. Typically, a membrane-bound ring of a tubulin homologue forms at the cell membrane around the pinching site, and the localization and assembly of this ring requires a set of accessory proteins[217]. Both contractile forces of the ring filaments, and external forces by the accessory proteins contribute to the pinching of the cell membrane[217].

Reconstruction

M. genitalium has the tubulin homologue, FtsZ (MG224), but does not contain the accessory proteins required for cell division in other bacterial species. Therefore, cytokinesis in *M. genitalium* is not fully understood. However, Li *et al.* propose an "iterative pinching" model by which a cell can divide using only FtsZ[611]. Their proposed mechanism invokes the formation of a contractile Z ring along the interior surface of the midline of the cell membrane. The ring is comprised of GTP-activated FtsZ polymer filaments that bend when their GTP is hydrolyzed. The bending draws the membrane-bound ends of the filaments together, thereby constricting the cell membrane. Many such pinching events result in a complete constriction of the cell. *M. genitalium* cytokinesis is thus a cycle of filament binding, bending, and dissociation.

Lluch-Senar *et al.* suggest that *M. genitalium* cell division can occur in the absence of FtsZ through motility[796]. We chose to use the Li *et al.* model because we do not have a descriptive model of *M. genitalium* motility and FtsZ is known to be an essential gene in *M. genitalium.*

All of the parameters used in the `Cytokinesis` process class are described in List S9.

List S9. Fixed parameters used in the `Cytokinesis` process class.

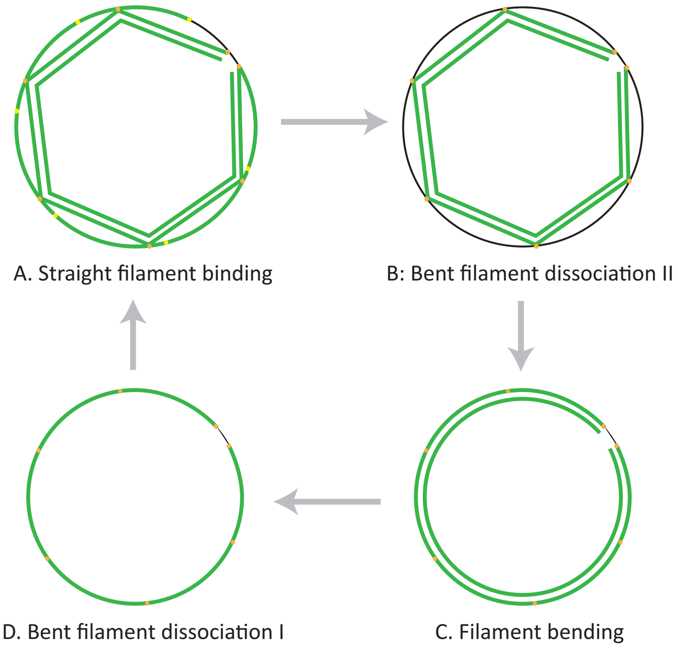
Parameters	Value	Symbol	Source
Length of an FtsZ filament	40 nm	L	[217]
Number of FtsZ subunits in an FtsZ filament	9 subunits	S	[217]
Rate of FtsZ-GTP hydrolyzing to FtsZ-GDP	$0.15\,\mathrm{s}^{-1}$	k_{hyd}	[421]
Rate of FtsZ polymers binding to the membrane	$0.7\,\mathrm{s}^{-1}$	k_{bind}	Fit
Rate of FtsZ polymers dissociating from the membrane	$0.7\,\mathrm{s}^{-1}$	k_{unbind}	Fit

Parameter Assignment
The kinetic rates of FtsZ filaments binding and dissociating from the membrane, k_{bind} and k_{unbind}, were uncharacterized. Therefore, the rates were set to be rapid such that their values under typical protein expression do not significantly limit the progression of cytokinesis.

Computational Representation

We represent cytokinesis using a modified version of the Li *et al.* model[611]. There are four main steps to our iterative pinching model, as seen in Schematic S6. First straight FtsZ-GTP filaments bind inside the cell membrane, such that they form an inscribed polygon (Schematic S6A). Next the GTP molecules in these filaments hydrolyze to GDP, causing the filaments to bend. Since the filaments are cell membrane bound, this bending pinches the membrane inwards. The filaments bend just enough to approximately form a new circle. The new circumference of the pinched region is now smaller, and equal to the perimeter of the inscribed polygon (Schematic S6C). The precise timing of the binding, hydrolysis, and dissociation of the contractile rings allows us to maintain the contraction progress across iterative pinching steps. After hydrolysis, one of the two bent filament rings is dissociated. The other remains, to maintain the progress of cell pinching (Schematic S6D). Now a new set of straight filaments can bind and form a new polygon that will be smaller than the previous (Schematic S6A). Upon binding, the residual ring of bent filaments can dissociate (Schematic S6B). The cycle repeats, resulting in smaller and smaller pinched circumferences, until the cell has divided.

We adapted the Li *et al.* model into our model framework as follows. First, we implemented cytokinesis as a process that is only initiated upon chromosomal segregation. Next, the Li *et al.* model only explicitly involves a single set of filaments binding at each edge during the binding stage. We have modified this such that two sets of filaments bind (Schematic S6A). Multiple filament sets are required to maintain the progress of membrane constriction. Indeed, Li *et al.* propose that the FtsZ filaments are more abundant and that a new ring of filaments attaches to stabilize the membrane and maintain progress before the depolymerization event. Osawa *et al.* also report that the FtsZ ring is 3-9 filaments thick depending on the bacterial strain[455]. Using the lower bound, since *M. genitalium* is a very small bacterium, we assume that the FtsZ maintains a thickness of up to 3 filaments. Our modified version of the Li *et al.* model requires two sets of straight FtsZ filaments to inscribe the diameter of the cell. Then up to one set of filaments can depolymerize while two


A. Straight filament binding
B: Bent filament dissociation II
C. Filament bending
D. Bent filament dissociation I

Schematic S6. Algorithm to bind, bend, and dissociate FtsZ filaments to pinch cell.

new sets of straight filaments bind the diameter of the cell. Only once the two new sets of straight filaments have bound, can the second bent set of filaments depolymerize.

The output of this process is an indication of the profess of cell pinching, including the determination of when the cell has successfully split into two daughter cells. Complete cell division is also the trigger to end the entire simulation.

Integration

List S10. State classes connected to the `Cytokinesis` process class.

Connected State	Read from state	Written to state
`FtsZ Ring`	▪ Number of filaments bound to each edge, and whether they are bent or straight ▪ Calculation of number of polygon edges in current cell diameter	▪ Updated number of filaments bound to each edge, and whether they are bent or straight
`Geometry`	▪ Pinched diameter of the cell ▪ Filament length parameters (L, S)	▪ Updated pinched diameter of the cell
`Chromosome`	▪ Has chromosome segregated?	

Initial Conditions
Initialization steps are not required for this process, as Cytokinesis does not take place until late in the cell cycle. Initialization of FtsZ polymers is handled by the `FtsZ Polymerization` process class.

Dynamic Computation
At each timestep we perform our modified version of the Li. *et al.* algorithm[611]. This involves binding a pair of full length (9mer) FtsZ polymers along the circumference of the midplane of the cell. When these filaments hydrolyze and bend we use circle and arc formulas to calculate the new smaller circumference of the cell. Depending on the state of the FtsZ ring, one or more of the following five steps is performed.

1. Straight filament binding (Schematic S6A): If cytokinesis has just initiated (triggered by Chromosome Segregation) or if the previous FtsZ ring has hydrolyzed and undergone dissociation I, attempt to bind

two straight filaments at each polygon edge, forming a regular polygon that inscribes the pinched cell circumference:

   (a) Determine the number of filaments to bind, based on the number of edges on the polygon that inscribes the pinched cell circumference. The number of edges (numEdges) is determined using the pinched diameter of the cell calculated in the previous timestep (D_{prev}), and the length of the FtsZ filament (L):

$$\text{numEdges} = \left\lfloor (\frac{\pi}{\sin^{-1}\left(\frac{L}{D_{\text{prev}}}\right)} \right\rfloor$$

   (b) Bind straight filaments such that each edge has a random chance of being bound and the probability of binding each of two filaments is equal to k_{bind}
   (c) Upon each binding event, update the state of the FtsZ ring

2. Bent filament dissociation II (Schematic S6B): If there was a previous cycle and at least one straight filament has bound each polygon edge, then dissociate the bent filament from each edge.
   (a) Randomly dissociate bent FtsZ-GDP filaments such that the probability of dissociating each filament is equal to k_{unbind}.
   (b) Dissociate unbound FtsZ-GDP filaments into FtsZ-GDP monomers.
   (c) Upon each dissociation event, update the state of the FtsZ ring.
3. Filament bending (Schematic S6C): If all polygon edges have two filaments bound and all residual bent filaments have dissociated, bend the edges of the newly completed polygon. When the filaments bend, their length does not change, and the amount of bending is only sufficient to form a new circle. The new circumference is therefore equal to the old polygon's perimeter. Each fragment is now an arc. For simplicity, all of the GTP molecules in the pair of filaments at a particular edge hydrolyze at the same time. In this version of the model, all filaments are of a fixed length. When they are joined end-to-end to form a regular polygon, the polygon may not fully inscribe the entire circumference. This remaining portion of the circumference does not bend when the polygon does. It is preserved and accounted for in the next iteration.
   (a) Randomly determine whether to bend the filaments, such that the probability of bending each edge is k_{hyd}. Bending at an edge can only take place if there are sufficient water molecules available to hydrolyze all of the GTP molecules in the two filaments.
   (b) Update the state of the the FtsZ ring and change the bent filament from FtsZ-GTP to FtsZ-GDP.
   (c) If all the edges have been bent, calculate the new pinched diameter of the cell:

$$\begin{aligned}\text{Diameter}_{\text{AfterHydrolysis}} &= \frac{\text{circumference}}{\pi} \\ &= \frac{\text{numEdges} \times L}{\pi}\end{aligned}$$

   Adjust this diameter to account for the polygon not perfectly inscribing the cell circumference by adding back the length of the old perimeter not accounted for by the inscribed polygon.

4. Bent filament dissociation I (Schematic S6D): If all the filaments have been bent, dissociate one bent filament from each polygon edge. The other ring must remain to preserve pinching progress by maintaining the new smaller circumference.
   (a) Randomly dissociate bent FtsZ-GDP filaments such that the probability of dissociating each filament is equal to k_{unbind}.
   (b) Dissociate unbound FtsZ-GDP filaments into FtsZ-GDP monomers.
   (c) Upon each dissociation event, update the state of the FtsZ ring.
5. Termination: If the pinched diameter is smaller than the length of one filament, conclude cytokinesis.

3.4 DNA Damage

Biology

DNA is not absolutely stable. Rather, DNA is susceptible to spontaneous modification including base loss and deamination. DNA is also susceptible to modification by many exotic agents including UV-A and UV-B radiation, α-, β-, and γ-radiation, oxygen, hydroxyl, carbon, and nitrogen radicals, and alkylating agents. These diverse DNA modification modes generate equally diverse DNA structures including missing and modified sugar-phosphates, missing and modified nucleobases, intra- and inter-strand cross links, and strand breaks. Table S3O lists the modeled causes and types of DNA damage.

DNA damage is an evolutionarily important source of genetic variation. However, damaged DNA has also been shown to deleterious to many processes including transcription and replication[780–783]. Consequently, cells employ dedicated DNA repair machinery. See Section 3.5 for further discussion. The exact effect of DNA damage on physiology is not well characterized. Tornaletti and Hanawalt have extensively reviewed the effects of several specific DNA modifications on transcription[781–783].

Reconstruction

DNA damage modes were reconstructed based on a review by Lindahl and Barnes[462] and the primary literature[461,465–468,471–474,476,479,480,490–493,497,501,504,506,507]. Table S3O lists the reconstructed causative agent, resultant base configuration, metabolite stoichiometry, and kinetic rate of each DNA damage reaction.

Computational Representation

Mathematical Model
Because DNA damage is rare, this process models each DNA damage reaction as an independent Poisson process. The rate, r_i, of each spontaneous DNA damage reaction i is given by the observed specific rate k_i. The rate, r_i, of each DNA damage reaction i caused by radiation j is given by the product of the experimentally observed specific rate, k_i, and the radiation flux, s_j,

$$r_i = k_i s_j. \tag{S11}$$

Chemically-induced damage reactions were reconstructed, but not modeled because *M. genitalium* is not typically cultured with DNA damaging agents.

Integration
The `Chromosome` state represents the copy number, modification status, and protein occupancy of each chromosome. The `Metabolite` state represents the copy number of each metabolite. The `Stimulus` state represents the fluxes of six types of radiation: α-, β-, and γ-particles, electrons, protons, and UV-A and UV-B.

The `Metabolism` process models the generation of hydroxyl radicals. The `DNA Repair` process models four DNA repair pathways: direct damage reversal, base excision repair, nucleotide excision repair, and homologous recombination. The `DNA Repair` process also models DNA methylation at restriction/modification (R/M) sites. All DNA modifications except methylations of R/M sites are assumed to impede protein DNA-binding.

Initial Conditions
The `Chromosome` state initializes one chromosome, fully methylated at R/M sites and otherwise unmodified. Several other processes including `Transcription` initialize the protein occupancy of the chromosome.

Dynamic Computation
Algorithm S5 outlines the implementation of the DNA damage model.

Algorithm S5 | DNA damage simulation. See the Mathematical Model section above and List S2 for definition of the mathematical notation.

```
Input: $m_i$ copy number of metabolite $i$
Input: $M_{ij}$ stoichiometry of metabolite $i$ in reaction $j$
Input: $z_i^g$, $z_i^a$, $Z_{\bullet i}^p$, $Z_{\bullet i}^b$, $z_i^c$, and $z_i^s$: final base configuration resulting from reaction $i$
foreach DNA modification reaction $i$ do
    Calculate base modification rate
    switch trigger of reaction $i$ do
        case spontaneous
            $r_i \leftarrow k_i$
        case radiation
            $j \leftarrow$ index of radiation trigger
            $r_i \leftarrow k_i s_j$
    foreach base $j$ in strand $k$ of chromosome $l$ susceptible to reaction $i$ in a random order do
        if insufficient metabolic resources to support reaction $i$ ($\exists\ j$ s.t. $m_j < -M_{ji}$) then
            break
        if poissonRand($r_i$) $\geq 1$ then
            Update the configuration of base $j$ of strand $k$ of chromosome $k$
            $m_{jkl}^g \leftarrow z_i^g$
            $m_{jkl}^a \leftarrow z_i^a$
            $m_{jkl\bullet}^p \leftarrow Z_{\bullet i}^p$
            $m_{jkl\bullet}^b \leftarrow Z_{\bullet i}^b$
            $m_{jkl}^c \leftarrow z_i^c$
            $m_{jkl}^s \leftarrow z_i^s$
            Update metabolite copy numbers: $m \leftarrow m + M_{\bullet i}$
```

3.5 DNA Repair

Biology

DNA is susceptible to several intrinsic and extrinsic modes of damage (see Section 3.4). Because damaged DNA is deleterious to many processes including transcription and replication[780–783], organisms have evolved specialized machinery to detect and repair damaged DNA. Eisen and Hanawalt have shown that the *M. genitalium* genome contains the DNA damage sensor DisA and four DNA repair pathways: direct damage reversal (DDR), base excision repair (BER), global genomic nucleotide excision repair (GG-NER), and homologous recombination double strand break repair (HR-DSBR)[891]. In addition, *M. genitalium* employs the type II restriction/modification (R/M) system MunI to selectively degrade foreign DNA.

Reconstruction

Damage Recognition

M. genitalium employs only one dedicated machine – DisA (MG105) – to recognize DNA damage. Bejerano-Sagie *et al.* have shown that *B. subtilis* DisA recognizes strand breaks, base damage, and cross links[513]. The mechanism of DisA damage recognition is not well understood. DisA is believed to signal DNA damage through the absence of the second messenger cyclic-di-AMP. In the absence of DNA damage, DisA cyclizes ATP to cyclic-di-AMP. In the presence of DNA damage, DisA is believed to bind DNA, adopt a catalytically inactive conformation, and via an unknown mechanism possibly involving cyclic-di-AMP and the transcriptional regulator Spo0A, delay sporulation. *M. genitalium*, however, does not have most of the downstream signaling machinery associated with *B. subtilis* DisA-dependent sporulation delay.

Direct Damage Reversal
DNA ligation is the only DDR pathway employed by *M. genitalium*[891]. DNA ligation is catalyzed by DNA ligase LigA (MG254) by a NAD-dependent mechanism. DNA ligase repairs single strand breaks as well as ligates DNA synthesized during replication, BER, NER, and HR.

Base Excision Repair
The primary role of BER is to repair small patches of oxidized nucleobases[487,488,718]. *M. genitalium* BER repairs single damaged nucleobases. *M. genitalium* does not undergo long patch BER because *M. genitalium* does not have a flap endonuclease.

M. genitalium BER repairs DNA via a four step mechanism. First, glycosylase Fpg (MG498) or Ung (MG097) hydrolytically cleaves the glycosidic bond between the damaged nucleobase and the DNA backbone, creating an abasic site. Second, two parallel subpathways cleave the phosphodiester bonds 3' and 5' to the abasic site, introducing a gap site. The first subpathway begins with cleavage of the 3' phosphodiester bond site by AP lyase Fpg and ends with cleavage of the 5' phosphodiester bond by 5'-deoxyribosephosphodiesterase Nfo. The second subpathway begins with cleavage of the 5' phosphodiester bond by 5'-AP endonuclease Nfo (MG235) and ends with cleavage of the 3' phosphodiester bond by 3'-(deoxyribose-5'-phosphate) lyase DnaN (MG001). Third, DNA polymerase DnaN restores the missing base using the template provided by the opposite strand. Finally, DNA ligase ligates the inserted base.

Nucleotide Excision Repair
The primary role of NER is to repair bulky distortions in the shape of the DNA helix of up to 12-13 bases in length caused by UV radiation and nitric oxide. *M. genitalium* NER provides global protection against small DNA lesions. *M. genitalium* does not undergo transcription-coupled NER because *M. genitalium* does have the transcription-repair coupling factor *mfd*. Compared to BER, NER employs less specific endonucleases and consequently has broader repair capacity.

M. genitalium NER repairs DNA via a four step mechanism. First, UvrABC (MG421, MG073, MG206) identifies a DNA lesion and cleaves the phosphodiester bonds 6-7 bases 3' and 5' to the lesion. Second, helicase PrcA excises the cleaved DNA. Third, DNA polymerase DnaN (MG001) restores the missing bases using the template provided by the opposite strand. Finally, DNA ligase ligates the inserted base.

Homologous Recombination Double Strand Break Repair
HR repairs double strand breaks caused by ionizing radiation and stalled replication forks. HR also repairs strand gaps generated by the interaction of replication forks with unrepaired lesions[797]. *M. genitalium* has a very reduced complement of recombination repair proteins. Recombination repair is modeled as an eight step process:

1. Initiation: 5'-3' exonuclease removes dNMPs from the 5' ends of the strand break, leaving 3' overhangs of at least 8 bases[526]. No traditional initiation gene has been identified in *M. genitalium.* We assumed that polI-like 5'-3' exonuclease (PolA, MG262) is the *M. genitalium* HR initiator.
2. Strand exchange: RecA (MG339) catalyzes the formation of a Holliday junction between one of the damaged 3' overhands and the undamaged homologous chromosome.
3. Polymerization: DnaN (MG001) polymerizes DNA guided by the undamaged chromosome template.
4. Ligation: LigA (MG254) ligates the newly produced DNA.
5. Second strand exchange: RecA catalyzes the formation of a second Holliday junction.
6. Holliday junction migration: RuvA and RuvB widen the distance between the two strand cross over points by moving the Holliday junctions to the preferred sequence 5'-`[AT]TTN[GC]`-3'[532].
7. Resolution: RecU nicks the DNA at the cross over points, creating four single strand breaks.
8. Ligation: LigA ligates the four strand breaks.

Additionally, HR is believed to be critical for *M. genitalium* antigenic variation[797].

Restriction/Modification
Organisms employ R/M systems to distinguish foreign from self DNA. These systems maintain self DNA in a fully methylated configuration by methylating self DNA during chromosomal replication and cleaving

unmethylated foreign DNA which has not been exposed to self DNA methylases. *M. genitalium* contains a reduced R/M system consisting of the type I DNA recognition subunit EcoD (MG438) and the type II methylase MunI (MG184). EcoD is the DNA binding domain of a type I multimeric methylation and restriction complex which recognizes the sequence 5'-TCARTTC-3'. Because *M. genitalium* contains only the DNA recognition subunit and not the methylation and restriction subunits, we do not model type I R/M. MunI methylates the third base of both strands of the palindromic sequence 5'-CAATTG-3', producing N^6-methyladenine. Because *M. genitalium* does not contain a separate type II endonuclease and because type II R/M systems are often monomeric, we assumed MunI also has restriction activity.

Metabolism of Damaged Nucleo-bases, -sides, and -tides

The metabolic byproducts of DNA repair are salvaged by oligonucleases. However, the mechanism of *M. genitalium* oligonucleotide salvage is not well understood. No gene has been identified which cleaves DNA oligonucleotides. Candidate oligonucleotide salvage genes include *pcrA* (MG244), *mgpA* (MG190), and *polA* (MG262). Of these genes, we assumed that *mgpA* is responsible oligonucleotide salvage. Individual damaged nucleotides are further metabolized and exported. See Section 3.10 for further discussion.

Reaction Stoichiometry & Kinetics

The DNA specificity and geometry, stoichiometry, and kinetics of each DNA repair reaction were reconstructed based on an extensive review of the primary literature[72,82,84,96,195,487,488,511–516,525,526,532,538,542,652,653,707,708,714,718,719,733,742,743,780]. Table S3O lists the reconstructed stoichiometry and kinetics of each reaction. Table S3D lists the reconstructed DNA specificity and geometry of each DNA repair reaction.

Computational Representation

Mathematical Model

This process models four DNA repair pathways and methylation and restriction of MunI R/M sites. Because DNA repair is not well understood on the genomic level, this process makes several simplifying assumptions. First, this process represents each step in DNA repair, methylation, and restriction as a separate reaction, and assumes that each reaction is independent. Consequently, the rate of each reaction is determined only by the configuration of the DNA. Second, this process assumes that the mean arrive rate, v_i, of each reaction i is independently limited by (1) the copy numbers of intracellular metabolites, m_j, and (2) the DNA repair enzyme copy numbers, e_j. Based on these assumptions, the function form of v_i is given by

$$v_i = \min\left(\left\lfloor \frac{\overbrace{m_j}^{\text{metabolites}}}{M^s_{ji}}_{j} \right\rfloor, \left\lfloor \overbrace{\texttt{poissonRand}\left(\frac{e_j}{K_{ji}}\Delta t\right)_{j}}^{\text{enzymes}} \right\rfloor, \right) \tag{S12}$$

where M_{ji} is the stoichiometry of metabolite j in reaction i, $M^s = \max(0, -M)$ is the negative part of M, K_{ji} is the experimentally observed catalytic rate of enzyme j in reaction i, and $\Delta t = 1\,\text{s}$ is the simulation time step.

This process also stochastically models the binding of free DisA to DNA lesions.

Integration

The Chromosome state represents the copy number, modification status, and protein occupancy of each chromosome. The Protein Monomer and Protein Complex states represent the free and DNA-bound copy numbers of each protein species. The Metabolite state represents the copy number of each metabolite.

The DNA Damage process models spontaneous and chemical- and radiation-induced DNA modification. The Metabolism process models DisA diadenylate cyclase activity and modified nucleotide catabolism.

Initial Conditions
The `Chromosome` state initializes one chromosome, modified only at R/M sites. Several other processes including `Transcription` initialize the protein occupancy of the chromosome.

Dynamic Computation
Algorithm S6 outlines the implementation of the DNA repair model.

Algorithm S6 | DNA repair simulation.

```
Repair, methylate, and restrict DNA
foreach reaction $i$ in a random order do
    foreach base $j$ of strand $k$ of chromosome $l$ susceptible to reaction $i$ in a random order do
        if sufficient enzymatic and metabolic resources to support reaction $i$ then
            Execute reaction $i$
            Update DNA configuration
            Update metabolite copy numbers
            Decrement enzymatic capacity

Bind DisA to damaged DNA
while there is free DisA and at least 1 DisA-accessible DNA lesion do
    Let $i, j, k \leftarrow$ represent the base, strand, and chromosome of a DisA-accessible DNA lesion
    Bind DisA to base $i$ of strand $j$ of chromosome $k$
```

3.6 DNA Supercoiling

Biology

DNA naturally exists at a certain level of helicity, and this level of helical density is important for the DNA's stability, its ability to fit in the cell, and its ability to bind proteins[693]. DNA supercoiling can also have an effect on RNA transcription rates as the helicity of the DNA may make given genes more or less accessible[920].

Reconstruction

M. genitalium has three topoisomerase proteins: DNA gyrase, topoisomerase I, and topoisomerase IV. These proteins transiently break a DNA strand to wind (topoisomerase I) or unwind (topoisomerase IV, gyrase) the DNA. The opposing actions of the topoisomerase enzymes help maintain a stable level of DNA helicity[694]. This is especially important during replication because while the replication loops progress along the chromosome unwinding the DNA, the coils that previously existed in the DNA persist and accumulate downstream of the replication loop. This over-coilied region is relieved by the actions of topoisomerases[923].

All of the enzymes and parameters used in the `DNA Supercoiling` process class are described in List S11 and List S12.

List S11. Enzymes and complexes used in the `DNA Supercoiling` process class.

Enzyme/Complex	Composition	Gene Name(s)	DNA Footprint (nt)
DNA gyrase	(2) MG003, (2) MG004	*gyrB*, *gyrA*	140
Topoisomerase IV	(2) MG203, (2) MG204	*parE*, *parC*	34
Topoisomerase I	(1) MG122	*topA*	19

List S12. Fixed parameters used in the `DNA Supercoiling` process class.

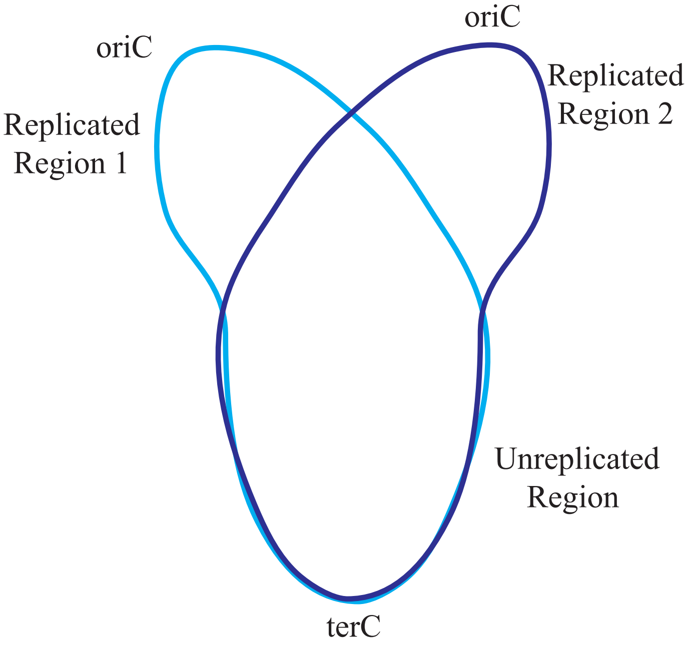
Parameter	Value	Symbol	Source
Supercoiling			
Bases per turn in relaxed DNA	10.5		
Equilibrium superhelical density	-0.06	σ_0	
Supercoils introduced by 1 gyrase strand passing event	-2	δ_{gyr}	[695]
Supercoils introduced by 1 topoisomerase I strand passing event	1	δ_{topI}	[502]
Supercoils introduced by 1 topoisomerase IV strand passing event	-2	δ_{topIV}	[922]
Strand passing events rate per gyrase	$1.2\,\text{s}^{-1}$	k_{gyr}	[922]
Strand passing events rate per topoisomerase I	$1\,\text{s}^{-1}$	k_{topI}	[502]
Strand passing events rate per topoisomerase IV	$2.5\,\text{s}^{-1}$	k_{topIV}	[750]
ATP hydrolyzed per gyrase-catalyzed strand passing event	2	E_{gyr}	[751]
ATP hydrolyzed per topoisomerase I-catalyzed strand passing event	0	E_{topI}	[922]
ATP hydrolyzed per topoisomerase IV-catalyzed strand passing event	2	E_{topIV}	[922]
Superhelical denisty above which gyrase acts	-0.1	T_{gyr}	[922]
Superhelical denisty above which topoisomerase I acts	0	T_{topI}	[502]
Superhelical density below which topoisomerase II acts	0	T_{topIV}	[922]
Gyrase logistic function steapness parameter	100	L_{gyr}	Fit
Topoisomerase I losgistic function steapness parameter	-100	L_{topI}	Fit
Number of bases in the chromosome	580076		
Enzyme DNA footprints	See List S11		[693–695]
Processivity			
Mean time gyrase stays bound to the DNA	45 s		[922]
Supercoiling's Effect on Gene Expression			
Slopes of linear fit of gyrase σ-gene expression data	42.8		[920]
Slopes of linear fit of topoisomerase I σ-gene expression data	-7.4		[920]
Slopes of linear fit of topoisomerase IV σ-gene expression data	1.1		[920]
y-intercepts of linear fit of gyrase σ-gene expression data	3.57		[920]
y-intercepts of linear fit of topoisomerase I σ-gene expression data	0.56		[920]
y-intercepts of linear fit of topoisomerase IV σ-gene expression data	1.07		[920]
σ bounds for applying linear fit of gene expression data	-0.08, 0.07	σ_{lower}, σ_{upper}	Fit

Parameter Assignment
The logistic function parameters (L_{gyr}, L_{topI}) were fit such that the equilibrium superhelical density, σ_0, could be maintained while still allowing both gyrase and topoisomerase I to act at the equilibrium σ_0. Linear fits were approximated for gene expression data experimentally measured at varying σ. The upper and lower bounds of σ for applying the linear fit of the σ-gene expression data (σ_{lower}, σ_{upper}) were set in there range where a linear fit was appropriate for the data.

Computational Representation

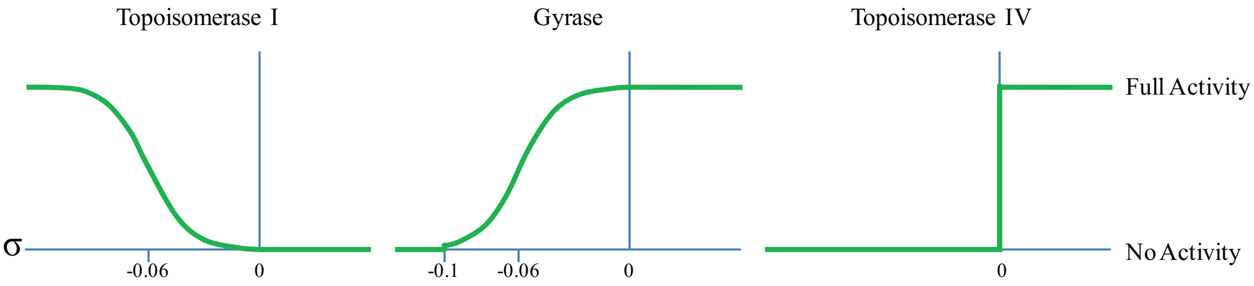
The supercoiling of the DNA is quanitfied using a metric known as the DNA linking number, LK. The LK_{relaxed} is defined as (# of base pairs)/10.5, where 10.5 is the number of bases per turn in a relaxed double helix. The LK_{current} may deviate from the LK_{relaxed} due to the actions of topoisomerases and the progression of replication. The ΔLK is defined as the difference between the current level of DNA supercoiling and the relaxed level of DNA supercoiling. DNA gyrase and topoisomerase IV each require 2 ATP to induce 2 negative supercoils each time they act[236]. Topoisomerase IV induces positive supercoils relatively rarely and its positive coiling activity is therefore not considered in our model[236]. Topoisomerase I acts to induce positive supercoils[502]. Gyrase, topoisomerase IV, and topoisomerase I act at rates of 1.2, 2.5, and 1 strand passing events per second respectively[502,750,922].

The helicity of different regions on the DNA are modeled separately. Before replication, the entire chromosome is represented as one region. During replication, two additional regions are defined and accounted for: the replicated DNA upstream of the replication loops on each of the two chromosomes (see Schematic S7). During replication, the unreplicated region of DNA (region downstream of the two replication loops) is shrinking in terms of number of bases (see Schematic S7), and as a result, the LK_{relaxed} decreases. However, as the DNA replicative helicases unwind the DNA they push the existing coils into this region, and there are more coils for fewer bases. This over-coiled region has a LK_{current} that is too high, and must be brought down towards the LK_{relaxed}. Gyrases and topoisomerases help bring the DNA back to the relaxed state by inducing negative supercoils. It is assumed that the newly replicated DNA is made in the relaxed state: $LK_{\text{current}} = LK_{\text{relaxed}}$, but the supercoiling enzymes act in this region as well, helping maintain equilibrium helicity. It is essential that the ΔLK in the region downstream of the replication loops be brought down to 0 by the end of replication. After replication is complete, only the two replicated regions exist (2 separate chromosomes). While in general, DNA supercoiling involves the regions described above, DNA damage and other processes can cause gaps in the DNA that result in additional regions.



Schematic S7. Regions of varying superhelical density on the replicating chromosome.

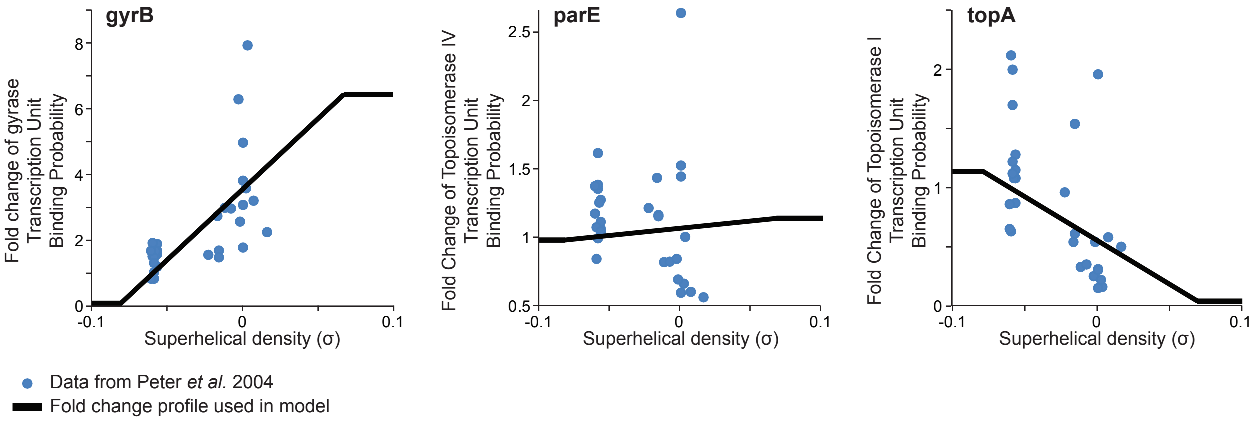
The superhelical density of a DNA region, σ, is defined as the $(LK_{\text{current}} - LK_{\text{relaxed}})/LK_{\text{relaxed}}$. Experimentally it has been shown that the steady state superhelical density, σ_0, is -0.06[923]. The activity of topoisomerases depends on the σ of the DNA[922]. Although there is likely a more complex relationship between the DNA supercoiling and enzyme activity, enzyme activity is modeled as a set of step functions and logistic functions (see Schematic S8). It is known that the topoisomerases are unable to act beyond certain σ thersholds (gyrase can only act above T_{gyr}, topoisomerase IV can only act above T_{topIV}, and topoisomerase I can only act below T_{topI}), so the activities of the topoisomerases outside of the thresholds are set to zero[922]. However, the exact enzyme activity profiles within these bounds is unknown. Fitting a logistic function within the thresholds of gyrase and topoisomerase I allows the maintainance of σ approximately near σ_0. Topoisomerase IV is not active near σ_0 and therefore acts more rarely than the other topoisomerases. As a result, there was no need to assume a logistic profile for its activity. The proportion of full activity for each topoisomerase was obtained from the profiles. This proportion was multiplied by the maximal activity to determine the number of strand passing events for each topoisomerase for a given timestep.



Schematic S8. Enzyme activity profiles.

Further, the model keeps track of enzyme processivity, or how long an enzyme can remain bound to the DNA and continue to act. Topoisomerase IV and DNA gyrase are highly processive and may perform several strand-passing events before falling off of the DNA. Topoisomerase I is not highly processive, it acts on the DNA and falls back off right away.

Finally, it has been shown that there is an effect of supercoiling on transcription rates, and experimental data in *E. coli* compares superhelical density to fold change of gene expression[920]. While expression fold changes at different superhelical densities have been calculated for many *E. coli* genes, none have been measured for *M. genitalium*, and therefore only the effects of supercoiling on the expression of the 5 supercoiling genes *gyrB*, *gyrA*, *parC*, *parE*, and *topA*, are included. These genes exist in 3 transcription units. Peter *et al.* have *E. coli* data for the fold change of expression (microarray data) of each of these genes at various values of σ (ranging from σ -0.06 to 0.02) at various experimental conditions[920]. Despite the large variation of data in a narrow σ range, we have decided to use linear fits of the data, as they fit the general trends in the data: increased helicity leads to decreased or increased gene expression. The linear fits are constrained such that the fold change is 1, at equilibrium σ, -0.06. The linear fits are extrapolated between the σ range of -0.08 to 0.07, where all the fold changes remain above 0. Outside of this range, a constant fold change of expression is estimated. While there is a separate set of data for each of the 5 genes, the `Transcription` process class requires a single probability of transcription for each transcription unit. The data for the first gene in each transcription unit is used (*gyrB* and *parE*). *topA* is transcribed with genes that are not supercoiling related. The expression of those genes (MG119, MG120, MG121) is also affected by supercoiling. The general trend is that gyrase and topoisomerase IV will have an increased expression when σ is higher than the equilibrium, and that topoisomerase I will have a higher expression when σ is lower than the equilibrium. The profiles of fold change in promoter binding probabilities for varying σ can be seen in Schematic S9.



Schematic S9. Fold change in the probability of RNA polymerase-gene promoter binding for three transcription units.

Integration

List S13. State classes connected to the DNA Supercoiling process class.

Connected State	Read from state	Written to state
Chromosome	▪ Accessible positions on the DNA for topoisomerases to bind ▪ Start positions of transcription units affected by supercoiling ▪ Linking number, length, and start sites of regions of DNA (considered independently in terms of superhelical density)	▪ Positions where topoisomerases are bound ▪ Fold change of transcription unit-RNA polymerase binding probabilities ▪ Updated linking number of regions of DNA

Initial Conditions
At the start of the simulation, the entire chromosome exists in a single double stranded region. The linking number of this region is set such that the superhelical density is at equilibrium, -0.06. All of the existing gyrase are bound to randomly designated positions around the chromosome. The transcription probabilities of the affected transcription units are adjusted according to the superhelical density.

Dynamic Computation
The activities of DNA gyrase, topoisomerase I, and topoisomerase IV, and the progress of replication loops can all affect the linking number (and superhelical density) of the DNA. The `DNA Supercoiling` process class, evaluates the effects of all the free topoisomerases in random order, determines what regions they can bind in, and randomly binds them to large enough open positions on the DNA. The linking number is then adjusted based on the actions of all the bound enzymes, and the usage of ATP is determined. The processivity of gyrase and topoisomerase IV are also tracked so that they can be unbound from the DNA when appropriate.

Algorithm
1. Determine regions of DNA
The superhelicity of the chromosome is tracked separately for different each double stranded region. The exact base positions and length in base pairs, l, of each region is determined based on the progress of replication and positions of breaks in the DNA. If there are no breaks in the DNA, the following DNA regions exist:

Before Replication:

- Region 1: Unreplicated DNA

During Replication:

- Region 1: Replicated DNA, Chromosome 1 (upstream of replication loops, increasing in length over time)
- Region 2: Replicated DNA, Chromosome 2 (upstream of replication loops, increasing in length over time)
- Region 3: Unreplicated DNA (downstream of replication loops, decreasing in length over time)

After Replication:

- Region 1: Replicated DNA, Chromosome 1
- Region 2: Replicated DNA, Chromosome 2

2. Calculate σ at each DNA region
The enzymes may act on the chromosome affecting the LK_{current}, and experimentally it has been shown that the enzymes would obtain an LK_{current} such that the steady state $\sigma = -0.06$[923]. For each region of DNA, the LK_{current} is stored across timesteps. The superhelical density, σ, of each region is calculated using the LK_{current}as follows:

$$LK_{\text{relaxed}} = \frac{l}{10.5}$$
$$\sigma = \frac{LK_{\text{current}} - LK_{\text{relaxed}}}{LK_{\text{relaxed}}}$$

3. Determine which topoisomerases can act in which DNA regions
Gyrase, topoisomerase I, and topoisomerase IV can only act on the DNA if the superhelical density (σ) is within a certain range. A combination of step functions and logistic functions were used to determine the activity of the enzymes for a given σ. The activity profiles can be seen in Schematic S8. An enzyme can act in a DNA region if its activity profile is greater than zero at the current σ.

4. Unbind Processive Enzymes
Topoisomerase IV and DNA gyrase are processive enzymes meaning that they may perform several strand-passing events before falling off of the DNA. Topoisomerase I is not highly processive, it acts on the DNA and immediately dissociates[922].

Topoisomerase IV is highly processive and will stay bound to the DNA as long as σ is greater than zero.

- For each region, if $\sigma \leq 0$,
  - unbind all Topoisomerase IVs

Gyrase will stay bound to the DNA for about 30-60 s. Gyrase processivity is modeled as a Poisson distribution with $\lambda = 45$ s.

- For each bound gyrase,
  - if rand $< 1/\lambda$, unbind gyrase.

5. Bind Enzymes to Each Region
Topoisomerases may bind to the DNA without acting on the DNA. Therefore, binding is only limited by the availability of free topoisomerases and space on the DNA.

- Randomize the order of binding each of the three topoisomerases, to fairly allocate space on the chromosome
- For each of the three topoisomerases i,
  - For each region j where the σ permits topoisomerase i binding,
    - Stochastically bind topoisomerases i in region j, up to:
      $$\max \begin{cases} \text{Available positions in DNA region}_j \text{ of length} > \text{footprint}_i \\ \text{Number of free topoisomerase}_i \end{cases}$$

6. Calculate new Linking Numbers and Perform Metabolite Accounting
The DNA linking number is affected by the bound topoisomerases, up to the limits of available resources and kinetic rates.

- For each DNA region j,
  - For each of the three topoisomerases i that is able to bind region j given σ,
    - Count the number of topoisomerase i bound in region j (C_{ij})
    - Use the logistic/step functions (See Schematic S8) to determine the Probability (P_{ij}) of topoisomerase i acting in region j at the current σ
    - Determine the number of strand passing events (S_{ij}) for enzyme i in region j, given the kinetic rate of topoisomerase$i$ (k_i) and the energy requirement per strand passing event for topoisomerase i (E_i):
      $$S_{ij} = min \begin{cases} round(C_{ij}k_iP_{ij}) \\ \frac{\text{available ATP}}{E_i} \\ \frac{\text{available H}_2\text{O}}{E_i} \end{cases}$$

- Calculate the new linking number (LK_{current}) for region j, given topoisomerase i's effect on the linking number (δ_i):

$$LK_{\text{current}} = LK_{\text{current}} + \delta_i S_{ij}$$

- Account for used metabolites, ATP and H_2O, and produced metabolites, ADP, Pi, and H^+
- Unbind all topoisomerase I, a non-processive enzyme

7. Determine the effects of supercoiling on transcription
Supercoiling effects the expression of the genes: *gyrB*, *gyrA*, *parC*, *parE*, and *topA*, which lie in three transcription units. The `DNA Supercoiling` process calculates the fold change in the probability of an RNA polymerase binding to the promoters of these transcription units.

- For each topoisomerase transcription unit i,
  - Find the double stranded region where transcription unit i lies and the σ of this region
  - Use the fold change profiles (See Schematic S9) to determine the fold change in probability of polymerase binding to transcription unit i

3.7 FtsZ Polymerization

Biology

Cell division in many bacterial species requires the assembly of an FtsZ ring at the cell membrane around the midplane of the cell[217]. FtsZ is a homologue of eukaryotic tubulin that assembles into long polymers. These polymers are typically localized to the center of the cell, forming a membrane-bound ring[217]. FtsZ is a GTPase, and GTP hydrolysis to GDP causes the FtsZ filaments to bend. This bending serves as one of the forces enabling cell division[611]. (For more information about cell division, see the `Cytokinesis` process class.) GTP only hydrolyzes to GDP at an active site formed at the interface of two FtsZ molecules, meaning that FtsZ polymerization is required for GTPase activity[611]. This process addresses the assembly FtsZ monomers into long polymers.

Reconstruction

This process addresses the assembly FtsZ (MG224) monomers into long polymers.

All of the parameters used in the `FtsZ Polymerization` process class are described in List S14.

List S14. Fixed parameters used in the `FtsZPolymerization` process class.

Parameter	Value	Symbol	Source
Maximum length of an FtsZ filament	9 subunits	n	[611]
Rate of forward activation of FtsZ by GTP	$1.1\,\mathrm{s}^{-1}$	k_{act1}	[200]
Rate of reverse activation of FtsZ by GTP	$0.01\,\mathrm{s}^{-1}$	k_{act2}	[200]
Rate of forward exchange of FtsZ-GTP from FtsZ-GDP	$1 \times 10^4\,\mathrm{M}^{-1}\,\mathrm{s}^{-1}$	k_{ex1}	[164]
Rate of reverse exchange of FtsZ-GTP to FtsZ-GDP	$5 \times 10^3\,\mathrm{M}^{-1}\mathrm{s}^{-1}$	k_{ex2}	[164]
Rate of forward nucleation of FtsZ-GTP to form a dimer	$4.2 \times 10^6\,\mathrm{M}^{-1}\,\mathrm{s}^{-1}$	k_{nuc1}	[164, 200]
Rate of reverse nucleation of FtsZ-GTP to dissociate a dimer	$40\,\mathrm{s}^{-1}$	k_{nuc2}	[164, 200]
Rates of forward elongation of FtsZ-GTP into polymers of length 3 to 9	$5.1 \times 10^6\,\mathrm{M}^{-1}\,\mathrm{s}^{-1}$	k_{el1}	[164]
Rates of reverse elongation of FtsZ-GTP from polymers of length 3 to 9	$2.9\,\mathrm{s}^{-1}$	k_{el2}	[164]

Computational Representation

Typically, FtsZ polymers of variable length are able to bind the cell membrane to form contractile rings. The mean filament length in *E. coli* has been measured to be 100 nm (23 FtsZ monomers)[217]. The minimum fragment length observed in *C. crescentus* is 40 nm (9 FtsZ monomers)[611]. Being such a small organism, we hypothesize that *M.genitalium* would have filaments on the lower end of what is sufficient for contractile force generation. For simplicity of calculating the `Cytokinesis` progress, we use polymers of a fixed length, 9 subunits[611].

This process class simulates polymerization of FtsZ-GTP monomers to 9mers. We use a differential equation model developed by Surovtsev *et al.* to assemble the polymers[164]. During polymerization, FtsZ-GTP can exist in any intermediate state: 1mer, 2mer, ... 8mer, 9mer. The set of differential equations models the following reactions:

Activation of FtsZ monomers (F) to FtsZ-GTP (F_T)
Deactivation of FtsZ-GTP (F_T) to FtsZ (F)
Exchange of FtsZ-GDP (F_D) to FtsZ-GTP (F_T)
Reverse exchange of FtsZ-GTP (F_T) to FtsZ-GDP (F_D)
Nucleation of two FtsZ-GTP molecules (F_T) to form a dimer (F_{T2})
Reverse nucleation of a dimer (F_{T2}) into two FtsZ-GTPs (F_T)
Polymer elongation adding an FtsZ-GTP (F_T) to an existing polymer of length 2-8 subunits (F_{T2}- F_{T8})
Reverse elongation removing an FtsZ-GTP (F_T) from an existing polymer of length 3-9 subunits (F_{T3}- F_{T9})

Integration
The count full-length activated FtsZ (9mer) filaments is used by the `Cytokinesis` process class for cell pinching.

Initial Conditions
The FtsZ polymerization differential equations are run a number of times (up to 50 iterations) before the simulation starts, to ensure starting at a steady state distribution of polymer lengths. Iterations are run until the change in the solution across consecutive iterations is less than a set tolerance (0.2% of the enzymatic counts).

Dynamic Computation
FtsZ can exist in one of multiple states: inactivated monomer, activated monomer (GTP bound), nucleated (dimer of two activated FtsZ molecules), elongated polymer of three or more GTP bound FtsZ molecules. The hydrolyzed FtsZ polymers are only considered during cell membrane pinching in the `Cytokinesis` process class. The FtsZ molecules can move to states of higher and lower polymerization at rates obtained from Chen *et al.* and Surovtsev *et al.*[164,200].

FtsZ polymerization is modeled using a set of differential equations modified from that described in Surovtsev *et al.*, involving the activation, nucleation, and elongation of FtsZ polymers[164]. The main modifications were that the equations were simplified to not include annealing and cyclization of FtsZ polymers.

The following differential equation model is evaluated at each timestep:

$$\begin{aligned}
\frac{dF}{dt} &= k_{\text{act}2}F_T - k_{\text{act}1}F \\
\frac{dF_D}{dt} &= k_{\text{ex}2}F_T\,[\text{GDP}] - k_{\text{ex}1}F_D\,[\text{GTP}] \\
\frac{dF_T}{dt} &= k_{\text{act}1}F - k_{\text{act}2}F_T + k_{\text{ex}1}F_D\,[\text{GTP}] - k_{\text{ex}2}F_T\,[\text{GDP}] - 2k_{\text{nuc}1}F_T^2 + \cdots \\
&\quad + \cdots 2k_{\text{nuc}2}F_{T2} - k_{\text{el}1}F_T\left(\sum_{i=2}^{8}F_{Ti}\right) + k_{\text{el}2}\left(\sum_{i=3}^{9}F_{Ti}\right) \\
\frac{dF_{T2}}{dt} &= k_{\text{nuc}1}F_T^2 - k_{\text{nuc}2}F_{T2} - k_{\text{el}1}F_TF_{T2} + k_{\text{el}2}F_{T3} \\
\frac{dF_{Ti}}{dt} &= k_{\text{el}1}F_TF_{Ti-1} - k_{\text{el}2}F_{Ti} - k_{\text{el}1}F_TF_{Ti} + k_{\text{el}2}F_{Ti+1},\ \text{for } i \in 3..8 \\
\frac{dF_{T9}}{dt} &= k_{\text{el}1}F_TF_{T8} - k_{\text{el}2}F_{T9}
\end{aligned}$$

Solving these equations results in a real-value distribution of monomers and filament lengths at each timestep. The model framework requires the storage of the count of each molecular entity rather than the concentration. This process class discretizes the distribution at each time step for compatibility with the rest of the simulation.

3.8 Host Interaction

Biology

M. genitalium is a common extracellular urogenital epithelial parasite. Although *M. genitalium* is estimated to colonize the urogenital tract of 3% of all women, most infections are asymptomatic, and most researchers regard *M. genitalium* as an "ideal parasite" which lives in harmony with its human host[313]. Only a small fraction of *M. genitalium* infections manifest clinically as urethritis, cervicitis, or pelvic inflammatory disease.

Reconstruction

Although *M. genitalium* is a common urogenital pathogen, the pathophysiology of *M. genitalium* is only beginning to be elucidated. Razin and colleagues have studied the *Mycoplasma* terminal organelle, and have shown that several lipoproteins (MG191, MG192, MG217, MG318, and MG386) are involved in host adhesion and activation of the host immune response[88,313]. Shimizu *et al.* and McGown *et al.* have identified 3 additional lipoproteins – MG149, MG309, and MG412 – which activate the host immune response by stimulating Toll-like (TLR) receptors 1, 2, and 6, which in turn activate the host transcriptional regulator NF-κB[194,842,844]. Additionally, Duffy *et al.* have reported that MG075 is immunoreactive[176]. Razin and colleagues have suggested that *Mycoplasma* also stimulates the host immune response by secreting hydrogen peroxide and superoxide radicals which cause membrane oxidative damage and trigger inflammation[313].

Computational Representation

Mathematical Model

Because the interaction between *M. genitalium* and its human host is not well understood, this process implements a Boolean model of the effect of the *M. genitalium* terminal organelle and accessory lipoproteins on six properties of the host human urogenital epithelium:

- Adherence – Mycoplasma adheres to its host if its terminal organelle is properly formed and all of its adhesion proteins are expressed.
- TLR 1, 2, and 6 activation – Host TLR 1 is activated if the bacterium is adherent and MG149 or MG412 is expressed. TLR 2 is activated if the bacterium is adherent and MG149, MG309, or MG412 is expressed. TLR 6 is activated if the bacterium is adherent and MG309 is expressed.
- NF-κB activation – NF-κB is activated if TLRs 2 and 1 or 6 are active.
- Inflammatory response activation – The host inflammatory response is activated if either NF-κB is active or the bacterium is adherent and at least one of the MG075, MG149, MG309, or MG412 antigens is expressed.

Integration

The `Host` state uses six Boolean variables to represent the status of the human urogenital epithelium. The `Protein Monomer` state represents the copy number of each terminal organelle and accessory lipoprotein synthesized and matured by the protein maturation pathway (see Section 2.10) and translocated into the terminal organelle (see `Terminal Organelle Assembly` process).

Initial Conditions

After the `Protein Monomer` and `Terminal Organelle Assembly` states initialize the copy number of each protein, the `Host` state is initialized to the steady-state of the `Host Interaction` Boolean model. Because the Boolean model is acyclic, the model converges in one iteration.

Dynamic Computation
At each time step, this process evaluates the six Boolean rules outlined in Algorithm S7 until convergence. Because the Boolean rules are acyclic, the model converges in one iteration.

Algorithm S7 | Host-parasite interaction simulation.

```
$adherent \leftarrow$ organelle proteins $MG191$, $MG192$, $MG217$, $MG218$, $MG312$, $MG317$, $MG318$, and $MG386$ expressed
$tlr1 \leftarrow adherent$ AND (lipoproteins $MG149$ and $MG412$ expressed)
$tlr2 \leftarrow adherent$ AND (lipoproteins $MG309$ expressed)
$tlr6 \leftarrow adherent$ AND (lipoproteins $MG149$, $MG309$, and $MG412$ expressed)
$nfkb \leftarrow tlr2$ AND ($tlr1$ OR $tlr6$)
$inflammation \leftarrow nfkb$ OR ($adherent$ AND $MG075$, $MG149$, $MG309$, or $MG412$ antigen expressed)
```

3.9 Macromolecular Complexation

Biology

Many enzymes are functional as multimeric proteins or ribonucleoproteins. Macromolecular complexes are believed to form quickly ($k_{on} \approx 10^3 - 10^6\,\mathrm{M}^{-1}\,\mathrm{s}^{-1}$), energetically favorably ($\Delta G \approx -12\,\mathrm{kcal\,mol}^{-1}$), and stably ($K_D \approx 10^{-4} - 10^{-10}\,\mathrm{M}$). This process models the formation of all macromolecular complexes except the 30S and 50S ribosomal particles, the 70S ribosome, the FtsZ ring, and the oriC DnaA complex. The `Ribosome Assembly`, `Translation`, `FtsZ Polymerization`, and `Replication Initiation` processes model the more complex and better characterized formation of these complexes.

Reconstruction

An extensive review of the primary literature and several databases (see Table S3AI) suggested that *M. genitalium* forms 201 distinct macromolecular complexes containing 269 protein and 5 RNA gene products[6,96,386,570]. Table S3N lists the reconstructed composition of each *M. genitalium* macromolecular complex.

Computational Representation

Mathematical Model
This process models the formation of macromolecular complexes as a stochastic process. Because macromolecular complexation is poorly understood, this process makes several simplifying assumptions. First, this process assumes that macromolecular complexes form spontaneously, uncoupled to other chemical reactions and without assistance from chaperones or other proteins. The contribution of chaperones to the three-dimensional folding of individual protein monomers is modeled by the `Protein Folding` process. Second, this process assumes that macromolecular complexation is fast and proceeds to completion within the 1 s time step of the simulation. Third, this process makes the simplifying assumption that each macromolecule complex forms with the same specific rate. Fourth, this process assumes that complexes form by simultaneous collision of each subunit. Together these assumptions imply that the relative formation rate, r_i of each complex, i, is described by mass-action kinetics,

$$r_i = \prod_j \left(\frac{m_j}{V}\right)^{S_{ij}}, \tag{S13}$$

where m_j is the copy number of gene product j, V is the cell volume, and S_{ij} is the stoichiometry of subunit j in complex i. This process models complex formation by (1) calculating the relative formation rate of each complex, r_i, (2) randomly forming one complex according to a multinomial distribution defined by the relative formation rates, (3) updating the copy numbers of RNA and protein subunits and complexes, and (4) repeating until insufficient protein and RNA subunits are available to form additional complexes. Importantly, this algorithm resolves the order of macromolecular complex formation before several subunits participate in multiple complexes.

Integration
The Rna and Protein Monomer states represent the copy number of each RNA and protein subunit synthesized by the RNA and protein synthesis pathways (see Section 2.12 and 2.10). The Protein Complex state represents the copy number of each complex, c_i. The Geometry state represents the cell volume.

Initial Conditions
After the cell volume and the total copy number of each RNA and protein gene product are initialized (see Geometry, Rna, and Protein Monomer states), the copy number of each macromolecular complex is initialized by iteratively evaluating the dynamic model until convergence, or more specifically until insufficient subunits are available to form additional complexes.

Dynamic Computation
Algorithm S8 outlines the implementation of the macromolecular complexation model.

Algorithm S8 | Macromolecular complexation simulation. See Mathematical Model above for mathematical notation.

```
repeat
    foreach protein complex $i$ do
        Calculate relative formation rate, $r_i \leftarrow \prod_j \left(\frac{m_j}{V}\right)^{S_{ij}}$
    Select a complex $k$ to form according to multinomialRand(1, $r_i / \sum_i r_i$)
    Increment copy number of complex $k$, $c_k \leftarrow c_k + 1$
    Decrement copy numbers of complex $k$ subunits, $m_j \leftarrow m_j - S_{k,j}$
until Insufficient subunits to form additional complexes
```

3.10 Metabolism

Biology

To grow and replicate cells transform external nutrients into cellular mass. The first step in this process is to import nutrients from the external environment and metabolize those nutrients primarily into macromolecule building blocks. Furthermore, cells must recycle and/or export byproducts. This process models the import of extracellular nutrients and their conversion into macromolecule building blocks.

M. genitalium is believed to have adapted to the rich environment provided by its host human urogenital epithelium by massive degenerative evolution from low G+C content Gram positive bacteria, eliminating non-essential genes involved in oxidative phosphorylation, ATP synthesis via the pentose phosphate pathway, and amino acid, nucleotide, lipid, and cofactor biosynthesis[57,881,883]. Several studies have also shown that *M. genitalium* has evolved metabolic enzymes with relaxed substrate specificity[444,596,798,882]. *M. genitalium* is generally cultured on SP-4 medium, a complex and undefined medium based on CMRL-1066[565].

Reconstruction

Extracellular Medium
The *in silico* medium was based on SP-4 medium with supplementary metabolites added to facilitate *in silico* growth (see Table S3BI). First, the chemical composition of SP-4 medium was estimated from the characterized composition of each medium component (see Section 2.7 and Table S3BI). Second, additional metabolites were added until the flux-balance analysis (FBA) metabolic model predicted non-zero growth. Finally, the concentrations of several metabolites were increased to support sustained cell growth.

Cellular Composition
The *in silico* chemical composition of *M. genitalium* was based on its reconstructed cellular mass composition (see Table S3AR-S3BG), and fit to match the 27 other modeled cellular processes (see Section 2.5 and 1.3).

Energy
In addition to metabolic building blocks such as nucleic and amino acids, cellular growth and maintenance requires energy. The *in silico M. genitalium* growth-associated maintenance energy (GAM) was based on the characterized energetic requirements of *E. coli*[558].

Reactions
The *M. genitalium* metabolic network illustrated in Figure S1B was reconstructed as previously described[884] with guidance from the *M. genitalium* FBA metabolic model recently reported by Suthers *et al.*[610]. Briefly, the metabolic network was reconstructed based on homology to previously modeled organisms[558] and an extensive review of the primary literature. In particular, the metabolic network was reconstructed to support the metabolic demands of all 27 other modeled cellular processes. Table S3O lists the reconstructed metabolic reactions.

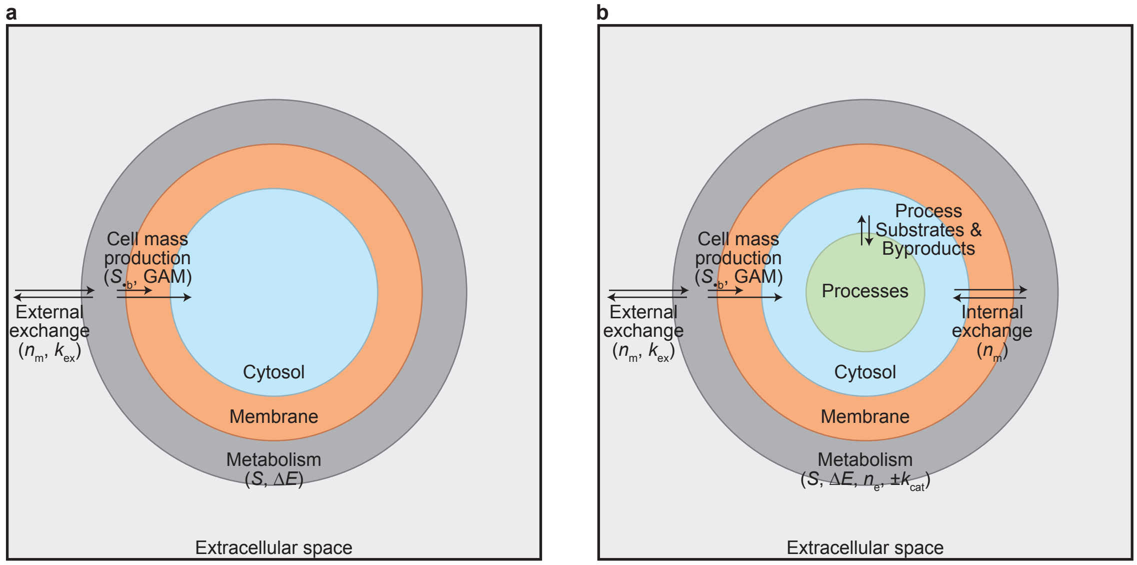
Kinetics
Reaction kinetics were curated primarily from the proteomic databases SABIO-RK[100], BRENDA[570], and BioCyc[6] (see Table S3AL). The maximum exchange rates of carbon- and non-carbon-containing metabolites were set to 12 and 20 $\text{mmol gDCW}^{-1}\,\text{h}^{-1}$, comparable to recent FBA models of *M. genitalium*[610] and *E. coli*[558] metabolism.

Computational Representation

Mathematical Model
This process models several critical cellular functions including (1) the uptake of external nutrients, (2) the synthesis of the metabolic building blocks and intermediate energy carriers, (3) lipid assembly, membrane insertion, and maturation, and (4) the recycling and export of the metabolic byproducts of other cellular processes. This process makes two key assumptions and uses FBA[686,687] to model *M. genitalium* metabolism. In particular, this process calculates the flux, v, of each metabolic reaction. First, this process assumes that the internal dynamics of the metabolome are fast compared to the 1 s time step of the *M. genitalium* simulation, or equivalently, that the metabolic network can be considered to be at steady-state on the 1 s simulation time scale. Second, this process assumes that the *M. genitalium* metabolic network maximizes cellular metabolite production given the available extracellular nutrients and metabolic enzymes. See Orth *et al.*[687] for further discussion of the assumptions and mathematical formalism of FBA.

To be compatible with the 27 other modeled cellular processes, the metabolic sub-model has four key differences from most FBA metabolic models. First, the metabolic network, S, and the cellular mass production pseudoreaction are expanded to produce all of the metabolites required by the 27 other processes, where S_{ij} is the stoichiometry of metabolite i in reaction j and S_{ib} is the stoichiometry of metabolite i in the cellular mass production pseudoreaction, b. Second, internal exchange reactions are added to recycle the metabolic byproducts of the other processes (e.g. recycle ADP and Pi to ATP), that is to exchange metabolites between the cytosol and the metabolic network. Third, the optimization objective, g, is expanded to include the construction of new cell mass as well as the recycling and export of the metabolic byproducts of the 27 other processes. g represents the relative fitness gains of intracellular metabolite recycling and novel cell mass production, that is g represents how heavily the metabolic network favors novel cell mass production over metabolite recycling. g was set assuming that the production/exchange of each molecule contributes equally to cellular fitness. Specifically g was set such that the production/exchange of each individual molecule is weighted equally. Fourth, the flux bounds, v_l are v_u, are functions of the cell dry mass, m, the copy number and maximum exchange rate of each extracellular metabolite, n_m and $k_{\pm ex}$, the thermodynamics, ΔE, of each reaction, and the copy number and maximum catalytic rate of each metabolic enzyme, n_e and $k_{\pm cat}$. Specifically, external exchange flux bounds are functions of the cell dry mass and the extracellular copy numbers and maximum exchange rates of the exchanged metabolites; internal exchange flux bounds are functions of the intracellular copy numbers of the exchanged metabolites; transport and chemical reaction flux bounds are functions of the copy numbers and kinetics of the enzymes and the reaction thermodynamics. Fifth, because the average energy used by the 27 other cellular processes is significantly less than the observed GAM, this process also represents the turnover of the otherwise unmodeled energy consumption.

Schematic S10 and Eq. S14 summarize the FBA metabolic model.


a
Cell mass production ($S_{\bullet b}$, GAM)
External exchange (n_m, k_{ex})
Cytosol
Membrane
Metabolism (S, ΔE)
Extracellular space
b
Process Substrates & Byproducts
Cell mass production ($S_{\bullet b}$, GAM)
Processes
External exchange (n_m, k_{ex})
Internal exchange (n_m)
Cytosol
Membrane
Metabolism (S, ΔE, n_e, $\pm k_{cat}$)
Extracellular space

Schematic S10. Metabolite perspective of the flux-balance analysis (FBA) metabolic model. a, Conventional FBA metabolic model. **b**, Integrated FBA metabolic model.

$$
\begin{aligned}
v^* &= \underset{v}{\operatorname{argmax}}\ g^\intercal v \text{ subject to} \\
Sv &= 0 \text{ and} \\
v_l \leq & v \leq v_u \text{ where} \\
g_i &= \begin{cases} \sum_{j \neq bm} S_{j,b} & \text{pseudoreaction } i = b \text{ represents the synthesis of biomass (metabolite } bm\text{)} \\ -1 & \text{reaction } i \text{ represents an internal exchange} \\ 0 & \text{otherwise} \end{cases} \\
v_l &= f_l\left(m, n_m, k_{-ex}, -\Delta E, n_e, k_{-cat}\right) \\
v_u &= f_u\left(m, n_m, k_{-ex}, \Delta E, n_e, k_{-cat}\right)
\end{aligned}
\tag{S14}
$$

Algorithm S9 describes the functional forms of f_l and f_u in detail.

Integration

The `Metabolism` process imports extracellular nutrients and converts them into the building blocks of macromolecular synthesis. The `Metabolite` state represents both the extracellular and cellular copy numbers of each metabolite. The rates of extracellular nutrient exchange are functions of the total cell mass represented by the `Mass` state and the extracellular metabolite copy numbers. Cytosolic and membrane metabolite copy numbers limit the rates of internal exchange/recycling. The rates of chemical and transport reactions are limited by the copy numbers and kinetics of metabolic enzymes synthesized by the protein synthesis and maturation pathway (see Section 2.10) and represented by the `Protein Monomer` and `Protein Complex` states. The `Metabolic Reaction` state records the calculated flux of each metabolic reaction. The `Simulation` object allocates metabolites produced by the `Metabolism` process to the other 27 cellular processes to support several functions including DNA, RNA, and protein synthesis and maturation. Additionally, the other 27 processes generate byproducts which the `Metabolism` process either recycles or exports from the cell.

Initial Conditions

The `Metabolite` state initializes the copy number of each metabolite. The `Protein Monomer` state initializes the total copy number of each protein monomer. The `Macromolecular Complexation` process initializes the copy number of each macromolecular complex. This process initializes the cellular growth rate and reaction fluxes to the steady-state of the metabolic network using the same FBA simulation used at each time step.

Dynamic Computation
Algorithm S9 outlines the implementation of the FBA metabolic model.

Fitting
The metabolic model was fit to match the observed *M. genitalium* mass doubling time, $\tau = 9\,\mathrm{h}$. Specifically, metabolic enzyme expression was fit using a modified version of minimization of metabolic adjustment (MOMA)[876]. Let μ_o, v_o be the FBA solution. Let $\mu^* = \ln 2/\tau$ be the target growth rate. First, find a neighboring flux distribution consistent with the target growth rate. If $\mu^* > \mu_o$ find a neighboring flux distribution within the current flux bounds v_l and v_u,

$$\begin{aligned} v^* &= \underset{v}{\operatorname{argmax}} \left| \frac{v - v_o}{v_o} \right| \text{ subject to} \\ Sv &= 0 \\ v_l \leq &v \leq v_u \\ v_b &= \mu^*. \end{aligned} \tag{S15}$$

If $\mu^* < \mu_o$ find a neighboring flux distribution by relaxing the flux bounds by μ^*/μ_o,

$$\begin{aligned} v^* &= \underset{v}{\operatorname{argmax}} \left| \frac{v - v_o}{v_o} \right| \text{ subject to} \\ Sv &= 0 \\ v_l \frac{\mu^*}{\mu_o} \leq &v \leq v_u \frac{\mu^*}{\mu_o} \\ v_b &= \mu^*. \end{aligned} \tag{S16}$$

Second, invert f_l and f_u to calculate a set of enzyme expression consistent with v^*,

$$n_m = \max\left(f_l^{-1}\left(v^*\right), f_u^{-1}\left(v^*\right)\right). \tag{S17}$$

The upper and lower bound of the expression of each enzyme consistent with the target growth rate was calculated using a similar procedure.

Algorithm S9 | Metabolism FBA simulation. See Mathematical Model section above for definition of the mathematical notation.

```
Calculate flux bounds
begin
    Initialize bounds: $v_{l,i} \leftarrow -\inf$, $v_{u,i} \leftarrow +\inf$
    foreach thermodynamically irreversible reaction $i$ do
        Constrain reverse flux to zero: $v_{l,i} \leftarrow 0$
    foreach chemically catalyzed reaction $i$ do
        $j \leftarrow$ index of enzyme which catalyzes reaction $i$
        if $k_{-,i}$ is known then
            bound flux by enzyme kinetics and expression: $v_{l,i} \leftarrow \max(v_{l_i}, k_{-cat,i} n_{e,j})$
        else bound flux by enzyme expression: $v_{l,i} \leftarrow v_{l,i} \cdot (n_{e,j} > 0)$
        if $k_{+,i}$ is known then
            bound flux by enzyme kinetics and expression: $v_{u,i} \leftarrow \min(v_{u_i}, k_{+cat,i} n_{e,j})$
        else bound flux by enzyme expression: $v_{u,i} \leftarrow v_{u,i} \cdot (n_{e,j} > 0)$
    foreach chemical reaction $i$ do
        foreach protein substrate $j$ of reaction $i$ do
            if protein substrate $j$ is not expressed then constrain flux to zero: $v_{l,i} \leftarrow 0$, $v_{u,i} \leftarrow 0$
    foreach internal exchange reaction $i$ do
        $j \leftarrow$ index of metabolite exchanged by reaction $i$
        Bound internal metabolite exchange by copy number: $v_{l,i} \leftarrow \max(v_{l_i}, -n_{m,j})$
    foreach external exchange reaction $i$ do
        $j \leftarrow$ index of metabolite exchanged by reaction $i$
        Bound external metabolite exchange by copy number and maximum exchange rate: $v_{l,i} \leftarrow \max(v_{l_i}, mk_{-ex,i})$,
        $v_{u,i} \leftarrow \min(v_{u_i}, mk_{+ex,i}, n_{m,j})$

Calculate the growth rate and flux of each reaction
begin
    $v^* \leftarrow \underset{v}{\operatorname{argmax}}\ \mu = v_b$ subject to $Sv = 0$ and $v_l \leq v \leq v_u$
    $\mu^* \leftarrow v_b^*$

Update the copy number of each metabolite species
begin
    foreach extracellular metabolite $i$ do
        $j \leftarrow$ index of reaction which exchanges metabolite $i$
        $n_{m,i} \leftarrow n_{m,i} + v_j^*$
    foreach intracellular metabolite $i$ do
        $j \leftarrow$ index of reaction which exchanges/recycle metabolite $i$
        $n_{m,i} \leftarrow n_{m,i} + v_j^*$
    foreach metabolic objective component $i$ do
        $n_{m,i} \leftarrow n_{m,i} - S_{i,b}\mu^*$
    Turnover extra energy produced beyond the demands of the 27 other modeled cellular processes,
    $\Delta GAM = GAM -$ used energy
    $n_{m,ATP} \leftarrow n_{m,ATP} - \Delta GAM\mu^*$
    $n_{m,ADP} \leftarrow n_{m,ADP} + \Delta GAM\mu^*$
    $n_{m,Pi} \leftarrow n_{m,Pi} + \Delta GAM\mu^*$
    $n_{m,H_2O} \leftarrow n_{m,H_20} - \Delta GAM\mu^*$
    $n_{m,H^+} \leftarrow n_{m,H^+} + \Delta GAM\mu^*$
```

3.11 Protein Activation

Biology

The activity of mature protein monomers and complexes is not fixed, but rather can be modulated by small molecules, DNA, RNA, and other proteins, as well as by temperature and pH. Furthermore, cells often purposefully modulate protein activity to respond to external signals and maintain homeostasis.

Reconstruction

Protein chemical regulation was reconstructed based on extensive review of the primary literature and several databases, with particular emphasis on antibiotics (see Table S3Q and S3AQ). The reconstructed protein chemical regulation network contains 16 metabolites, three Boolean-valued pseudometabolites or stimuli, and temperature which regulate 10 proteins including four transcription factors, topoisomerases II and IV, the 30S and 50S ribosomal particles. The 10 putative chemically regulated proteins are critical for transcription, supercoiling, and translation. The effects of physical properties such as temperature and pH on protein activity were curated from BRENDA[570], BioCyc[6], and UniProt[96] (see Table S3AO and S3AP). The prosthetic groups and coenzymes required for maturation and catalysis were also curated (see Table S3AM).

Computational Representation

Mathematical Model

Because the exact effects of small molecules, temperature, and pH on the functional activity of proteins are not well characterized, this process implements a Boolean model of their effects on the functional state – enzymatically competent or incompetent – of mature proteins. Less well characterized effects of other physical properties such as pH are not modeled. The functional state, I_i, of every copy of protein species i is governed by a single independent Boolean function, f_i (see Table S3Q),

$$I_i = f_i\left(\frac{\vec{m}}{V}, \vec{s}, T\right), \tag{S18}$$

where m_j is the concentration of metabolite j, s_j is the value of stimulus j, and T is the temperature. Proteins with no known chemical regulation were assumed to be constitutively competent.

The `Protein Folding` process separately models the role of chaperones in protein folding and accounts for the small molecule prosthetic groups required for protein folding. Several processes including `Metabolism` model the coenzymes required for catalytic activity.

Integration

The `Stimulus` state represents the value of each pseudometabolite and temperature. The `Metabolite` state represents the copy number of each metabolite. The `Geometry` state represents the cell volume. The `Protein Monomer` and `Protein Complex` states represent the competent and incompetent copy numbers of each protein species. Proteins regulated by this process are synthesized and matured by several processes (see Section 2.10).

Initial Conditions

After the temperature, cell volume, pseudometabolite values, and metabolite and total protein copy numbers are initialized, the competency state of each of the regulated proteins is initialized to the steady state of the regulatory network. Because the regulatory network is acyclic, the network converges in one iteration.

Dynamic Computation

Algorithm S10 outlines the implementation of the protein activation model.

Algorithm S10 | Protein activation simulation.

```
Input: $p_i^c$ competent copy number of protein $i$
Input: $p_i^i$ incompetent copy number of protein $i$
foreach regulated protein $i$ do
    if regulatory rule $f_i(\vec{m}/V, \vec{s}, T)$ then
        Mark all copies of protein $i$ competent
        $p_i^c = p_i^c + p_i^i$
        $p_i^i = 0$
    else
        Mark all copies of protein $i$ incompetent
        $p_i^i = p_i^i + p_i^c$
        $p_i^c = 0$
```

3.12 Protein Decay

Biology

Protein degradation serves two critically important functions. First, protein degradation eliminates abnormal and potentially toxic proteins including prematurely aborted polypeptides tagged with SsrA degradation signals, and salvages their valuable metabolic resources to support protein synthesis. Second, degradation controls protein expression, enabling cells to direct valuable amino acids away from ineffective or even harmful proteins toward productive proteins, regulate protein activity, and respond to external signals. Tobias *et al.* have shown that the degradation rates of *E. coli* proteins are correlated with their N-terminal residue, suggesting that cells target proteins for degradation by manipulating their N-termini[586]. This relationship is known as the N-end rule.

Protein refolding is an efficient mechanism for eliminating abnormally folded proteins, requiring less energy than protein degradation and avoiding the large cost of resynthesis. The *M. genitalium* chromosome contains one protein chaperone – the cytosol-localized and ATP-dependent protease ClpB (MG355) – dedicated to protein refolding[14,182]. The kinetics and energetics of protein refolding are not well characterized.

This process models the degradation of protein monomers, macromolecular complexes, cleaved signal sequences, and prematurely aborted polypeptides as well as the misfolding and refolding of protein monomers and complexes.

Reconstruction

Protein Half-Lives
The half-life of each protein monomer was predicted using the N-end rule reported by Tobias *et al.*[586]. The half-life of each macromolecular complex was set equal to the weighted mean half-life of its RNA and protein subunits. The half-lives of the damaged and misfolded configurations of each protein species, cleaved signal sequences, and prematurely aborted polypeptides were set to zero to reflect their rapid degradation. Section 2.12 and Table S3Y describe how RNA half-lives were reconstructed.

Proteolytic Machinery
The *M. genitalium* genome contains homologs of proteases FtsH (MG457) and La (MG239) and six peptidases: aminopeptidase (MG324), cytosol aminopeptidase (MG391), glycoprotease (MG208), metalloendopeptidase (MG046), oligoendopeptidase F (MG183), and proline iminopeptidase (MG020)[182,634]. The membrane-localized protease FtsH is believed to cleave prematurely aborted polypeptides tagged with an N-terminal SsrA tag into approximately 15 amino long fragments by an ATP-dependent mechanism[30,634]. Bruckner *et al.* reported the kinetics and energetics of FtsH proteolysis[31]. The cytosol-localized protease La is believed to processively cleave most other proteins into fragments 10-20 amino acids in length by an ATP-dependent mechanism[29]. Lee and Suzuki reported the kinetics and energetics of La proteolysis[29]. Pep-

tidases are believed to hydrolytically degrade the polypeptide fragments produced by peptidases FtsH and La. The specific function of each of the six *M. genitalium* peptidases is unknown.

Computational Representation

Mathematical Model

Under conditions of excess proteases, peptidases, and ATP, this process models the configuration of each protein as a five-state system – aborted, nascent, mature, misfolded, and damaged – and models the degradation of protein i as Poisson process with rate parameter $k_{d,i} = \ln 2/\tau_{1/2,i}$ where $\tau_{1/2,i} = 20\,\text{h}$ for the nascent and mature states and zero for the aborted, misfolded, and damaged states. Under conditions of limited proteases, peptidases, or ATP the model stochastically degrades proteins as a function of the copy number and catalytic rate of each enzyme and the number of cytosolic ATP molecules. This models assumes (1) cytosol-localized nascent, mature, misfolded, and damaged proteins are degraded by protease La and six peptideases, (2) membrane and extracellular-localized proteins are not vulnerable to degradation by the cytosol localized protease La, (3) aborted polypeptides are degraded by protease FtsH and six peptidases, (4) protein degradation is kinetically fast and therefore this model doesn't represent intermediate degradation states, and (5) aborted, misfolded, and damaged proteins are immediately degraded if proteases are expressed and energy is available. Algorithm S11 outlines the implementation of the protein degradation model.

This process models protein unfolding as a Poisson process with rate parameter k_u set to the nominal rate $10^{-6}\,\text{s}^{-1}$ molecule^{-1}. Cytosol-localized protein refolding is modeled as a single chemical event catalyzed by the cytosol-localized chaperone ClpB and driven by the hydrolysis of a single ATP molecule. Because protein refolding is not well characterized, this process assumes that if ClpB is expressed and excess ATP is available, protein refolding proceeds to completion within the 1 s simulation time step. If ClpB is expressed, but ATP is not present in excess, the model stochastically refolds proteins equal to the intracellular copy number of ATP.

This model makes several simplifying assumptions. First, the model assumes that all protein species misfold and refold at the same rate. Second, the model assumes that the kinetics of protein misfolding and refolding are fast and therefore intermediate misfolded states can be ignored.

Integration

The `Protein Monomer` and `Protein Complex` states represent the copy number of each protein monomer and macromolecular complex including the signal sequence, misfolded, and damaged copy numbers of each protein species and the copy number of each proteolytic enzyme in each of five compartments: cytosol, membrane, terminal organelle cytosol, terminal organelle membrane, and extracellular space. The `Polypeptide` state represents the amino acid sequence of each prematurely aborted polypeptide. The `Chromosome`, `FtsZ Ring`, `Ribosome`, and `RNA Polymerase` states represents the detailed configurations of DNA-bound proteins, the FtsZ septal ring, RNA polymerases, and ribosomes. These detailed configurations are updated when DNA-bound proteins, FtsZ, RNA polymerase, or 70S ribosomes are degraded to reflect decreased enzyme copy numbers. The `Transcript` and `Polypeptide` states are also updated upon RNA polymerase or 70S ribosome degradation to reflect transcript and polypeptide abortion. The `Rna` state represents the damaged copy number of each RNA species. See below for further discussion. The `Metabolite` state represents the copy number of each intracellular metabolite.

This process marks RNA subunits of degraded macromolecules as "damaged" for immediate degradation by the `RNA Decay` process.

Initial Conditions

The misfolded and damaged configurations of each RNA and protein species are initialized with zero copy number because the typical occupancies these configurations are small compared to that of the mature configuration. Similarly, the cell is initialized with no prematurely aborted polypeptides. The `Rna`, `Protein Monomer`, and `Protein Complex` states initialize the total copy numbers of each RNA and protein species.

Dynamic Computation

Algorithm S11 outlines the implementation of the protein decay model.

Algorithm S11 | Protein decay simulation.

```
Input: $k_u = 10^{-6}\,$s protein unfolding rate
Input: $k^p_{d,i} = \ln 2/\tau_{1/2,i}$ degradation rate of protein monomer species $i$
Input: $k^c_{d,i} = \ln 2/\tau_{1/2,i}$ degradation rate of macromolecular complex species $i$
Input: $k_{\text{Lon}}$, $k_{\text{FtsH}}$ protease kinetic rates in amino acids per second
Input: $m_i$ copy number of metabolite $i$
Input: $r^d_i$ damaged copy number of RNA species $i$
Input: $p^a_i$, $p^u_i$, $p^d_i$ mature, unfolded, and damaged copy numbers of protein monomer species $i$
Input: $c^a_i$, $c^u_i$, $c^d_i$ mature, unfolded, and damaged copy numbers of macromolecular complex species $i$
Input: $p \leftarrow \{p^a_i, p^u_i, p^d_i\}$ copy number of all forms of protein monomer species $i$
Input: $c \leftarrow \{c^a_i, c^u_i, c^d_i\}$ copy number of all forms of macromolecular complex species $i$
Input: $e_{\text{Lon}}$, $e_{\text{FtsH}}$ copy numbers of proteases La and FtsH
Input: $s_i$ sequence of prematurely aborted polypeptide $i$
Input: $l^p_i$ length of protein monomer species $i$
Input: $l^s_i$ length of prematurely aborted polypeptide $i$
Input: $P^m_{ij}$ reaction stoichiometry of metabolite $i$ in the degradation of protein monomer $j$ including ATP hydrolysis
Input: $C^m_{ij}$ stoichiometry of metabolite species $i$ in macromolecular complex $j$
Input: $C^r_{ij}$ subunit stoichiometry of mature RNA species $i$ in macromolecular complex $j$
Input: $C^p_{ij}$ subunit stoichiometry of mature protein monomer $i$ in macromolecular complex $j$
Input: $S^m_{ij}$ reaction stoichiometry of metabolite $i$ in the degradation of polypeptide $j$ including ATP hydrolysis
Input: $\Delta t = 1\,$s is the simulation time step

Misfold proteins (see Algorithm S13)
Refold proteins (see Algorithm S12)
Degrade macromolecular complexes (see Algorithm S14)
Degrade prematurely aborted polypeptides (see Algorithm S15)
Degrade protein monomers (see Algorithm S16)
```

Fitting

The expression of the proteases and peptidases was fit to provide sufficient enzymes to quickly degrade damaged proteins, or more specifically to prevent sustained accumulation of damaged proteins. See Section 1.3 for further discussion.

Algorithm S12 | Protein refolding simulation. See Algorithm S11 for mathematical notation.

```
if ClpB chaperone is expressed ($e_{ClpB} > 0$) then
    repeat
        Select protein species $i \sim$ multinomialRand(1, $\{p^u, c^u\}/\left(\sum_j p^u_j + \sum_j c^u_j\right)$)
        Update protein copy numbers
        if species $i$ is a protein monomer then $p^u_i \leftarrow p^u_i - 1$, $p^a_i \leftarrow p^a_i + 1$
        else $c^u_i \leftarrow c^u_i - 1$, $c^a_i \leftarrow c^a_i + 1$
        Hydrolyze ATP
        $m_{ATP} \leftarrow m_{ATP} - 1$
        $m_{ADP} \leftarrow m_{ADP} + 1$
        $m_{Pi} \leftarrow m_{Pi} + 1$
        $m_{H_2O} \leftarrow m_{H_2O} - 1$
        $m_{H^+} \leftarrow m_{H^+} + 1$
    until $(p^u_i = 0 \ \forall \ i and c^u_i = 0 \ \forall \ i)$ or $m_{ATP} = 0$
```

Algorithm S13 | Protein unfolding simulation. See Algorithm S11 for mathematical notation.

```
foreach protein monomer $i$ do
    Calculate protein unfolding extent: $\Delta p^u_i \leftarrow$poissonRand($k_u p^a_i$)
    Update protein copy numbers: $p^a_i \leftarrow p^a_i - \Delta p^u_i$, $p^u_i \leftarrow p^u_i + \Delta p^u_i$
foreach macromolecular complex $i$ do
    Calculate protein unfolding extent: $\Delta c^u_i \leftarrow$poissonRand($k_u c^a_i$)
    Update protein copy numbers: $c^a_i \leftarrow c^a_i - \Delta c^u_i$, $c^u_i \leftarrow c^u_i + \Delta c^u_i$
```

Algorithm S14 | Macromolecular complex degradation simulation. See Algorithm S11 for mathematical notation.

```
Calculate rates of protein degradation
foreach macromolecular complex $i$ do
    $\Delta c_i \leftarrow$ poissonRand($k^c_{d,i} c_i$)
Degrade specific complexes
repeat
    Select protein species $i \sim$ multinomialRand($\Delta c_i / \sum_j \Delta c_j$)
    if sufficient metabolic resources available ($m_j \geq -C^m_{ji} \ \forall \ j$) then
        Breakdown complex into metabolites and damaged RNA and protein subunits
        $\Delta c_i \leftarrow \Delta c_i - 1$
        $c_i \leftarrow c_i - 1$
        $p^d \leftarrow p^d_i + C^p_{\bullet i}$
        $r^d \leftarrow r^d_i + C^r_{\bullet i}$
        $m \leftarrow m C^m_{\bullet i}$
    else
        $\Delta c_i \leftarrow 0$
until $\Delta c_i = 0 \ \forall \ i$
```

Algorithm S15 | Aborted polypeptide degradation simulation. See Algorithm S11 for mathematical notation.

```
Let $z \leftarrow$poissonRand($e_{FtsH}/k_{FtsH} \Delta t$) be the FtsH enzymatic capacity
while all peptidases expressed ($e_i \ \forall$ peptidases$i$) do
    Select an aborted polypeptides $i \sim$ multinomialRand(1, $1/n_{pep}$)
    if insufficient resources to degrade polypeptide ($\exists j s.t - S^m_{ji} > m_j$ and $z \geq$length($l^s_i$) $\geq 1$) then
        break
    Degrade polypeptide: $s_i \leftarrow \emptyset$, $m \leftarrow m + S^m_{\bullet i}$, $z \leftarrow z -$length($l^s_i$)
```

Algorithm S16 | Protein monomer degradation simulation. See Algorithm S11 for mathematical notation.

```
if all peptidases expressed ($e_i\ \forall$ peptidasesi) then
    Let $z \leftarrow$poissonRand($e_{Lon}/k_{Lon}\Delta t$) be the Lon enzymatic capacity
    Calculate rates of protein degradation
    foreach protein monomer $i$ do
        $\Delta p_i \leftarrow$ poissonRand($k^p_{d,i} p_i$)
    Degrade specific monomers
    repeat
        Select protein species $i \sim$ multinomialRand($\Delta p_i / \sum_j \Delta p_j$)
        if sufficient metabolic resources available ($m_j \geq -P^m_{ji}\ \forall\ j$ and $z \geq l^p_i \geq 1$) then
            Breakdown protein into metabolites
            $\Delta p_i \leftarrow \Delta p_i - 1$
            $p_i \leftarrow p_i - 1$
            $m \leftarrow mP^m_{\bullet i}$
            $z \leftarrow z - l^p_i$
        else
            $\Delta p_i \leftarrow 0$
    until $\Delta p_i = 0\ \forall\ i$
```

3.13 Protein Folding

Biology

Proteins are synthesized as long, catalytically inactive linear chains of amino acids (see `Translation`). Subsequently, proteins fold into energetically favorable, compact, and enzymatically competent three-dimensional structures. While some protein species are believed to fold spontaneously, other proteins are believed to require helper chaperone proteins to properly fold[17,718]. In addition, some protein species require metal ions and other small molecule prosthetic groups to fold. This process models chaperone-mediated protein folding.

M. genitalium is believed to employ three chaperones to assist protein folding. First, trigger factor (Tig, MG238) co-translationally binds all nascent polypeptides at the ribosome exit site (L23) and assists in early protein folding[5,9,14,17,388,644,718].

Second, chaperones GroEL (MG392) and DnaK (MG305) and their co-chaperones are believed to assist in late folding. GroEL and DnaK are believed to fold 10-15% and 5-18% of all proteins, respectively. GroEL and its co-chaperone GroES (MG393) are believed to help fold intermediate sized proteins (20-60 kDa)[14,644]. GroEL is believed to bind each protein for 30 s to 10 min, and to couple folding to ATP hydrolysis[14,644].

DnaK and its co-chaperones DnaJ (MG019) and GrpE (MG201) are believed to help fold large proteins (> 30 kDa)[14,644]. DnaK is a monomeric protein which transiently (< 2 min) binds to the backbones of short, linear, unfolded peptide segments. GrpE is believed to couple peptide release to ATP hydrolysis. DnaJ is believed to regulate the activity of DnaK, as well bind the side chains of hydrophobic and aromatic residues and directly assist protein folding.

Finally, FtsH and the lipids phosphatidyl ethanolamine and phosphatidyl glycerol have been suggested to assist membrane protein folding[14]. However, the role of FtsH as a molecular chaperone is not well established, and FtsH has been associated with several additional functions. Consequently, we chose not to model FtsH as a chaperone. The contributions of phosphatidyl ethanolamine and phosphatidyl glycerol to protein folding are also poorly understood, and are not modeled.

Reconstruction

M. genitalium GroEL substrates were reconstructed based on two proteome-scale studies of *E. coli* and *B. subtilis* (see Table S3AG)[389,391]. *M. genitalium* DnaK substrates were reconstructed based on a proteome-scale study of *E. coli* (see Table S3AG)[388]. The prosthetic group requirements for protein folding were reconstructed based on an extensive review of the literature and several databases (see Table S3AM). Table S3M lists the chaperones required to fold each protein monomer and the prosthetic group stoichiometry of each protein monomer.

Computational Representation

Mathematical Model

This process implements a model of the chaperone-mediated folding of processed, translocated polypeptides. Because the kinetics and energetics of protein folding are not well understood, this process represents the three-dimensional configuration of each protein as a two-state – folded, unfolded – Boolean variable. Furthermore, this process makes the simplifying assumption that the folding rate, r_i, of protein species i is a Boolean function of the copy numbers of metabolites and chaperones,

$$r_i = \min\left(\overbrace{p_i^u}^{\text{protein}},\ \overbrace{\left\lfloor \min_j \frac{m_j}{M_{ji}} \right\rfloor}^{\text{metabolites}},\ \overbrace{\min_j \frac{e_j}{C_{ji}}}^{\text{enzymes}}\right), \tag{S19}$$

where p_i^u is the unfolded copy number of protein species i, m_i is the copy number of metabolite i, e_i is the copy number of chaperone i, M_{ji} is the stoichiometry of prosthetic group j in protein i, and C_{ji} is true if protein species i requires chaperone j to fold. p_i^f is the folded copy number of protein species i.

Integration

The `Metabolite` state represents the copy number of each metabolite species. The `Protein Monomer` and `Protein Complex` states represent the copy numbers of unfolded and folded proteins. Protein folding is one of the last steps in protein maturation following protein processing and translocation (see `Protein Processing I` and `Protein Translocation` processes) and preceding protein modification (see `Protein Modification` process). See Section 2.10 for further discussion.

Initial Conditions

The `Protein Monomer` and `Protein Complex` states initialize the total copy number of each protein species, and set all protein monomers and complexes to their mature – folded and modified – configuration.

Dynamic Computation

Algorithm S17 outlines the implementation of the protein folding model.

Algorithm S17 | Protein folding simulation.

```
repeat
    foreach protein species $i$ do
        Calculate the relative Boolean-valued folding rate, $r_i$, of each protein species
        $r_i \leftarrow \min\{p_i^u, m_j/M_{ij}, c_k \geq C_{ik}\}$
    Select a protein species $i$ to fold according to multinomialRand(1, $\vec{r}$)
    Increment the folded copy number of protein species $i$, $p_i^f \leftarrow p_i^f + 1$
    Decrement the unfolded copy number of protein species $i$, $p_i^u \leftarrow p_i^u - 1$
    Decrement the copy numbers of the prosthetic groups of protein species $i$, $m_j \leftarrow m_j - M_{ij}$
until no additional proteins can fold ($\vec{r} = 0$)
```

Dynamic Computation

Chaperone expression was fit to provide sufficient enzymes to quickly fold all newly synthesized proteins, or more precisely, to prevent accumulation of unfolded proteins. See Section 1.3 for further discussion.

3.14 Protein Modification

Biology

Post-translational protein modification serves several important functions[281,890]. First, post-translation modification can increase the structural and chemical diversity of the proteome by stabilizing alternative conformations and providing catalytic cofactors. Second, post-translational modification, and in particular phosphorylation, provides a mechanism to regulate protein activity. Third, post-translational modification can be used to regulate protein expression through proteasome recruitment. This process models protein covalent modification including phosphorylation, lipoyl transfer, and α-glutamate ligation.

Reconstruction

The *M. genitalium* protein modification complement was reconstructed in three steps. First, protein modifications that have been observed in *M. genitalium*[94], *M. pneumoniae*[94,277,282,283,672], or other related bacteria[96,105,276,280,673] or which have been computationally predicted[268,278,575] were curated. Second, curated protein modifications were mapped to *M. genitalium* homologs using sequence alignment. Third, based on the genome annotation[182], *M. genitalium* was determined to have three protein modification pathways: (1) phosphorylation catalyzed by serine/threonine kinase PrkC (MG109), (2) lipoyl transfer to lysine catalyzed by LplA (MG270), and (3) C-terminal α-glutamate ligation catalyzed by RimK (MG012). Fourth, modifications were rejected which do not belong to one of these three pathways, or for which a specific locus was not reported. Table S3AH outlines the reconstruction process and lists the reconstructed protein modifications. The reconstructed protein modification network contains one kinase that modifies 16 proteins, one lipoyl transferase that modifies the E2 subunit of pyruvate dehydrogenase (PdhC, MG272), providing an important organosulfur cofactor which contains a catalytic disulfide bond, and one α-glutamate ligase that modifies 50S ribosomal protein L6 (RplF, MG166). The stoichiometry and kinetics of all three modeled protein modification reactions were based on a review of the primary literature[110,288–290,294] (see Table S3O). Catalytic disulfide bonds were separately reconstructed and are modeled by several chemical reactions in the `Metabolism` process (see Table S3O).

Computational Representation

Mathematical Model

This process models protein covalent modification, the fifth step of post-translational processing. In particular, this process models protein phosphorylation, lipoyl transfer, and α-glutamate ligation. Because the mechanisms of protein modification are not well characterized on the genomic scale, this process make several simplifying assumptions. First, this process assumes that each protein is fully modified in a single time step, collapses the modification of each protein into a single reaction, and only represents unmodified and fully modified proteins. Intermediate protein configurations are not represented. Second, this process assumes that the mean arrival rate, v_i of modification events of each protein species i is independently limited by (1) the copy number of unmodified protein, p_i^u, (2) the copy numbers of intracellular metabolites, m_j, and (3) the copy numbers of the protein modification enzymes, e_j. Based on these assumptions, the functional form of v_i is given by

$$v_i = \min\left(\overbrace{p_i^u}^{\text{Protein}},\ \overbrace{\left\lfloor \min_j \frac{m_j}{M_{ji}^s} \right\rfloor}^{\text{metabolites}},\ \overbrace{\texttt{poissonRand}\left(\min_j \frac{e_j}{K_{ji}}\Delta t\right)}^{\text{enzymes}}\right), \tag{S20}$$

where M_{ji} is the stoichiometry of metabolite j in the modification of protein species i, $M^s = \max(0, -M)$ is the negative part of M, K_{ji} is the experimentally observed catalytic rate of enzyme j in the modification of protein species i, and $\Delta t = 1\,\text{s}$ is the simulation time step.

This process implements a stochastic model of the arrival of protein monomer and macromolecular complex modification events with relative rates v_i. Until protein, metabolic, and/or enzymatic resources are exhausted, the model iteratively (1) computes the arrival rate, v_i, of each modification event, (2) selects a single modification event to execute according to a multinomial distribution parameterized by v_i, and (3) executes the selected modification reaction, updating the copy numbers of proteins and metabolites and decrementing the available enzymatic capacity. Algorithm S18 outlines the implementation of the protein modification model.

Integration

The `Protein Monomer` and `Protein Complex` states represent the copy number of each protein modification enzyme. The `Protein Monomer` state also represents the unmodified and modified copy numbers of each protein monomers species. The `Metabolite` state represents the copy number of each intracellular metabolite.

Protein are synthesized and matured in six steps (see Section 2.10). This process models the fifth step in protein synthesis following protein folding (see `Protein Folding` process) and preceding macromolecule assembly (see `Macromolecular Complexation`, `Ribosome Assembly`, `Protein Folding`, and `Protein Modification` processes).

Initial Conditions

Section 1.4 outlines the cell state initialization algorithm. Briefly, after the `Mass` state initializes the total cell mass, the `Protein Monomer` state initializes the total copy number of each protein monomer species and initializes all monomers to their mature – processed, folded, modified, and localized – configuration. Second, the `Macromolecular Complexation` and `Ribosome Assembly` processes initialize the copy number of each ribonucleoprotein complex, and set all initialized complexes to their mature – folded and modified – configuration. Finally, several processes including `Transcription` and `Translation` initialize the detailed configurations of RNA polymerases, 70S ribosomes, DNA-binding proteins, and FtsZ.

The unmodified configurations of each protein monomer species is initialized with zero copy number because the typical occupancy this configuration is small compared to that of the modified/mature configuration.

Dynamic Computation

Algorithm S18 outlines the implementation of the protein modification model.

Algorithm S18 | Protein modification simulation. See Mathematical Model section above for definition of the mathematical notation.

```
Input: $p_i^m$ copy number of modified protein species $i$
Let $k_i \leftarrow e_i \Delta t$ be the capacity of enzyme $i$ for protein modification
repeat
    Calculate modification rates
    foreach protein species $i$ do
        Calculate $v_i$ according to Eq. S20
    Select protein species $i \sim$ multinomialRand(1, $v_i / \sum_j v_j$)
    Update protein copy numbers: $p_i^u \leftarrow p_i^u - 1$, $p_i^m \leftarrow p_i^m + 1$
    Update metabolites: $m \leftarrow m - M_{\bullet i}$
    Update enzyme catalytic capacity: $k \leftarrow k - K_{\bullet i}$
until no further modification possible ($v_i = 0 \; \forall \; i$)
```

Fitting

The expression of the protein modification enzymes was fit to provide sufficient enzymes to quickly modify newly synthesized proteins, or more specifically to prevent sustained accumulation of unmodified proteins. See Section 1.3 for further discussion.

3.15 Protein Processing I

Biology

Bacterial translation is initiated by the formation of the 70S ribosome and the recruitment of the initiator tRNA$^{\text{fMet}}$ to the ribosomal P site. This process models N-terminal formylmethionine deformylation and N-terminal methionine cleavage, the first steps in post-translational processing.

The exact function of the initiator tRNA$^{\text{fMet}}$ is unknown. Several authors suggest that bacteria employ a separate tRNA for translation initiation in order to control the rate of protein synthesis through the expression of initiator tRNA[887,889]. Others believe that cells use a separate initiator tRNA to differentiate between initiation factor 2-dependent recruitment of the first tRNA to the ribosomal P site and EF-Tu-dependent recruitment of subsequent tRNA to the A site[887,888]. The unique formyl group of tRNA$^{\text{fMet}}$ has been suggested to enable initiation factor 2 to discriminate between initiator and other tRNA[887]. The role of formylmethionine as the starting amino acid is also not well understood[886,888]. Some authors believe that bacteria employ methionine as the first amino acid because it is the most expensive to synthesize and therefore couples translation to general cell health[886]. Furthermore, neither the formyl group of tRNA$^{\text{fMet}}$ nor methionine is essential for translation initiation[886,888]. Several studies have shown that other amino acids are able to support translation initiation[886,888].

Following translation, bacteria deformylate the N-terminal methionine of most proteins and cleave the N-terminal methionine of approximately 7% of proteins. The function of peptide deformylation and methionine cleavage is not well understood. Some authors believe that bacteria employ these reactions to recycle formylmethionine which is a metabolically expensive amino acid[886]. Other authors, citing the N-end rule[586], believe that bacteria cleave the N-terminal methionine of specific proteins in order to regulate protein half-lives by exposing the second-most N-terminal amino acid[886].

Reconstruction

M. genitalium peptide deformylase (MG106) was assumed to deformylate all protein monomers. The protein substrates of *M. genitalium* methionine aminopeptidase (MG172) where reconstructed based on specific N-terminal methionine cleavages of *M. genitalium* homologs observed in *Shewanella oneidensis* MR-1[280]. The kinetics of peptide deformylation and N-terminal methionine cleavage were reconstructed from the primary literature[21,26]. Table S3O lists the stoichiometry and kinetics of the deformylation and cleavage reactions.

Computational Representation

Mathematical Model

This process models N-terminal formylmethionine deformylation and N-terminal methionine cleavage, the first step of post-translational processing. Because the mechanisms of early protein processing are not well characterized on the genomic scale, this process make several simplifying assumptions. First, this process assumes that each protein is both deformylated and cleaved in a single time step. Consequently, this process collapses the processing of each protein into a single reaction, and only represents nascent and fully processed proteins. Intermediate protein configurations are not represented. Second, this process assumes that the mean arrival rate, v_i of processing events of each protein species i is independently limited by (1) the copy number of unprocessed protein, p_i^u, (2) the copy numbers of intracellular metabolites, m_j, and (3) the copy numbers of processing enzymes, e_j. Based on these assumptions, the functional form of v_i is given by

$$v_i = \min\left(\overbrace{p_i^u}^{\text{Protein}}, \overbrace{\left\lfloor \min_j \frac{m_j}{M_{ji}^s} \right\rfloor}^{\text{metabolites}}, \overbrace{\texttt{poissonRand}\left(\min_j \frac{e_j}{K_{ji}} \Delta t\right)}^{\text{enzymes}}\right), \tag{S21}$$

where M_{ji} is the stoichiometry of metabolite j in the processing of protein species i, $M^s = \max(0, -M)$ is the negative part of M, K_{ji} is the experimentally observed catalytic rate of enzyme j in the processing of protein species i, and $\Delta t = 1\,\text{s}$ is the simulation time step.

The `Protein Processing I` process implements a stochastic model of the arrival of early protein monomer processing events with relative rates v_i. Until protein, metabolic, and/or enzymatic resources are exhausted, the model iteratively (1) computes the arrival rate, v_i, of each processing event, (2) selects a single processing event to execute according to a multinomial distribution parameterized by v_i, and (3) executes the selected processing reaction, updating the copy numbers of protein monomers and metabolites and decrementing the available enzymatic capacity. Algorithm S19 outlines the implementation of the early protein processing model.

Integration

The `Protein Monomer` and `Protein Complex` states represent the copy number of each protein processing enzyme. The `Protein Monomer` state also represents the nascent and processed copy numbers of each protein monomers species. The `Metabolite` state represents the copy number of each intracellular metabolite. Proteins are synthesized and matured in six steps (see Section 2.10). This process models the first step in post-translational processing following translation (see `Translation` process) and preceding membrane and extracellular protein translocation and cytosolic protein folding (see `Protein Translocation` and `Protein Folding` processes).

Initial Conditions

Section 1.4 outlines the cell state initialization algorithm. Briefly, after the `Mass` state initializes the total cell mass, the `Protein Monomer` state initializes the total copy number of each protein monomer species and initializes all monomers to their mature – processed, folded, modified, and localized – configuration. Second, the `Macromolecular Complexation` and `Ribosome Assembly` processes initialize the copy number of each ribonucleoprotein complex and set all initialized complexes to their mature – folded and modified – configuration. Finally, several processes including `Transcription` and `Translation` initialize the detailed configurations of RNA polymerases, 70S ribosomes, DNA-binding proteins, and FtsZ.

The nascent and processed configurations of each protein monomer species are initialized with zero copy number because the typical occupancy these configurations is small compared to that of the mature configuration.

Dynamic Computation

Algorithm S19 outlines the implementation of the protein processing (I) model.

Algorithm S19 | Protein processing (I) simulation. See Mathematical Model section above for definition of the mathematical notation.

```
Input: $p_i^p$ copy number of processing protein monomer species $i$
Let $k_i \leftarrow e_i \Delta t$ be the capacity of enzyme $i$ for protein processing
repeat
    Calculate processing rates
    foreach protein monomer species $i$ do
        Calculate $v_i$ according to Eq. S21
    Select protein monomer species $i \sim$ multinomialRand(1, $v_i / \sum_j v_j$)
    Update protein monomer copy numbers: $p_i^u \leftarrow p_i^u - 1$, $p_i^p \leftarrow p_i^p + 1$
    Update metabolites: $m \leftarrow m - M_{\bullet i}$
    Update enzyme catalytic capacity: $k \leftarrow k - K_{\bullet i}$
until no further processing possible ($v_i = 0 \; \forall \; i$)
```

Fitting

The expression of the protein processing enzymes was fit to provide sufficient enzymes to quickly process proteins, or more specifically to prevent sustained accumulation of unprocessed proteins. See Section 1.3 for further discussion.

3.16 Protein Processing II

Biology

This process models the third step of post-translational processing: lipoprotein diacylglyceryl adduction and lipoprotein and secreted protein signal peptide cleavage.

Lipoproteins

As discussed in Section 3.17, *M. genitalium* lipoproteins are translated in the cytosol and subsequently targeted to the plasma membrane by type II N-terminal signal sequences. Following membrane insertion, *M. genitalium* lipoproteins are first anchored to the outer leaflet of the plasma membrane. This is achieved via covalent adduction of diacylglyceryl to the sulfhydryl group of the lipobox cysteine by diacylglyceryl transferase (MG086)[266,629,654]. Many bacteria further anchor lipoproteins through phospholipidation. However, *M. genitalium* does not have an apolipoprotein transacylase, and is not believed to phospholipidate lipoproteins[655–657]. Second, lipoprotein N-terminal signal sequences are cleaved immediately C-terminal to the lipobox cysteine by type II signal peptidase (MG210)[654]. See Section 3.17 for further discussion of the structure of type II signal sequences.

Secreted Proteins

Extracellular *M. genitalium* proteins are transcribed in the cytosol and targeted to the plasma membrane by type II signal sequences. In contrast to lipoproteins, extracellular proteins do not undergo diacylglyceryl transfer, and instead are cleaved immediately C-terminal to the cysteines of their lipoboxes by type II signal peptidase, releasing the resultant protein and signal peptide into the extracellular space.

Integral Membrane Proteins

M. genitalium does not have a type I signal sequence protease[634] and does not cleave the signal sequences of integral membrane proteins. Integral membrane protein signal peptides are believed to help anchor proteins to the membrane[3].

Reconstruction

The localization and N-terminal signal sequence length of each protein monomer was reconstructed based on an extensive review of the primary literature, several proteomic database, and several computational predictions. See Section 3.17 for further discussion. The stoichiometry and kinetics of each protein processing reaction was reconstructed from the primary literature[8,158,167,266,654]. Table S3O lists the reconstructed stoichiometry and kinetics of each reaction.

Computational Representation

Mathematical Model

This process models lipoprotein diacylglyceryl adduction and lipoprotein and secreted protein signal peptide cleavage, the third step of post-translational processing. Because the mechanisms of protein processing are not well characterized on the genomic scale, this process make several simplifying assumptions. First, this process assumes that each protein is both covalently modified and cleaved in a single time step. Consequently, this process collapses the processing of each protein into a single reaction and only represents unprocessed and fully processed proteins. Intermediate protein configurations are not represented. Second, this process assumes that the mean arrival rate, v_i of processing events of each protein species i is independently limited by (1) the copy number of unprocessed protein, p_i^u, (2) the copy numbers of intracellular metabolites, m_j, and (3) the copy numbers of processing enzymes, e_j. Based on these assumptions, the functional form of v_i

is given by

$$v_i = \min\left(\overbrace{p_i^u}^{\text{Protein}},\ \overbrace{\left\lfloor \min_j \frac{m_j}{M_{ji}^s} \right\rfloor}^{\text{metabolites}},\ \overbrace{\texttt{poissonRand}\left(\min_j \frac{e_j}{K_{ji}} \Delta t\right)}^{\text{enzymes}}\right), \tag{S22}$$

where M_{ji} is the stoichiometry of metabolite j in the processing of protein species i, $M^s = \max(0, -M)$ is the negative part of M, K_{ji} is the experimentally observed catalytic rate of enzyme j in the processing of protein species i, and $\Delta t = 1\,\text{s}$ is the simulation time step.

This process implements a stochastic model of the arrival of protein monomer processing events with relative rates v_i. Until protein, metabolic, and/or enzymatic resources are exhausted, the model iteratively (1) computes the arrival rate, v_i, of each processing event, (2) selects a single processing event to execute according to a multinomial distribution parameterized by v_i, and (3) executes the selected processing reaction, updating the copy numbers of protein monomers and metabolites and decrementing the available enzymatic capacity. Algorithm S20 outlines the implementation of the protein processing model.

Integration
The `Protein Monomer` and `Protein Complex` states represent the copy number of each protein processing enzyme. The `Protein Monomer` state also represents the unprocessed and processed copy numbers of each protein monomers species. The `Metabolite` state represents the copy number of each intracellular metabolite.

Protein are synthesized and matured in six steps (see Section 2.10). This process models lipoprotein diacylglyceryl transfer and lipoprotein and secreted protein signal sequence cleavage following translocation (see `Protein Translocation` process) and preceding folding and modification (see `Protein Folding` and `Protein Modification` processes).

Initial Conditions
Section 1.4 outlines the cell state initialization algorithm. Briefly, after the `Mass` state initializes the total cell mass, the `Protein Monomer` state initializes the total copy number of each protein monomer species and initializes all monomers to their mature – processed, folded, modified, and localized – configuration. Second, the `Macromolecular Complexation` and `Ribosome Assembly` processes initialize the copy number of each ribonucleoprotein complex, and set all initialized complexes to their mature – folded and modified – configuration. Finally, several processes including `Transcription` and `Translation` initialize the detailed configurations of RNA polymerases, 70S ribosomes, DNA-binding proteins, and FtsZ.

The unprocessed and processed configurations of each protein monomer species are initialized with zero copy number because the typical occupancy these configurations is small compared to that of the mature configuration.

Dynamic Computation
Algorithm S20 outlines the implementation of the protein processing (II) model.

Fitting
The expression of the protein processing enzymes was fit to provide sufficient enzymes to quickly process proteins, or more specifically to prevent sustained accumulation of unprocessed proteins. See Section 1.3 for further discussion.

Algorithm S20 | Protein processing (II) simulation. See Mathematical Model section above for definition of the mathematical notation.

```
Input: $p_i^p$ copy number of processed protein monomer species $i$
Input: $p_i^s$ copy number of cleaved signal sequence of protein monomer species $i$
Let $k_i \leftarrow e_i \Delta t$ be the capacity of enzyme $i$ for protein processing
repeat
    Calculate processing rates
    foreach protein monomer species $i$ do
        Calculate $v_i$ according to Eq. S22
    Select protein monomer species $i$ ~ multinomialRand(1, $v_i / \sum_j v_j$)
    Update protein monomer copy numbers: $p_i^u \leftarrow p_i^u - 1$, $p_i^p \leftarrow p_i^p + 1$, $p_i^s \leftarrow p_i^s + 1$
    Update metabolites: $m \leftarrow m - M_{\bullet i}$
    Update enzyme catalytic capacity: $k \leftarrow k - K_{\bullet i}$
until no further processing possible ($v_i = 0 \; \forall \; i$)
```

3.17 Protein Translocation

Biology

Bacterial proteins are translated in the cytosol. However, many functionally important proteins including nutrient transporters, ATP synthase, metabolic enzymes, adhesins, receptors, transducers, virulence factors, and the protein translocation machinery itself must be embedded in the cell membrane or secreted in order to properly function[658]. Cells employ short N-terminal and C-terminal signal sequences to selectively translocate proteins. This process models membrane and extracellular protein localization, the second step in post-translational processing.

Reconstruction

The subcellular localization – cytosol, integral membrane, lipoprotein, terminal organelle cytosol, terminal organelle lipoprotein, or extracellular space – of each protein monomer and the N-terminal signal sequence of each lipoprotein and secreted protein was reconstructed based on an extensive review of the primary literature[88,89,91–93,284,303,406–409], several proteomic databases[96,253,386,570,572–574], and several computational predictions[252,254,255,257–263]. Table S3AC-S3AE describe the reconstruction process in detail and list the reconstructed localization and signal sequence of each protein monomer.

M. genitalium employs two convergent SecA-dependent pathways – co-translational and post-translational – to translocate approximately 35-45% of all protein monomers into the plasma membrane[3]. *M. genitalium* does not contain a type I Tat transporter or sortase[631]. The co-translational SecA pathway translocates integral membrane proteins into the cell membrane. First, GTP-dependent signal recognition particles (SRP; MG0001, MG048) and molecular chaperones co-translationally recognize the type I signal sequence of nascent integral membrane proteins[3]. Second, SRPs deliver nascent proteins to their cognate receptor FtsY (MG297) which is associated with the SecA type II preprotein translocon[3]. Finally, the preprotein translocase SecA (MG072) iteratively pushes nascent proteins through the SecYEG (MG170, MG055, MG476) translocase pore by an ATP-dependent, step-wise mechanism[3]. Several studies have shown that SecDF (MG277)[6] and YidC (MG464)[57,328] also participate in the translocase pore. SecDF and YidC are believed to increase the efficiency of protein translocation[3]. The exact functions of SecDF and YidC are not known.

The *M. genitalium* post-translational SecA pathway translocates lipoproteins and secreted proteins. First, nascent proteins complete translation. Second, SecA translocons directly recognize the type II signal sequence of nascent proteins. Type II signal sequences are short, positively charged N-terminal sequences[3]. Type II signal sequences are composed of three regions – n, h, and c. The n region is composed of 1-5 positively charged amino acids. The h region is composed of 7-15 hydrophobic amino acids and forms an α-helix. The c region is composed of 3-7 polar amino acids and forms a β-strand. Signal peptidase II cleaves type II signal sequences at lipoboxes distinguished by the sequence `L[ASI][GA]C` inside the c region. Finally, similar to the co-translational pathway, SecA iteratively translocates nascent proteins into the plasma membrane.

Following translocation *M. genitalium* anchors lipoproteins to the outer leaflet through the covalent ligation of diacylglyceryl by diacylglyceryl transferase and extracellularly cleaves the type II signal sequence of each lipoprotein and secreted protein at conserved lipoboxes (see `Protein Processing II`). *M. genitalium* does not have a I signal sequence protease[634] and does not cleave the signal sequences of integral membrane proteins. Integral membrane protein signal peptides are believed to help anchor proteins to the membrane[3].

Following insertion into the plasma membrane and signal peptide cleavage, terminal organelle-localized lipoproteins are recruited into the terminal organelle. See the `Terminal Organelle Assembly` process for further discussion.

The stoichiometry, energetics, and kinetics of each protein translocation reaction were reconstructed from the primary literature[1,2,10] (see Table S3D and S3O). Tomkiewicz *et al.* reported that SecA translocates 270 pmol amino acid min^{-1}[1]. Doyle *et al.* reported that SecA translocates 20-30 amino acids per ATP[2].

Computational Representation

Mathematical Model
This process models integral membrane, lipoprotein, and secreted protein translocation into the cell membrane by the SecA preprotein translocase, the second step of post-translational processing. Because the mechanisms of protein translocation are not well characterized on the genomic scale, this process make several simplifying assumptions. First, this process decouples translation, N-terminal formylmethionine processing, and translocation by assuming that translocation does not begin until after translation termination and N-terminal formylmethionine processing. Second, this process assumes that each protein is fully translocated in a single time step. Consequently, this process collapses the translocation of each protein into a single reaction, and only represents untranslocated and fully translocated proteins. Intermediate protein localizations are not represented. Third, this process assumes that the mean arrival rate, v_i of translocation events of each protein species i is independently limited by (1) the copy number of untranslocated protein, p_i^u, (2) the copy numbers of intracellular metabolites, m_j, and (3) the copy numbers of the translocation enzymes, e_j. Based on these assumptions, the functional form of v_i is given by

$$v_i = \min\left(\overbrace{p_i^u}^{\text{Protein}},\ \overbrace{\left\lfloor \min_j \frac{m_j}{M^s_{ji}} \right\rfloor}^{\text{metabolites}},\ \overbrace{\texttt{poissonRand}\left(\min_j \frac{e_j}{K_{ji}} \Delta t \right)}^{\text{enzymes}} \right), \tag{S23}$$

where M_{ji} is the stoichiometry of metabolite j in the translocation of protein species i including ATP and GTP hydrolysis, $M^s = \max(0, -M)$ is the negative part of M, K_{ji} is the experimentally observed catalytic rate of enzyme j in the translocation of protein species i, and $\Delta t = 1\,\text{s}$ is the simulation time step.

This process implements a stochastic model of the arrival of protein monomer translocation events with relative rates v_i. Until protein, metabolic, and/or enzymatic resources are exhausted, the model iteratively (1) computes the arrival rate, v_i, of each translocation event, (2) selects a single translocation event to execute according to a multinomial distribution parameterized by v_i, and (3) executes the selected translocation reaction, updating the copy numbers of protein monomers and metabolites and decrementing the available enzymatic capacity. Algorithm S21 outlines the implementation of the protein translocation model.

Integration
The `Protein Monomer` and `Protein Complex` states represent the copy number of each protein translocation enzyme. The `Protein Monomer` state also represents the cytosolic- and membrane-localized copy numbers of each protein monomers species. The `Metabolite` state represents the copy number of each intracellular metabolite.

Protein are synthesized and matured in six steps (see Section 2.10). This process models protein translocation following deformylation and N-terminal methionine cleavage (see Protein Processing I process) and preceding lipoprotein diacylglyceryl transfer and lipoprotein and secreted protein signal sequence cleavage (see Protein Processing II process).

Initial Conditions
Section 1.4 outlines the cell state initialization algorithm. Briefly, after the Mass state initializes the total cell mass, the Protein Monomer state initializes the total copy number of each protein monomer species and initializes all monomers to their mature – processed, folded, modified, and localized – configuration. Second, the Macromolecular Complexation and Ribosome Assembly processes initialize the copy number of each ribonucleoprotein complex, and set all initialized complexes to their mature – folded and modified – configuration. Finally, several processes including Transcription and Translation initialize the detailed configurations of RNA polymerases, 70S ribosomes, DNA-binding proteins, and FtsZ.

The untranslocated and translocated configurations of each protein monomer species are initialized with zero copy number because the typical occupancy these configurations is small compared to that of the mature configuration.

Dynamic Computation
Algorithm S21 outlines the implementation of the protein translocation model.

Algorithm S21 | Protein translocation simulation. See Mathematical Model section above for definition of the mathematical notation.

```
Input: $p_i^t$ copy number of translocated (membrane- or extracellular-localized) protein monomer species $i$
Let $k_i \leftarrow e_i \Delta t$ be the capacity of enzyme $i$ for protein translocation
repeat
    Calculate translocation rates
    foreach protein monomer species $i$ do
        Calculate $v_i$ according to Eq. S23
    Select protein monomer species $i \sim$ multinomialRand(1, $v_i / \sum_j v_j$)
    Update protein monomer copy numbers: $p_i^u \leftarrow p_i^u - 1$, $p_i^t \leftarrow p_i^t + 1$
    Update metabolites: $m \leftarrow m - M_{\bullet i}$
    Update enzyme catalytic capacity: $k \leftarrow k - K_{\bullet i}$
until no further translocation possible ($v_i = 0 \; \forall \; i$)
```

Fitting
The expression of the protein translocation enzymes was fit to provide sufficient enzymes to quickly translocate proteins, or more specifically to prevent sustained accumulation of untranslocated proteins. See Section 1.3 for further discussion.

3.18 Replication

Biology

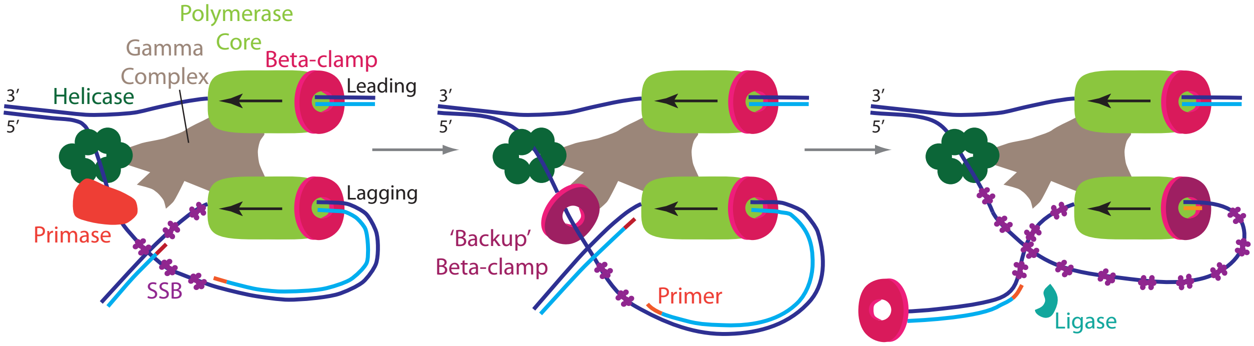
DNA replication is an integral part of the cell cycle which produces a complete chromosome for each daughter cell. DNA replication is initiated by the formation of a large multimeric DnaA complex at the origin of replication (oriC), which enables the recruiting of the DNA replication machinery to the oriC. Replication proceeds bidirectionally from the oriC to the terminus (terC), half way around the circular chromosome. The chromosome has two strands of base pairing DNA called the leading and lagging strands. Replication of the leading strand proceeds continuously the 5' to 3' direction. Replication of the lagging strand occurs in short Okazaki fragments, also in the 5' to 3' direction[927].

Reconstruction

The DNA replication machinery consists of multiple proteins (See Schematic S11), and there is one set of machinery for each of the two replication bubbles (on each side of the oriC). The replicative DNA helicase serves to unwind the coiled DNA[628]. The DNA primase makes short primers that help initiate the polymerization of a long stretch of DNA. Primers are made once at the origin for the replication of the leading strand, and one primer is made to start each Okazaki fragment[606]. DNA polymerization is carried out by DNA polymerase III molecules, consisting of two core subunits, a gamma complex, and beta-clamps[739]. One core resides on each of the leading and lagging strands, and is made up of two alpha subunits. The two cores are held together by a gamma complex, consisting of delta, delta prime, gamma, and tau subunits. The core is also bound to a sliding beta-clamp which helps anchor the polymerase to the DNA and maintain processivity. On the lagging strand, beta-clamps are swapped out each time a new DNA loop is created to make a new Okazaki fragment (See Schematic S11). A "back-up" beta-clamp binds downstream of the lagging DNA loop to facilitate this switch and formation of the next loop[739]. (The lagging stand polymerization requires the formation of DNA loops, so that the DNA can be polymerized in the opposite, 5' to 3', direction.) DNA ligases connect the separate polymerized Okazaki fragments together[733]. Finally, single-stranded binding proteins (SSBs) stabilize and protect single-stranded DNA, which is created as the DNA is unwound to make room for the DNA machinery[637,857]. For example, the lagging stand DNA loop often has long stretches of unwound DNA waiting to be polymerized. All of the replication proteins are described in List S15, and all of the parameters used in replication are described in List S16.

List S15. Enzymes and complexes used in the `Replication` process class.

Enzymes/Complexes	Composition	Gene Name(s)	DNA Footprints (nt)
DNA helicase	(6) MG094	dnaB	20
DNA primase	(1) MG250	dnaG	14
β-clamp	(2) MG001	dnaN	25
DNA polymerase core	(1) MG031, (1) MG261	polC, polC-2	24
γ-complex	(1) MG007, (1) MG351, (4) MG419	holB, holA, dnaX	26
DNA ligase	(1) MG254	ligA	19
Single stranded binding protein (SSB) 8mer	(8) MG091	ssb	145



Schematic S11. Schematic of DNA replication.

Computational Representation

Upon replication initiation (the binding of 29 DnaA-ATP molecules near the oriC by the `Replication Initiation` process class), the `Replication` process class tracks the progression of the replication proteins on the known chromosome sequences. Some bacterial species are known to have multiple replication initiation events during their life cycles. This has never been demonstrated in *M. genitalium* and we model up to 1 chromosome duplication event per cell cycle. Further, the exact mechanism of replication initiation in *M. genitalium* is unknown. *M. genitalium* does not include a DnaC homolog, which in other species is an

List S16. Fixed parameters used in the `Replication` process class.

Parameter	Value	Symbol	Source
OriC position	Base: 1	oriC	[182]
TerC position	Base: 290038	terC	[182]
Primer length	11 nt	l_{prim}	[606]
DNA polymerase elongation rate	100 nt s^{-1}	k_{el}	[851]
SSB complex spacing	30 nt	s_{ssb}	[637]
Okazaki fragment mean length	1500 nt	l_{OF}	[623, 926, 927]
Ligase kinetic rate	0.04 s^{-1}	k_{lig}	[488]
Current Okazaki fragment length before "back-up" beta-clamp can bind	750 nucleotide	l_{beta}	Set as half l_{OF}
Length of lagging strand loop at the start of Okazaki fragment polymerization	50 nt	l_{loop}	Set to be larger than polymerase footprint
DNA polymerase stall time upon anti-direction collision with RNA polymerase	1.7 s	t_{stall}	[799]
SSB dissociation rate	0.3 s^{-1}	k_{dssb}	[857]
Chromosome length	580076		[182]
Chromosome sequence	See [182]		[182]
DNA footprints of Proteins	See 'Enzymes/ Complexes'		[628, 733, 739, 743]
ATP cost of beta-clamp binding	1 ATP molecule	e_{beta}	[628]
ATP cost of 1 base pair unwinding	1 ATP molecule	e_{hel}	[739]
NAD requirement per ligation event	1 NAD molecule	e_{lig}	[733]

essential cue for the binding of the replication machinery to the oriC. Here, the binding of 29 DnaA-ATP molecules to the oriC is the cue for replication initiation.

To duplicate the chromosome, the exact position of each replisome protein is tracked over time on both the leading and lagging strands. The polymerization of Okazaki fragments is explicitly modeled. Since the exact primer binding sites for *M. genitalium* Okazaki fragments are uncharacterized, the Okazaki fragment lengths are randomly determined based on a Poisson distribution centered around the mean fragment length (l_{OF}). Polymerization of both the leading and lagging strands is limited by the average rate of polymerization (k_{el}) in Mycoplasmas[851], the availability of nucleotides and energy, and the available protein binding sites on the chromosome. Further, polymerization of the leading strand is prevented if additional unwinding will lead to too many unprotected single-stranded bases on the lagging stand, or if the leading polymerase will progress over two Okazaki fragment length distances beyond the lagging strand polymerase. Leading strands are not polymerized past the terC, and the lagging strands are not polymerized past the ends of Okazaki fragments.

This process outputs the progress of replication, which has a big impact on many other processes in the system. For example, the copy number of each gene plays a role in RNA polymerase binding in the Transcription process, and affects the cell's gene expression. The duplicated regions of the chromosome have to be wound appropriately by the DNA supercoiling process, and the completion of replication triggers other cell cycle events such as the decatenation of the chromosomes by the Chromosome Segregation process.

Collisions of the replication machinery with the transcription machinery (as determined by the `Chromosome` state class) are also handled by the `Replication` process class. A helicase may collide with a RNA polymerase in two ways, head-on (RNA polymerase is traveling in a direction opposite to the helicase) or co-directionally (RNA polymerase is traveling in the same direction as the helicase)[638]. There is some debate in the literature regarding the pausing of the replication loop upon collisions and whether RNA polymerases are displaced upon collisions[638,659,711,852,853]. Based on these studies as well as modeling considerations, we assume that the RNA polymerase is always displaced upon collisions, and the replication loop is only stalled upon a head-on collision. If the helicase hits an RNA polymerase, polymerization pauses, the RNA polymerase falls off, and its transcript is degraded. If it is a head-on collision (the helicase and RNA polymerase were traveling in opposite directions on the DNA), the impact will stall progression of the replication bubble for some amount of time (t_{stall}). If it is a codirectional collision, polymerization will continue at full speed the following time step[638,659,711].

Integration
List S17 outlines the states integrated with the `Replication` process class.

List S17. State classes connected to the `Replication` process class.

Connected States	Read from state	Written to state
`Chromosome`	▪ Whether a DnaA complex has formed at oriC ▪ DNA-bound protein locations ▪ Superhelicity ▪ DNA strand breaks to be ligated ▪ DNA sequence ▪ DNA footprints of proteins ▪ Chromosome regions accessible for protein binding ▪ Damaged DNA bases ▪ OriC position ▪ TerC position ▪ Sequence Length	▪ Polymerized regions of DNA ▪ DNA-bound protein locations ▪ Unwound bases (Effect on superhelicity) ▪ DNA strand breaks to be ligated

Initial Conditions
At the beginning of the simulation, we choose Okazaki fragment lengths randomly based on a Poisson distribution around the mean length of 1500 nucleotides (l_{OF}). From this we obtain the start position of each Okazaki fragment, to be used to determine primer, beta-clamp, and polymerase core binding during the simulation.

Dynamic Computation
The `Replication` process class is built up of 8 subfunctions which move the replication proteins along the chromosomes, and evolve the DNA copy number of the cell from 1 to 2. These subfunctions are evaluated in random order, such that shared resources across subfunctions, such as energy, are allocated fairly.

1. Initiate Replication

- If a 29-mer DnaA-ATP complex exists at the oriC,
  - If sufficient proteins and metabolites exist to make 2 replisomes,
    - Unwind enough DNA bases (b_{unwound}) to bind 2 sets of replication machinery. One set binds on either side of the oriC. All proteins are bound to the leading strand of DNA. Each set includes:
      - 1 helicase
      - 1 primase
      - 2 polymerase cores
      - 1 gamma complex
      - 1 beta-clamp
    - Account for total ATP usage (as well as H_2O usage, and ADP, Pi, and H^+ production):

$$\text{ATP usage} = 2e_{\mathrm{beta}} + e_{\mathrm{hel}}b_{\mathrm{unwound}}$$

- Note: following the initiation of replication, the dissociation of the DnaA complex is handled by the `Chromosome` state class.

2. Advance replisomes

- If an Okazaki fragment is starting at the current timestep,
  - Bind a primase and a polymerase core to the lagging strand
- Model the unwinding and polymerizing of the DNA. The leading and lagging polymerases and helicases are advanced only so far as the following conditions are met:
  Let N = the number of polymerized/unwound bases on a given strand in the given timestep.
  Let P be the last polymerized position. P varies from 0 (at the oriC) to 290038 (at the terC).

  - $N \leq l_{\text{prim}}$ if a primer is being synthesized
  - $N \leq k_{\text{el}}$
  - Available dNTPs $\leq dNTP$ requirements
  - Available ATP $\leq Ne_{\text{hel}}$
  - # Bound SSBs $= \frac{\text{Length of lagging single strand region}}{\text{SSB 8mer footprint} \times s_{\text{ssb}}}$
  - Upstream DNA regions accessible to replisome proteins (no DNA damage, non-displacable proteins)
  - Leading strand $P \leq terC$, Lagging strand $P \leq Okazakifragmentlength$
  - Leading strand $P \leq 2L_{\text{OF}} +$ lagging strand P
  - $P \leq$ position of colliding RNA polymerase

3. Bind and Unbind SSBs

SSBs nucleate as a tetramer, binding single-stranded DNA around them. These tetramers generally bind in sets of two.

- For all single-stranded stretches of DNA,
  - Randomly bind SSBs (as 8mers) to deterministically selected position (with fixed spacing: s_{ssb})
- Randomly release SSB 8mers according to the dissociation rate (k_{dssb})
- Dissociate all free 8mer SSBs into two tetramer SSBs

4. Bind "Back-up" Beta-Clamp

The formation of a new Okazaki fragment requires the formation of a new lagging strand DNA loop. To prepare for this event, while the current Okazaki fragment is being polymerized, a "back-up" beta-clamp is bound just downstream of the start position of the next Okazaki fragment. The beta-clamp assembles as a dimer on the chromosome, and the binding event requires hydrolysis of one ATP molecule to ADP (e_{beta}).

- If $\begin{cases} \text{\# available beta-clamp monomers} > 2 \\ \text{\# available } ATP > e_{\text{beta}} \\ \text{Position on the DNA is accessible} \\ \text{Helicase has passed beta-clamp binding site} \\ \text{Okazaki fragment length} > l_{\text{beta}} \end{cases}$
  - Bind "back-up" beta-clamp
  - Decrement ATP, H_2O, and increment ADP, Pi, and H^+

5. Terminate Okazaki fragments

When an Okazaki fragment has been completely polymerized, its beta-clamp is released, and the end of the Okazaki fragment is marked as having a single-strand break to be ligated by DNA ligase. The lagging strand primase and polymerase associate with the "back-up" beta clamp that was previous bound. This forms a new lagging strand DNA loop.

- If $\begin{cases} \text{Okazaki fragment has been completely polymerized} \\ \text{Number of bound SSBs} = \frac{\text{Length of lagging single strand region}}{\text{SSB 8mer footprint} \times s_{\text{ssb}}} \\ \text{A "back-up" clamp has been bound on the lagging strand} \\ \text{The replication machinery on the leading strand has advanced beyond the Okazaki start site} \end{cases}$
  - Release beta-clamp
  - Mark Okazaki fragment as having a single-strand break to be ligated by DNA ligase
  - Associate lagging strand primase and polymerase with the "back-up" beta clamp that was previously bound (This forms a new lagging strand DNA loop.)

6. Ligate DNA

- If single-strand breaks exist (usually between polymerized Okazaki fragments),
  - Ligate these breaks in random order up to the limits of:

    - DNA ligase availability
    - Ligase kinetics (k_{lig})
    - NAD availability
  - Decrement NAD, and increment AMP, NMM, and H^+

7. Terminate Replication

- If leading and lagging strands to both sides of the terC have been completely polymerized,
  - Release all replication machinery from the chromosome
  - Mark terC as having a single-strand break to be ligated by DNA ligase

Representation of Replication Machinery on the Chromosome
The Replication process class involves keeping track of the specific positions of the replication proteins on the chromosome. We define where on the chromosome the proteins bind based on the following rules:

1. The Helicase is centered on the boundary between double stranded DNA (wound) and single-stranded DNA (unwound). The position over which it is centered is the next position to be unwound.
2. There is no gap between the helicase and leading strand DNA polymerase core or between the leading strand polymerase core and the leading strand beta-clamp. Therefore, movement of the helicase controls the movement of the leading strand polymerase core.
3. There is no gap between the lagging strand DNA polymerase core and the beta-clamp on the current Okazaki fragment.
4. The positions over which the polymerase cores are centered are the next position to be polymerized.
5. "Back-up" beta-clamps bind slightly upstream of the start site of the next Okazaki fragment to be polymerized such that when the next Okazaki fragment starts polymerizing, there will be no gap between the polymerase and beta-clamp, and the polymerase core will be centered on the Okazaki fragment start site.
6. At replication initiation, the mother strands are separated such that the leading polymerase cores are centered at oriC ±1 base and the helicases are 11 nucleotides (l_{prim}) ahead
7. During replication initiation (and the final step of replication after the last Okazaki fragment has completed), the lagging polymerase, beta-clamp, and primase are accounted for as part of a complex on the leading strand (containing also the helicase, leading polymerase, gamma complex, and leading beta-clamp). At all other times, the lagging polymerase, lagging beta-clamp, and primase are accounted for as a complex on a different strand. This allows for separate tracking of the leading and lagging polymerase positions.
8. SSBs are bound at a fixed spacing (s_{ssb}) in the single-stranded regions of DNA.

3.19 Replication Initiation

Biology

The Replication Initiation process determines when during the cell cycle chromosome duplication begins. This replication initiation time is therefore very important in determining the cell's division time.

Reconstruction

The mechanism of replication initiation used here is modeled after that described for *E. coli* by Messer involving the protein DnaA (MG469)[924]. Chromosome Replication begins when a complex of 29 DnaA-ATP molecules assembles near the replication origin, OriC, at specific DNA motifs called R1-R5. The assembly of this complex is rare because the DnaA molecules are titrated out by approximately 2000 additional binding sites, or "DnaA boxes" that exist all around the chromosome[925]. DnaA needs to be bound to ATP or ADP to bind to these sites, and binds on and off these sites throughout the cell cycle. Binding of the DnaA-ATP at the R1-R5 sites is cooperative, enabling the large complex to form at this specific location on the chromosome.

All of the parameters used in the Replication Initiation process class are described in List S18.

List S18. Fixed parameters used in the `Replication Initiation` process class.

Parameter	Value	Symbol	Source
Positions of all the DnaA binding sites on the chromosome(s)	(see Table S3L)		Calculated from motif in [819]
Factor by which DnaA-ATP to oriC site binding probability increases when other sites are bound	85.6909	C_{site}	See Parameter Fitting
Factor by which DnaA-ATP to oriC site binding probability increases when x*4 sized DnaA complex has formed at oriC	1	C_{state}	See Parameter Fitting
Rate for DnaA-ATP binding high affinity DnaA boxes	25 nM^{-1} h^{-1}	kb1ATP	[448]
Rate for DnaA-ATP binding medium affinity DnaA boxes	0.6 nM^{-1} h^{-1}	kb2ATP	[448]
Rate for DnaA-ATP dissociating from the DNA	20 h^{-1}	kd1ATP	[448]
Rate for DnaA-ADP binding high affinity DnaA boxes	2.5 nM^{-1} h^{-1}	kb1ADP	[448]
Rate for DnaA-ADP binding medium affinity DnaA boxes	0.61 nM^{-1} h^{-1}	kb2ADP	[448]
Rate for DnaA-ADP dissociating from the DNA	20 h^{-1}	kd1ADP	[448]
Rate for DnaA-ADP to DnaA-ATP regeneration	2.3 h^{-1}	k_Regen	[447]
Rate for DnaA-ADP to DnaA-ATP regeneration catalyzed by membrane lipids	0.018 g L^{-1}	K_Regen_P4	[447]

Parameter Assignment
All of the rate constants used in this process class were obtained from previous models of replication initiation[447,448]. The site and state cooperativity constants were fit according to the cell cycle length. The site and state cooperativity constants directly affect the time required for replication initiation. The total cell cycle length has been experimentally measured, and the time required for `Cytokinesis` and Chromosome `Replication` can be estimated from their process classes. Thus:

$$\mathrm{time}_{\mathrm{ReplicationInitiation}} = \mathrm{time}_{\mathrm{CellCycle}} - \mathrm{time}_{\mathrm{Cytoknesis}} - \mathrm{time}_{\mathrm{Replication}} \tag{S24}$$

The site and state cooperativity constants were fit such that on average the simulated cell divides in the experimentally measured amount of time.

Computational Representation

The methods we use to model replication initiation are based on existing models by Atlas *et al.* and Browning *et al.*[447,448]. First, DnaA rapidly associates with ATP, and in some cases DnaA-ATP can be converted to DnaA-ADP. DnaA-ATP and DnaA-ADP molecules bind to and release from the binding sites on the chromosome. The number of binding/unbinding events that occur in a given period of time is determined by the number of available DnaA-ATP and DnaA-ADP molecules and binding/unbinding rates calculated as in Browning *et al.*[448].

Binding Sites
In addition to the R1-R5 sites at the origin, there 2227 DnaA binding sites around the chromosome. All of the DnaA binding sites are based on the*M. genitalium* motifs described by Cordova *et al.*[819]. The motifs are 9 bases long. A high affinity site is defined as one that exactly matches the motif or its reverse complement (148 sites). A medium affinity motif is defined as one that matches 8 of the 9 bases in the motif or its reverse complement (2079 sites), and a low affinity site is one that only matches 7 of 9 bases. Our model allows binding to all of the high and medium affinity sites outside of the oriC region.

We recognize 5 DnaA binding sites in the OriC region: one high affinity (R4), three medium affinity (R1-R3), and one low affinity (R5), to mimic *E. coli*'s 5 R-sites[854] and the 5 R-sites predicted for *M. genitalium*[819]. These sites reside between the genes MG469 and MG470 (bases: 578581-579224). There are 9 high and medium affinity motifs in this region, but we only recognize 4 to match the *E. coli* pattern, and therefore ignore the sites at genome positions 578837, 578855, 578881, 578966, and 579139. R5 matches 7 of 9 bases of the motif, so it is a very weak binder of DnaA. Since we do not know its exact mechanism/purpose we bind this site after a 28-mer DnaA-ATP initiator complex at R1-R4 is formed, and the R5 binding triggers initiation. This is the only low affinity binding site included in our model.

7 States

The 28-mer initiator complex consists of 7 DnaA-ATP molecules bound to each of the R1-R4 sites[854]. This binding is split up into 7 serial states, and in each state one additional DnaA-ATP molecule binds to each of the R1-R4 sites. All four sites must be bound before transition to the next state.

Cooperative Binding

DnaA-ATP binding at the origin is cooperative. There are two forms of cooperativity in our model of replication initiation. The first type of cooperativity is state cooperativity, which helps increase the probability of transitioning into higher states. Once the binding within a state is completed, the cooperativity to transition into the next state is calculated as:

$$\text{State Cooperativity Factor} = C_{\text{state}}(\text{State} - 1) \tag{S25}$$

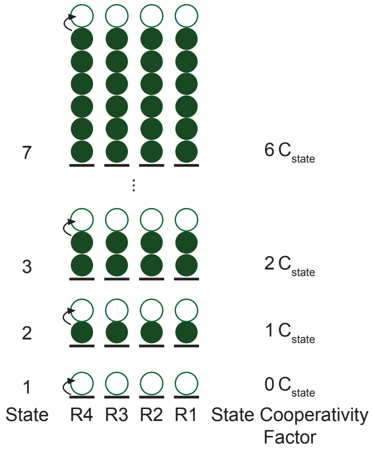
where C_{state} is a fixed cooperativity constant. This calculation is described in Schematic S12.

The second type of cooperativity is site cooperativity, which describes the effects of the binding of one or more of the R1-R4 sites, on the probability of binding the other sites. The site cooperativity factor of a given site is 1 unless certain other R-sites are already bound, in which case the cooperativity factor changes to C_{site}. R4 is a high affinity site, so it generally is the first to bind and has the highest cooperative effect on the other sites. The site cooperativity rules can be seen in Schematic S13).

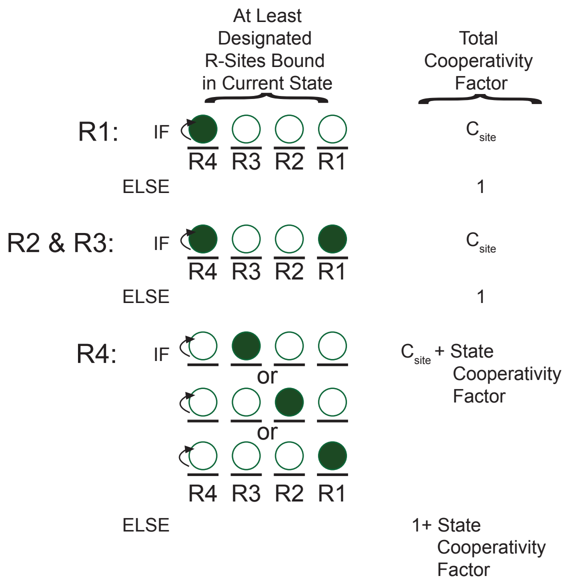
R5 is also bound by cooperativity, given the presence of a complete 28-mer DnaA-ATP initiator complex at the R1-R4 sites. The resulting DnaA-ATP 29-mer is the trigger to begin the `Replication` of the chromosome.

Post-Replication Initiation

Following initiation, the DnaA-ATP complex at the origin dissociates and hydrolyzes. DnaA molecules can continue to bind around the chromosome. The formation of a second complete initiation complex is rare because the time left in the cell cycle is generally less than that needed for an initiation event and because DnaA molecules are further titrated with a 2^{nd} set of binding sites on the 2^{nd} chromosome. For simplicity, we do not model a second DNA replication event.



Schematic S12. Calculation of the state cooperativity factor for DnaA-ATP binding at the origin of replication.


At Least Designated R-Sites Bound in Current State
Total Cooperativity Factor
R1:
IF
R4
R3
R2
R1
C_site
ELSE
1
R2 & R3:
IF
R4
R3
R2
R1
C_site
ELSE
1
R4:
IF
or
or
R4
R3
R2
R1
C_site + State Cooperativity Factor
ELSE
1+ State Cooperativity Factor

Schematic S13. Calculation of the total cooperativity factor for DnaA-ATP binding at the origin of replication.

Integration

List S19. State classes connected to the `Replication Initiation` process class.

Connected States	Read from State	Written to State
`Chromosome`	▪ Positions on chromosome(s) where DnaA molecules are bound ▪ Accessible DnaA binding sites ▪ Copy numbers of DnaA binding sites	▪ Updated positions on chromosome(s) where DnaA molecules are bound
`Mass`	▪ Membrane mass (used to calculate membrane concentration)	
`Geometry`	▪ Cell volume (used to calculate membrane concentration)	

Initial Conditions

This process is initialized to a steady state. The steady state amounts of free, medium affinity site bound, and high affinity site bound DnaA-ATPs and DnaA-ADPs are found using non-linear constrained optimization where we try to identify a state which is a stable point and which maximizes the amount of high affinity site bound DnaA-ATP. All DnaA proteins are either ADP or ATP bound at the initial timestep. There are also no DnaA polymers at the R1-R4 sites at the start of the simulation.

Dynamic Computation

At each timestep, we perform the following algorithm:

1. DnaA activation

- Deterministically form DnaA-ATP complexes up to the limit of available DnaA and ATP. The kinetics of DnaA activation are not known, and are not modeled.

$$\text{Number of activation events} = \min \begin{cases} \text{Number of available ATP molecules} \\ \text{Number of free DnaA molecules} \end{cases}$$

- Update the counts of free DnaA, DnaA-ATP, and ATP

2. DnaA-ATP complex dissociation

A complex of multiple bound DnaA-ATPs is only found near the oriC. This complex can be released from the chromosome by the DNA polymerase at the start of DNA replication. The resulting DnaA-ADP molecules are able to re-bind to the chromosome or reactivate to DnaA-ATP.

- If 29-mer DnaA-ATP complex is disrupted by the replisome machinery,
  - Dissociate the complex to individual DnaA-ATP molecules
  - Hydrolyze the ATP to form free DnaA-ADP molecules
  - Decrement the counts of DnaA-ATP complexes and H_2O molecules
  - Increment the counts of free DnaA-ADP, H^+, and Pi

3. Binding DnaA-ATP and DnaA-ADP
Up to 7 DnaA-ATPs can polymerize at each of the origin sites and only one DnaA-ATP can bind at each site outside of the origin. Only one DnaA-ADP can bind at each site.

- If the chromosomes are sufficiently supercoiled,
  - Stochastically select both high and low affinity sites at the origin and around the chromosome to bind DnaA-ATP and DnaA-ADP molecules. Binding proceeds such that the rate of binding each site is as follows (where the cooperativity factor, depends on the polymerization status of the R1-R4 boxes):

$$\begin{aligned}\text{rate of DnaA-ATP binding R4 (high affinity)} &= \frac{\text{kb1ATP} \times \text{numFreeDnaAATP}}{\text{cell volume} \times \text{cooperativity factor}}\\ \text{rate of DnaA-ATP binding R1-R3 (medium affinity)} &= \frac{\text{kb2ATP} \times \text{numFreeDnaAATP}}{\text{cell volume} \times \text{cooperativity factor}}\\ \text{rate of DnaA-ATP binding non-origin high affinity sites} &= \frac{\text{kb1ATP} \times \text{numFreeDnaAATP}}{\text{cell volume}}\\ \text{rate of DnaA-ATP binding non-origin medium affinity sites} &= \frac{\text{kb2ATP} \times \text{numFreeDnaAATP}}{\text{cell volume}}\end{aligned}$$

  The rate equations for DnaA-ADP binding are the same as above, except that the rate constants are kb1ADP and kb2ADP.

4. Cooperativity
The binding of DnaA-ATP to the R1-R4 sites is cooperative at two levels. Site cooperativity increases the probability of binding successive sites within a state. State cooperativity increases the probability of entering successive states. The cooperativity factors for sites R1-R4 are calculated as follows:

- Calculate the State Cooperativity Factor as described in Schematic S12:

$$\text{State Cooperativity Factor} = C_{\text{state}}(\text{State} - 1)$$

- Calculate the Site Cooperativity as described in Schematic S13. The Site Cooperativity Factor for a given site is 1 or C_{site} depending on the occupancy of the other R-sites
- Assign the total Cooperativity Factor for sites R1, R2, and R3 as their respective Site Cooperativity Factors
- Assign the total Cooperativity Factor for R4 as its Site Cooperativity Factor+State Cooperativity Factor

The cooperativity factors calculated here will be used in the rate calculation in Step 3.

5. Displacing DnaA-ATP and DnaA-ADP
We model the release of DnaA-ATP and DnaA-ADP molecules from their binding sites throughout the cell cycle at the following rates:

- Stochastically select DnaA-ATP and DnaA-ADP molecules to unbind from both high and low affinity sites at the origin and around the chromosome, such that the number of unbinding events is in agreement with the following rates:

$$\text{rate of DnaA-ATP displacement} = kd1ATP$$
$$\text{rate of DnaA-ADP displacement} = kd1ADP$$

6. Reactivation of DnaA-ADP to DnaA-ATP
The rejuvenation of DnaA-ADP to DnaA-ATP is incorporated similarly to that described by Atlas *et al.*[447]. This reaction is promoted by the acidic phospholipids cardiolipin and phosphatidylglycerol.

- Deterministically reactivate free DnaA-ADP at a rate of:

$$\text{rate of reactivation} = \text{numFreeDnaAADP}\frac{\text{k\_Regen membraneConcentration}}{\text{K\_Regen\_P4} + \text{membraneConcentration}}$$

  where the membrane concentration is the grams of membrane in the cell divided by the volume of the cell. These values are read from the `Mass` and `Geometry` state classes.

7. Replication-dependent bound DnaA-ATP release
Atlas *et al.* included a global term for the effect of active beta-clamps (a part of the DNA polymerase machinery) on bound DnaA-ATP inactivation[447]. Our model predicts the exact position of the active beta-clamps on the chromosome. Therefore, we do not need to use a global term, and instead can model the release of a DnaA-ATP molecule exactly when the beta-clamp encounters it on the DNA. We however cannot distinguish between free DnaA-ATP in the cytosol and DnaA-ATP that has been recently released by a beta-clamp. Therefore, we model the release in the form of DnaA-ATP and not the hydrolysis to DnaA-ADP. This beta-clamp-dependent release is handled by the DNA `Replication` process class and `Chromosome` state class.

3.20 Ribosome Assembly

Biology

Ribosomes are large ribonucleoproteins which synthesize polypeptides. The *M. genitalium* 70S ribosome is composed of two subunits – the 30S and 50S ribosomal particles – which assemble at the mRNA Shine-Dalgarno sequence with assistance from initiation factors 1-3[393,660]. This process models the enzyme-catalyzed formation of 30S and 50S ribosomal particles.

Reconstruction

Table S3N lists the observed subunit composition of the 30S and 50S particles[6]. The 30S particle is composed of 1 RNA and 20 protein monomer subunits. The 50S particle is composed of 2 RNA and 32 protein monomer subunits. The 30S and 50S particles have both been shown to assemble in stereotyped patterns with assistance from several GTPases[58,102,104,105,222,223,660,661]. Era (MG387) and RbfA (MG143) have been associated with 30S particle formation[104,105]. EngA (MG329), EngB (MG335), Obg (MG384), and RbgA (MG442) have been associated with 50S particle formation[104,105,222,223]. The exact functions, kinetics, and energetics of the six GTPases are unknown.

Computational Representation

Mathematical Model
Because 30S and 50S ribosomal assembly are not well characterized, this process makes several simplifying assumptions to model ribosomal particle assembly. First, the process only represents individual rRNA transcripts and protein monomers and fully formed ribosomal particles. Intermediate states of ribosomal particle formation are not represented. Second, this process assumes that ribosomal particle formation is kinetically fast and energetically favorable such that ribosomal particle formation proceeds to completion within the 1 s simulation time step. Finally, the process assumes that each GTPase hydrolyzes 1 GTP

molecule per ribosomal particle. With these assumptions, the rate of formation of ribosomal particle i is given by

$$\Delta c_i = \begin{cases} \min\left(\overbrace{\min_j \frac{r_j}{R_{ji}}}^{\text{rRNA}}, \ \overbrace{\min_j \frac{p_j}{P_{ji}}}^{\text{protein}}, \ \overbrace{\frac{m_{GTP}}{\eta_i}}^{\text{GTP}}\right) & \overbrace{\left(\min_j \frac{e_j}{E_{ji}}\right) > 0}^{\text{GTPases}} \\ 0 & \text{otherwise} \end{cases} \tag{S26}$$

where m, r, p, c, and e represent the copy number of each metabolite, rRNA transcript, protein monomer, ribosomal particle, and GTPase, R_{ij} and P_{ij} represent the stoichiometry of rRNA transcript or protein monomer i in ribosomal particle j, E_{ij} is true if ribosomal particle j formation requires GTPase i and false otherwise, and $\eta_i = \sum_j E_{ij}$ is the GTP cost of forming ribosomal particle i.

This process implements a stochastic model of the arrival of ribosomal particle assembly events with relative rates Δc_i. Until RNA, protein, metabolic, and/or enzymatic resources are exhausted, the model iteratively (1) computes the arrival rate, Δc_i, of each assembly event, (2) selects an event to execute according to a binomial distribution parameterized by Δc_i, and (3) executes the selected assembly reaction, updating the copy numbers of RNA, protein, and metabolites and decrementing the available enzymatic capacity. Algorithm S22 outlines the implementation of the Boolean ribosomal particle assembly model.

Integration

The `Rna` and `Protein Monomer` states represents the free copy numbers of each RNA and protein monomer species. The `Protein Complex` state represents the copy numbers of 30S and 50S ribosomal particles and the 70S ribosome. The `Ribosome` state represents the (t)mRNA location of each 70S ribosome.

Several processes including `Transcription` and `Translation` model the synthesis and maturation of rRNA transcripts and protein monomers (see Section 2.12 and 2.10). The `Translation` process models (1) translation initiation: 70S ribosome formation at mRNA Shine-Dalgarno sequences, (2) translation elongation: 70S ribosome-catalyzed polypeptide synthesis, and (3) translation termination: mRNA release and 70S ribosome disassembly. The `Translation` process also models ribosome stalling whereby the mRNA template is replaced by a tmRNA and the 70S ribosome synthesizes a SsrA degradation tag at the polypeptide C-terminus, marking the nascent polypeptide for degradation. The `Protein Activation` process models the effect of antibiotics on the catalytic activity of 30S and 50S ribosomal particles.

Initial Conditions

After the `Mass` state initializes the total cell mass, the `Rna` and `Protein Monomer` states initialize the total copy number of each rRNA and protein monomer species. Next, the `Ribosome Assembly` process initializes the total copy numbers of 30S and 50S ribosomal particles to a steady-state of the ribosome assembly model by evaluating Algorithm S22 with excess GTP. Finally, the `Translation` process initializes the copy number of 70S ribosomes, and randomly positions each 70S ribosome on mRNA weighted by the expressed copy number of each codon (see Algorithm S4).

Dynamic Computation

Algorithm S22 outlines the implementation of the Boolean ribosomal particle assembly simulation.

Fitting

The expression of each ribosomal RNA and protein monomer was fit to provide sufficient ribosomes for translation and prevent sustained amino acid accumulation. See Section 1.3 for further discussion.

Algorithm S22 | Ribosome assembly simulation. See Mathematical Model section above for definition of the mathematical notation.

```
foreach ribosomal particle $i$ in random order do
    Calculate the extent of ribosomal particle formation, $\Delta c_i$, using Eq. S26
    Form ribosomal subunits and decrement the copy numbers of rRNA transcripts and protein monomers.
    $r \leftarrow r - R_{\bullet i}\Delta c_i$
    $p \leftarrow p - P_{\bullet i}\Delta c_i$
    $c_i \leftarrow c_i + \Delta c_i$
    Hydrolyze GTP
    $m_{GTP} \leftarrow m_{GTP} - \eta_i\Delta c_i$
    $m_{GDP} \leftarrow m_{GDP} - \eta_i\Delta c_i$
    $m_{Pi} \leftarrow m_{Pi} - \eta_i\Delta c_i$
    $m_{H_2O} \leftarrow m_{H_2O} - \eta_i\Delta c_i$
    $m_{H+} \leftarrow m_{H+} - \eta_i\Delta c_i$
```

3.21 RNA Decay

Biology

In presence of ribonucleases such as ribonuclease R, RNAs have relatively short half-lives compared to that of other macromolecules (eg. protein, DNA) and the *M. genitalium* doubling time. The relatively short half-lives of RNAs enables the small *M. genitalium* with its very small pool of RNAs and particularly mRNAs to sample a broader range of configurations of the RNA pool over a shorter period that would be possible with longer half-lives. This helps the cell more finely tune the expression of proteins, more efficiently execute cell-cycle dependent events, and respond to the external environment. However, this enhanced fitness due to short RNA half-lives comes at a large energetic cost. In addition to ribonucleases, aminoacylated RNAs require peptidyl tRNA hydrolase to release their conjugated amino acids. This process decays all species of RNA, and at all maturation states including aminoacylated states.

Reconstruction

RNA decay is modeled as requiring ribonuclease R (MG104). The decay of tRNAs also requires peptidyl tRNA hydrolase (Pth: MG083). Given the availability of the necessary decay enzymes, RNAs are randomly selected for degradation by a Poisson probability distribution based on the RNA half lives. All of the parameters used in the Replication Initiation process class are described in List S20.

List S20. Fixed parameters used in the `RNA Decay` process class.

Parameter	Value	Symbol	Source
Rate of peptidyl tRNA hydrolase activity	0.7 s^{-1}	k_{hyd}	[26]
Half lives of all RNA species	See Table S3Y	$t_{1/2i}$	[602]
Decay rates of all RNA species		$k_{\text{decay }i}$	Derived from RNA half lives. See `Transcription` Process Class.
Reactants and products of decay reactions of all RNAs		M_{decay}	Computed from RNA sequences

Computational Representation

Half-lives of all the RNA species are largely based on experimental measurements of homologous *E. coli* genes. The *E. coli* genes are mapped to the *M. genitalium* genes by homology. (Refer to the `Transcription` Process Class for additional details regarding the determination of RNA half-lives.) RNA species are randomly selected to decay from a Poisson distribution based on the half-life of each RNA species. This process class contains no intermediate representation of RNA degradation. RNA degradation is treated as an all-or-nothing event that proceeds to completion in a single timestep. Upon degradation, water is used to break the nucleotide-nucleotide bonds and the nucleotides are recycled. All aborted transcripts (due to stalled RNA polymerases or RNA polymerases that have been knock off of the DNA) are also degraded by this process.

Integration

List S21. State classes connected to the `RNA Decay` process class.

Connected State	Read from state	Written to state
`Rna`	▪ Count of each RNA species ▪ Decay Rates of all RNAs	▪ Updated count of each RNA species
`Transcript`	▪ Aborted transcript sequences	

Initial Conditions
No initialization steps are required for this process.

Dynamic Computation

1. Determine the counts of free ribonuclease R (N_R) and peptidyl tRNA hydrolase (N_T) from the `Protein Monomer` State Class
2. Determine the limits on the number of RNA decay events, D_R:
   (The number of tRNA decay events is designated D_T)
   (a) If $N_R = 0$, then $D_R = 0$
   (All RNAs require ribonuclease R in order to decay, but ribonuclease R has a high rate of activity, so as long as $N_R > 0$, RNA decay can occur)
   (b) $D_T = N_T k_{\text{hyd}}$
   (The tRNA decay limit is dependent on the peptidyl tRNA hydrolase availability and kinetic rate)
3. Considering each RNA independently, decide whether to decay an RNA by random number selection from a Poisson distribution with a rate parameter, λcalculated as:

$$\lambda = RNA_i k_{\text{decay } i} \tag{S27}$$

   where RNA_i is the count of a given RNA species i
4. If aborted transcripts exist in the cell due to RNA polymerase stalling or RNA polymerase displacement on the DNA, decay all aborted sequences
5. Calculate the total number of metabolites used and produced from the decay of the given RNA species, using the matrix M_{decay}, update the counts of existing RNAs and metabolites.

3.22 RNA Modification

Biology

Bacteria employ post-transcriptional base modification in order to degenerately encode approximately 61 triplet codes using far fewer tRNA species than possible using only Watson-Crick base pairing[885]. Modification of wobble position 34 is believed to be most important for improving codon recognition[885]. Bacteria modify several positions in addition to position 34[885]. Modifications distant from the anticodon are believed to help tRNAs properly fold and stabilize their catalytically active structures[885]. Bacteria also post-transcriptionally modify rRNA[105]. rRNA modifications are believed to help stabilize rRNA, participate in protein synthesis, and confer resistance to ribosomal inhibitors[105]. The exact role of rRNA modification is unknown[105]. This process models tRNA and rRNA modification.

Reconstruction

The *M. genitalium* tRNA modification complement was reconstructed based on the observed complement of *E. coli* modifications. Table S3AA describes the reconstruction process in detail. First, the *E. coli* modification complement was reconstructed[45,47,55,56,64,65,560,651]. Second, modifications situated in conserved motifs were transferred to *M. genitalium*. The *M. genitalium* rRNA modification complement was similarly reconstructed (see Table S3Z). The stoichiometry and kinetics of each RNA modification reaction were reconstructed based on an extensive review of the literature (see Table S3O, S3Z, and S3AB).

Computational Representation

Mathematical Model

This process models non-coding RNA modification. Because the kinetics of RNA modification are not well characterized, this process make several simplifying assumptions. First, this process assumes that each RNA is fully modified in a single time step. Consequently, this process collapses the modification of each RNA into a single reaction, and only represents unmodified and fully modified RNA. Intermediate modification configurations are not represented. Second, this process assumes that the mean arrival rate, v_i of modification events of each RNA species i is independently limited by (1) the copy number of unmodified RNA, r_i^u, (2) the copy numbers of intracellular metabolites, m_j, and (3) the copy numbers of RNA modification enzymes, e_j. Based on these assumptions, the functional form of v_i is given by

$$v_i = \min\left(\overbrace{r_i^u}^{\text{RNA}},\ \overbrace{\left\lfloor \min_j \frac{m_j}{M^s_{ji}} \right\rfloor}^{\text{metabolites}},\ \overbrace{\texttt{poissonRand}\left(\min_j \frac{e_j}{K_{ji}}\Delta t\right)}^{\text{enzymes}}\right), \tag{S28}$$

where M_{ji} is the stoichiometry of metabolite j in the modification of RNA species i, $M^s = \max(0, -M)$ is the negative part of M, K_{ji} is the experimentally observed catalytic rate of enzyme j in the modification of RNA species i, and $\Delta t = 1\,\text{s}$ is the simulation time step.

This process implements a stochastic model of the arrival of RNA modification events with relative rates v_i. Until RNA, metabolic, and/or enzymatic resources are exhausted, the model iteratively (1) computes the arrival rate, v_i, of each modification event, (2) selects a single modification to execute according to a multinomial distribution parameterized by v_i, and (3) executes the selected modification reaction, updating the copy numbers of RNA and metabolites and decrementing the available enzymatic capacity. Algorithm S23 outlines the implementation of the RNA modification model.

Integration

The `Rna` state represents the unmodified and modified copy numbers of each RNA species. The `Metabolite` state represents the copy number of each intracellular metabolite. The `Protein Monomer` and `Protein Complex` states represent the copy number of each RNA modification enzyme.

RNA are synthesized and matured in four steps (see Section 2.12). This process models the modification of individual non-coding RNA following RNA processing (see `RNA Processing` process). The `Macromolecular Complexation` and `Ribosome Assembly` processes model the formation of macromolecular complexes, including the 30S and 50S ribosomal particles. The `Translation` process models the function of m-, r-, s-, and tRNA in translation.

Initial Conditions

Section 1.4 outlines the cell state initialization algorithm. Briefly, after the `Mass` state initializes the total cell mass, the `Rna` state initializes the total copy number of each RNA species and initializes all RNA to their mature – processed and modified – configuration. Second, the `tRNA Aminoacylation` process initializes tRNA to the aminoacylated configuration. Next, the `Macromolecular Complexation` and `Ribosome Assembly` processes initialize the copy number of each ribonucleoprotein complex. Finally, the `Translation` process initializes the mRNA location of each 70S ribosome.

Dynamic Computation

Algorithm S23 outlines the implementation of the RNA modification model.

Algorithm S23 | RNA modification simulation. See Mathematical Model section above for definition of the mathematical notation.

```
Input: $r_i^p$ copy number of modified RNA species $i$
Let $k_i \leftarrow e_i \Delta t$ be the capacity of enzyme $i$ for RNA modification
repeat
    Calculate modification rates
    foreach non-coding RNA species $i$ do
        Calculate $v_i$ according to Eq. S28
    Select non-coding RNA species $i \sim$ multinomialRand(1, $v_i / \sum_j v_j$)
    Update RNA copy numbers: $r_i^u \leftarrow r_i^u - 1$, $r_i^m \leftarrow r_i^m + 1$
    Update metabolites: $m \leftarrow m - M_{\bullet i}$
    Update enzyme catalytic capacity: $k \leftarrow k - K_{\bullet i}$
until no further modification possible ($v_i = 0 \; \forall \; i$)
```

Fitting
The expression of the RNA modification enzymes was fit to provide sufficient enzymes to quickly modify RNA, or more specifically to prevent sustained accumulation of unmodified RNA. See Section 1.3 for further discussion.

3.23 RNA Processing

Biology

Bacteria transcribe genes both individually as well in groups referred to as transcription units or operons. Operonic transcription has several advantages compared to single-gene transcription. First, operonic transcription allows cells to minimize the number of transcriptional regulators required to control gene expression. Second, operonic transcription increases the likelihood that groups of gene products have similar stoichiometry. At the same time, operonic transcription carries the costs of synthesizing intercistronic RNA and cleaving operonic non-coding transcripts into individual RNAs. Additionally, operonic transcription doesn't provide separate control of the expression of each gene. This process models operonic RNA cleavage into individual RNA gene products.

Reconstruction

Transcription Unit Structure
The transcription unit organization of the *M. genitalium* chromosome was reconstructed by mapping the "suboperon" structure of *M. pneumoniae* chromosome defined by Güell *et al.*[418] on to that of *M. genitalium* (see Table S3U) with two modifications. First, non-coding RNA were organized into transcription units according to their "reference operons"[418]. Second, all mRNAs either not associated with suboperons, without *M. pneumoniae* homologs, or which have been rearranged since divergence from *M. pneumoniae*, were assigned to their own transcription units. The *in silico M. genitalium* chromosome is organized into 335 transcription units containing 525 genes.

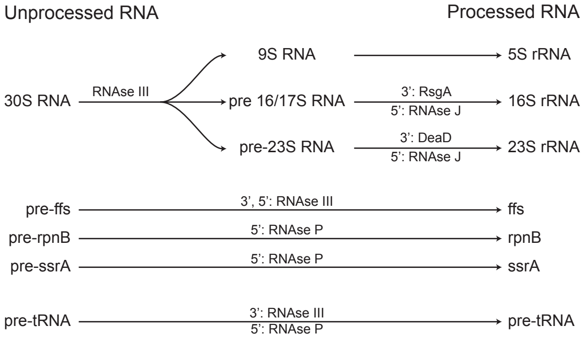
Leader Sequences
Bacteria also transcribe 3' and 5' leader sequences before and after each non-coding RNA gene. These leader sequences must be cleaved to produce functional non-coding RNA.

mRNA Cleavage
M. genitalium mRNA cleavage is not well described and is not modeled. The model assumes *M. genitalium* polycistronic mRNA are not cleaved.

rRNA Cleavage
E. coli rRNA have been shown to be transcribed as a single 30S transcript which is cleaved into individual 5S, 16S, and 23S rRNA transcripts by the action of several ribonucleases[105]. The *E. coli* rRNA cleavage scheme was adapted and simplified for the reduced ribonuclease complement of *M. genitalium* by removing cleavage reactions catalyzed by non-homologous enzymes. Schematic S14 illustrates the reconstructed *M. genitalium* rRNA cleavage scheme. First, ribonuclease III (MG367) hydrolytically cleaves the 30S rRNA into 5S, 16S, and 23S rRNA precursors. Second, hydrolytic ribonuclease J (MG139) and phosphorolytic ribonucleases RsgA (MG110) and DeaD (MG425) cleave the 3' and 5' ends of the 5S and 16S rRNA precursors. *M. genitalium* doesn't contain a homolog of the *E. coli* 9S RNA cleavage enzyme ribonuclease E. The mechanisms of 3' and 5' cleavage of the *M. genitalium* 5S rRNA precursor are unknown. 3' and 5' cleavage of the 5S rRNA precursor are modeled as a spontaneous process.

sRNA Cleavage
The reconstructed *M. genitalium* small non-coding RNA (sRNA) precursor cleavage scheme illustrated in Schematic S14 was based on that of *E. coli*[39,649]. ffs (MG0001) is cleaved at both its 3' and 5' ends by ribonuclease III. ribonuclease P cleaves the 5' end of rpnB (MG0003) and ssrA (MG0004).

tRNA Cleavage
The reconstructed *M. genitalium* tRNA precursor cleavage scheme illustrated in Schematic S14 was based on that of *E. coli*[39,649]. Ribonucleases III (MG367) and P (MG0003, MG465) hydrolytically cleave the 3' and 5' ends of each pre-tRNA, removing the 3' and 5' leader regions and intercistronic regions to produce individual tRNA.



Schematic S14. Non-coding RNA cleavage.

Cleavage Reactions
The stoichiometry, kinetics, and energetics of each non-coding RNA cleavage reaction were reconstructed based on extensive review of the primary literature (see Table S3D and S3O).

Computational Representation

Mathematical Model
This process models the cleavage of operonic non-coding RNA into individual gene products and intercistronic RNA fragments. Because the kinetics of RNA cleavage are not well characterized, this process makes several simplifying assumptions. First, this process assumes that each RNA is fully cleaved in a single time step. Consequently, this process collapses the cleavage of each RNA into a single reaction, and only represents uncleaved and fully cleaved RNA. Intermediate cleavage configurations are not represented. Second, this process assumes that the mean cleavage rate, v_i of each RNA species i is independently limited by (1) the copy number of unprocessed RNA, r_i^u, (2) the copy numbers of intracellular metabolites, m_j, and (3) the copy numbers of RNA processing enzymes, e_j. Based on these assumptions, the functional form of v_i is

given by

$$v_i = \min\left(\overbrace{r_i^u}^{\text{RNA}},\ \overbrace{\left\lfloor \min_j \frac{m_j}{M_{ji}^s} \right\rfloor}^{\text{metabolites}},\ \overbrace{\texttt{poissonRand}\left(\min_j \frac{e_j}{K_{ji}} \Delta t\right)}^{\text{enzymes}}\right), \tag{S29}$$

where M_{ji} is the stoichiometry of metabolite j in the processing of RNA species i including NTP hydrolysis coupled to phosphorolytic cleavage, $M^s = \max(0, -M)$ is the negative part of M, K_{ji} is the experimentally observed catalytic rate of enzyme j in the processing of RNA species i, and $\Delta t = 1\,\mathrm{s}$ is the simulation time step.

This process implements a stochastic model of the arrival of RNA processing events with relative rates v_i. Until RNA, metabolic, and/or enzymatic resources are exhausted, the model iteratively (1) computes the arrival rate, v_i, of each processing event, (2) selects a single processing event to execute according to a multinomial distribution parameterized by v_i, and (3) executes the selected processing reaction, updating the copy numbers of RNA and metabolites and decrementing the available enzymatic capacity. Algorithm S24 outlines the implementation of the RNA processing model.

Integration
The `Rna` state represents the copy numbers of unprocessed, processed, and intercistronic RNA. The `Metabolite` state represents the copy number of each intracellular metabolite. The `Protein Monomer` and `Protein Complex` states represent the copy number of each RNA processing enzyme.

RNA are synthesized and matured in four steps (see Section 2.12). This process models the cleavage of RNA transcripts produced by the `Transcription` process. The `RNA Modification` process models the modification of transcripts cleaved by this process. The `Macromolecular Complexation` and `Ribosome Assembly` processes model the formation of macromolecular complexes, including the 30S and 50S ribosomal particles. The `Translation` process models the function of m-, r-, s-, and tRNA in translation. The `RNA Decay` process models the degradation of cleaved intercistronic RNA fragments.

Initial Conditions
Section 1.4 outlines the cell state initialization algorithm. Briefly, after the `Mass` state initializes the total cell mass, the `Rna` state initializes the total copy number of each RNA species and initializes all RNA to their mature – processed and modified – configuration. Second, the `tRNA Aminoacylation` process initializes tRNA to the aminoacylated configuration. Next, the `Macromolecular Complexation` and `Ribosome Assembly` processes initialize the copy number of each ribonucleoprotein complex. Finally, the `Translation` process initializes the mRNA codon location of each 70S ribosome.

Dynamic Computation
Algorithm S24 outlines the implementation of the RNA processing model.

Fitting
The expression of the RNA processing enzymes was fit to provide sufficient enzymes to quickly process RNA, or more specifically to prevent sustained accumulation of unprocessed RNA. See Section 1.3 for further discussion.

Algorithm S24 | RNA processing simulation. See Mathematical Model section above for definition of the mathematical notation.

```
Input: $r_i^p$ copy number of processed RNA species $i$
Input: $r_i^i$ copy number of intercistronic fragment $i$
Input: $R_{ji}^g$ is one if operonic RNA $i$ contains gene $j$, and zero otherwise
Input: $R_{ji}^i$ is one if operonic RNA $i$ contains intercistronic fragment $j$, and zero otherwise
Let $k_i \leftarrow e_i \Delta t$ be the capacity of enzyme $i$ for RNA processing
repeat
    Calculate cleavage rates
    foreach operonic non-coding RNA species $i$ do
        Calculate $v_i$ according to Eq. S29
    Select operonic non-coding RNA species $i \sim$ multinomialRand(1, $v_i / \sum_j v_j$)
    Update RNA copy numbers: $r_i^u \leftarrow r_i^u - 1$, $r^p \leftarrow r^p + R_{\bullet i}^g$, $r^i \leftarrow r^i + R_{\bullet i}^i$
    Update metabolites: $m \leftarrow m - M_{\bullet i}$
    Update enzyme catalytic capacity: $k \leftarrow k - K_{\bullet i}$
until no further cleavage possible ($v_i = 0 \; \forall \; i$)
```

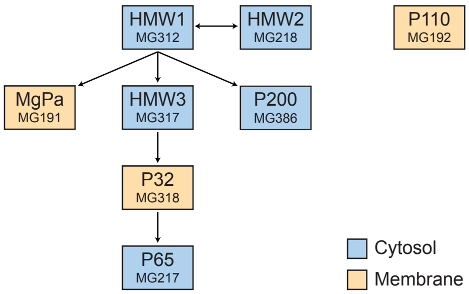
3.24 Terminal Organelle Assembly

Biology

Balish and Krause have shown that *M. genitalium* maintains a flask shape with a single 300×80 nm membrane-bound bleb or terminal attachment organelle throughout most of its life cycle[91]. Krause and Balish have also shown that the *M. genitalium* terminal organelle divides during cell division, producing a daughter organelle which subsequently migrates to the opposite pole[409]. The *M. genitalium* terminal organelle been associated with several cellular processes including motility, adhesion, replication, and cytokinesis[88,91,406–409,794,803,813]. This process models the assembly of the protein content of the terminal organelle.

Reconstruction

Krause and Balish have shown that the terminal organelle is composed of eight proteins – high molecular weight cytadherence accessory proteins (HMW) 1-3 (MG312, MG218, MG317), adhesins MgPa (MG191), P32 (MG318) and P65 (MG217), and proteins P110 (MG192) and P200 (MG386)[406–409]. The terminal organelle protein content is believed to assemble in the stereotyped hierarchical pattern illustrated in Schematic S15[408,409]. First, HMW1 and HMW2 mutually recruit each other into the terminal organelle. Second, HMW1 recruits MgPa, HMW3, and P200 into the terminal organelle. Third, HMW3 recruits P32 which in turn recruits P65. P110 independently localizes to the terminal organelle. The kinetics of terminal organelle assembly are unknown.



Schematic S15. Hierarchical assembly of the *M. genitalium* terminal organelle.

Computational Representation

Mathematical Model

This process implements a Boolean model of the observed hierarchical assembly of the protein content of the *M. genitalium* terminal organelle. Schematic S15 outlines the Boolean assembly model. Algorithm S25 describes the model implementation in detail.

Integration

The `Protein Monomer` state represents the copy number of each protein species in each of four compartments: cytosol (c), membrane (m), and terminal organelle cytosol (tc) and membrane (tm). The `Host` state represents the status of the host urogenital epithelium to which the *M. genitalium* terminal organelle attaches.

Several processes including `Translation` model the synthesis and maturation of protein monomers (see Section 2.10). The `Host Interaction` process models the role of the terminal organelle in host attachment and immune activation.

Initial Conditions

After the total cell mass is initialized, the `Protein Monomer` state initializes the terminal organelle copy number of each terminal organelle protein.

Dynamic Computation

Algorithm S25 outlines the implementation of the Boolean terminal organelle assembly model.

Algorithm S25 | Terminal organelle assembly simulation.

```
if HMW1 and HMW2 expressed ($HMW1_c + HMW1_{tc} > 0$ and $HMW2_c + HMW1_{tc} > 0$) then
    Localize HMW1 and HMW2 to the terminal organelle
    $HMW1_{tc} \leftarrow HMW1_{tc} + HMW1_c$
    $HMW2_{tc} \leftarrow HMW2_{tc} + HMW2_c$
    $HMW1_c \leftarrow 0$
    $HMW2_c \leftarrow 0$
if HMW1 localized to terminal organelle ($HMW1_{tc} > 0$) then
    Localize HMW3, MgPa, P200 to terminal organelle
    $HMW3_{tc} \leftarrow HMW3_{tc} + HMW3_c$
    $MgPa_{tm} \leftarrow MgPa_{tm} + MgPa_m$
    $P200_{tc} \leftarrow P200_{tc} + P200_c$
    $HMW3_c \leftarrow 0$
    $MgPa_m \leftarrow 0$
    $P200_c \leftarrow 0$
if HMW3 localized to terminal organelle ($HMW3_{tc} > 0$) then
    Localize P32 to terminal organelle
    $P32_{tm} \leftarrow P32_{tm} + P32_m$
    $P32_m \leftarrow 0$
if P32 localized to terminal organelle ($P32_{tm} > 0$) then
    Localize P65 to terminal organelle
    $P65_{tc} \leftarrow P65_{tc} + P65_c$
    $P65_c \leftarrow 0$
Localize P110 to terminal organelle
$P110_{tm} \leftarrow P110_{tm} + P110_m$
$P110_m \leftarrow 0$
```

3.25 Transcription

Biology

Transcription is the first step in the synthesis of functional gene products where RNA polymerase and several accessory enzymes translate transcription units (regions of the DNA containing 1 or more genes) into RNA molecules. An RNA polymerase can bind on and off of the DNA either specifically to a gene promoter or non-specifically to a non-promoter site[775]. Transcription begins with the recruitment of RNA polymerase to a gene promoter with the help of a sigma initiation factor and possibly transcription factors. Next elongation factors are recruited, RNA begins to be polymerized, and the sigma factor is released. In this stage, the RNA polymerase is said to be in the actively transcribing state. Finally the RNA polymerase reaches a terminator at the end of the transcription unit, and with the help of termination factors releases the polymerized RNA and dissociates from the DNA.

Termination of transcription in some bacteria can require the hexameric ATP-dependent helicase Rho[233]. However, Rho is not essential in *B. subtilis* (gram positive) or *M. genitalium*[234], and therefore we chose to only include Rho-independent termination in our model. Rho-independent termination occurs via the intrinsic properties of RNA which disrupt RNA polymerase-DNA binding.

As soon as the RNA begins to polymerize, even prior to termination, the mRNA transcripts may be bound by ribosomes and translated to polypeptides. For simplicity, our model doesn't represent this phenomenon, allowing translation only of completed mRNAs.

Reconstruction

Enzymes
All of the enzymes required for transcription are detailed in List S22.

List S22. Enzymes and complexes used in the Transcription process class.

Enzymes/Complexes	Composition	Gene Name(s)	DNA (bases)	Footprint
RNA polymerase sigma factor	(1) MG249	rpoD	58	
Transcription elongation factor	(1) MG282	greA		
Transcription termination factor	(1) MG141	nusA		
Transcription termination/antitermination protein	(1) MG027	nusB		
DNA-directed RNA Polymerase	(1) MG022, (2) MG177, (1) MG340, (1) MG341	rpoE, rpoA, rpoC, rpoB	75	

Parameters
The parameters required for transcription were derived from multiples sources including mircoarray gene expression data and RNA half life data. The data used and calculations performed pre-fitting, are described below:

mRNA Gene expression data
Gene expression data for *M. genitalium* was unavailable, so we used data (measured at 37°C) from *Mycoplasma pneumonie*, the closest phylogenetic relative of *M. genitalium*[569]. The microarray data by Weiner III *et al.* was presented in normalized log form. Since the exact method of normalization was unknown to us, to re-derive expression levels we simply calculated 2^(presented value). The *M. genitalium* genome is contained in the larger *M. pneumonie* genome and we were able to map a *M. pneumonia* gene to all but 6 *M. genitalium* genes[576]. The genes were mapped based on matching gene names, matching gene descriptions, and Bio-cyc's Align Multi-Genome Browser[6].

tRNA expression data
We also require tRNA expression estimates analogous to the mRNA microarray data. Since no microarray data is available, we use cell composition data to approximate tRNA expression[870]. We approximate the

total tRNA expression as the total mRNA expression multiplied by the ratio of the tRNA to mRNA weight fractions of the cell. Next we need to approximate the expression of each individual tRNA. We use measurements of the relative abundances of each amino acid in the cell. A tRNA's expression is the total tRNA expression multiplied by the fraction of the total amino acid weight represented by its amino acid[53,182]. For example, alanine accounts for about 8% of the total amino acid mass. Then the expression of tRNA for alanine, MG471, is 8% of the total tRNA expression. In cases of degeneracy, where multiple tRNAs bind to the same amino acid, the amino acid weight fractions is split evenly between the tRNAs.

rRNA and sRNA expression data

Similar to the tRNA expression calculations, the total rRNA expression is calculated as the total mRNA expression multiplied by the ratio of the rRNA to mRNA weight fractions of the cell. This total rRNA expression is split between the three rRNA species (23S, 16S, 5S) based on their relative abundances in the cell. sRNA expression is also calculated in the same way, such that expression is proportional to RNA abundance in the cell.

Half-life data

We obtained mRNA half-lives from measurements in *E. coli* performed at 30°C in M9 minimal media[602]. Additional sets of *E. coli* half-life data are available, but we chose the set with the most comprehensive list of genes mapping to homologous *M. genitalium* genes, and the set whose average half-life was closest to the reported bulk *E. coli* mRNA half-life[601,603]. The gene mapping between *E. coli* and *M. genitalium* was based on common gene names and annotations, Bio-Cyc's Align Multi-Genome Browser, and UniProt[6,96,182]. This have us a half-life estimate for 274 *M. genitalium* genes. For the remaining genes, we assigned the average half-life, 4.425 minutes. We used separate sources to acquire the half-lives of tRNAs (45 minutes), rRNAs (150 minutes), and sRNAs (89 minutes)[604,608].

Gene assignment to transcription units

The transcription unit structure (sets of genes that are transcribed together following a single promoter binding event) was compiled from several sources including primary reports[176,186,188,244,247–249,411] and databases[182,250] of cotranscribed genes, and a computational model[251]that predicts promoter and transcription unit start sites. We also predicted transcription units using information on the conservation of gene order across multiple species, the related functions of adjacent genes, the similar expression levels of adjacent genes as measured by microarrays, and the strandedness (transcription direction) of adjacent genes[569]. We transcribe the 525 *M. genitalium* genes in 355 transcription units. 294 genes fall in transcription units of more than one gene. The longest transcription unit contains 4 genes.

Rate at which each transcript is produced

The rate of change of the concentration of an RNA_i, is its synthesis net its degradation:

$$\frac{d[RNA_i]}{dt} = \underbrace{k_{synth,i}}_{\text{synthesis}} - \underbrace{k_{deg,i}\,[RNA_i]}_{\text{degradation}}, \tag{S30}$$

where $k_{synth,i}$ is the rate of synthesis and $k_{deg,i}$ is the rate of degradation of RNA_i

The degradation rate is described by the first-order degradation constant of RNA_i with half-life, h_i (obtained as described above).

$$k_{deg,i} = \frac{\ln(2)}{h_i} \tag{S31}$$

At steady state RNA concentration, $\frac{d[RNA]}{dt} = 0$:

$$k_{synth,i} = \frac{\ln(2)}{h_i}\,[RNA_i]_{SS} \tag{S32}$$

The relative expression, E_i, as described above, was obtained from microarray and cell composition data.

Substituting $[RNA_i]_{SS}$ with $E_i[RNA]$, where the relative expression, E_i, as described above, was obtained from microarray and cell composition data, and $[RNA]$ is the total RNA concentration in the cell, we get the synthesis rate of RNA_i:

$$k_{synth,i} = \frac{\ln(2)}{h_i} E_i \, [RNA] \tag{S33}$$

Rate at which each transcript is produced (assuming exponential growth)
We adjust the above derivation for growth and dilution. Assuming exponential growth of the cell:

$$k_{synth,i} = k_{synth,iinit} e^{\ln(2)\frac{t}{\tau}} \tag{S34}$$

$$[RNA_i] = [RNA_i]_{init} e^{\ln(2)\frac{t}{\tau}} \tag{S35}$$

where τ is the cell cycle length,

$$\frac{d[RNA_i]}{dt} = \underbrace{k_{synth,iinit} e^{\ln(2)\frac{t}{\tau}}}_{\text{degradation}} - \underbrace{k_{deg,i}[RNA_i]_{init} e^{\ln(2)\frac{t}{\tau}}}_{\text{degradation}} \tag{S36}$$

Assuming that mother and daughter cells are identically distributed:

$$[RNA_i]_{t=\tau} = 2[RNA_i]_{init} \tag{S37}$$

$$[RNA_i]_{init} = \int_0^\tau \frac{d[RNA_i]}{dt} \tag{S38}$$

$$= (k_{synth,iinit} - k_{deg,i}[RNA_i]_{init}) \frac{\ln(2)}{\tau} \tag{S39}$$

$$k_{synth,iinit} = [RNA_i]_{init} \left(\frac{\tau}{\ln(2)} + k_{deg,i} \right) \tag{S40}$$

Again substituting $[RNA_i]$ with $E_i[RNA]$:

$$k_{synth,iinit} = E_i[RNA]_{init} \left(\frac{\tau}{\ln(2)} + k_{deg,i} \right) \tag{S41}$$

Synthesis Rate Adjustment for Transcription Units
Genes that are transcribed together in a transcription unit must have the same synthesis rate. This is because we do not model incomplete transcription of a transcription unit, where the RNA polymerase can fall off the unit, after transcribing a subset of its genes. For half-lives that have not been measured in *E. coli* and were set to an average 4.425 minutes, we adjusted the half-lives to even out the synthesis rates within a transcription unit. For the other cases, we set the synthesis rate of the transcription unit to be the average of the rates within the transcription unit.

Determining the probabilities of making transcripts
The probability of an RNA polymerase binding a given transcription unit, i, is based on the synthesis rate of transcription unit i relative to the synthesis rate of all the other transcription units:

$$\begin{aligned}\text{Probability}(\text{transcribing } i) &= P_{tu,i} && \text{(S42)}\\ &= \frac{k_{synth,i}}{\sum_{j=1:335} k_{synth,ij}} && \text{(S43)}\end{aligned}$$

Substituting in the synthesis rate from above:

$$\begin{aligned}P_{tu,i} &= \frac{E_i[RNA]_{init}(\frac{\tau}{ln(2)} + k_{deg,i})}{\sum_{j=1:335} E_i[RNA]_{init}(\frac{\tau}{\ln(2)} + k_{deg,i})} && \text{(S44)}\\ &= \frac{E_i\left(\frac{\tau}{ln(2)} + k_{deg,i}\right)}{\sum_{j=1:335} E_i(\frac{\tau}{\ln(2)} + k_{deg,i})} && \text{(S45)}\end{aligned}$$

Parameter Fitting

After and all of the raw data has been transformed according to the calculations described above, and fit to assure growth of a wildtype cell that will accommodate doubling in 9hours, we reach the parameters described in List S23.

List S23. Fixed parameters used in the `Transcription` process class.

Parameters	Value	Symbol	Source
RNA polymerase elongation rate	50 nt s^{-1}	k_{el}	[562, 563]
Transcription unit binding probabilities		P_{tu}	See 'Data and Calculations'
RNA polymerase state transition probabilities		P_{trans}	[775]
Enzyme DNA footprints	See 'Enzymes/complexes'		[747, 748]
Transcription unit sequences	See Table S3K, [182]		[182]
Transcription unit directions	See Table S3K		[182]
Transcription unit lengths	See Table S3K		[182]
Transcription unit 5' coordinates	See Table S3K		[182]
Cell Cycle Length	8.9 h	τ	Experimentally measured. See Experimental Methods.

Computational Representation

We use a Markov model to determine the state of each RNA polymerase. An RNA polymerase may exist in any of the four following states:

1. Free (not bound to a chromosome) (FS)
2. Non-specifically bound somewhere on a chromosome (NSB)
3. Specifically bound to the promoter of a transcription unit (SB)
4. Actively transcribing a transcription unit (AT)

Transition probabilities between these states are designed to maintain the occupancy of each state within a narrow window around their expected values[775]. The transition probabilities (P_{trans}) are determined by four logistic control functions, tuned by the RNA polymerase state expectations.

Free State

All newly created or released RNA polymerases are in the free state. From the free state, a polymerase can transition to the non-specifically bound state, transition to the specifically bound state, or remain in the free state.

Non-specifically bound state

From the non-specifically bound state, a polymerase can transition to the free state, transition to the specifically bound state, or remain in the non-specifically bound state. A random position on a chromosome for the non-specifically bound polymerase is selected from the polymerase accessible sites as determined by the Chromosome state.

Specifically bound state

From the specifically bound state, a polymerase can transition to the free state, non-specifically bound state, or actively transcribing state, or remain in the specifically bound state. A polymerase can only transition into the specifically bound state if a free sigma factor is available. Upon transition a polymerase into this state, we randomly pick a transcription unit to bind to. The probability of binding each transcription unit is based on experimentally measured gene expression and half life data, as detailed below.

Actively transcribing state

Once in the actively transcribing state, a polymerase will remain in the actively transcribing state until transcription is terminated, the RNA polymerase is displaced by another protein, or the RNA polymerase falls off due to a stall. In all these cases, the RNA polymerase falls off into the free state. For a polymerase in the actively transcribing state, we release any bound sigma factors and elongate the transcript according to nucleic acid limits (given the transcript sequence) and the elongation rate k_{el}. The transcript and polymerase are released if transcription is complete and the necessary termination factors are available.

Integration

List S24. State classes connected to the `Transcription` process class.

Connected States	Read from state	Written to state
`Chromosome`	▪ Regions accessible for transcription machinery to bind ▪ Polymerized regions of DNA	▪ Positions and strands on the DNA where transcription machinery is bound
`Rna`	▪ Counts of RNA species	▪ Updated couents of RNA species
`RNA Polymerase`	▪ RNA polymerase states (FS, NSB, SB, AT) ▪ DNA positions and strands of DNA bound RNA polymerases ▪ Expectations of RNA polymerases in FS, NSB, SB, and AT states (used to derive transition Probabilities P_{trans})	▪ Updated states of RNA polymerases ▪ Updated positions and strands of DNA bound RNA polymerases
`Transcript`	▪ Transcription unit indexes of bound RNA polymerases ▪ RNA polymerase position within transcript ▪ Chromosome on which RNA polymerase resides (1 or 2) ▪ Transcription unit sequences, directions, lengths, and 5' start coordinates	▪ Updated transcription unit indexes of bound RNA polymerases ▪ Updated RNA polymerase position

Initial Conditions

All RNAs are initialized to the mature state. RNA polymerases are initialized as follows:

1. Each RNA polymerase is randomly assigned (with replacement) to one of the actively transcribing, specifically bound, non-specifically bound, or free states weighted by the expected occupancy of each state
2. Actively transcribing and specifically bound polymerases are randomly assigned to transcription units weighted by the transcription unit binding probabilities (P_{tu}).
3. Each transcription unit to which one or more actively transcribing polymerases have been assigned is divided into 1 segment for each polymerase
4. Actively transcribing polymerases are randomly assigned to positions within the assigned segment of their assigned transcription unit (positions near the segment border are not allowed to prevent polymerases from being too close to each other) with uniform probability.

5. Non-specifically bound polymerases are randomly assigned to an accessible region on the chromosome.

Dynamic Computation
At each timestep, we follow the following algorithm:

1. Assign all newly synthesized RNA polymerases to the free state
2. Use the Markov model to randomly transition RNA polymerases among the FS, NSB, SB, and AT states, weighted by the state transition probabilities P_{trans}.
3. Randomly assign RNA polymerases entering the NSB state to an accessible position on the Chromosome (performed by Chromosome state).
4. Randomly assign RNA polymerases entering the SB state to specific transcription units weighted by the product of the transcription unit binding probabilities (P_{tu}) and the binding probability fold changes. (Fold changes arise from other cellular processes and are described in the `Transcriptional Regulation` and `DNA Supercoiling` process classes.) We determine whether the selected promoter site is accessible and bind the polymerase to a chromosome using the `Chromosome` state class. RNA polymerase specific binding can only occur if there is an available sigma factor, and a sigma factor is accordingly decremented.
5. Simulate RNA polymerization by actively transcribing RNA polymerases with the aid of elongation factors.
   (a) Release the sigma factor if this is the first second of elongation, and increment the free sigma factor count
   (b) If all of the necessary elongation factors are available, elongate the transcript according to nucleic acid limits (given the transcript sequence) and the elongation rate k_{el}. We allocate available nucleic acids among the actively polymerizing RNA polymerases, by adding a base to each elongating transcript before moving to the next base of a given transcript. Polymerization can also be limited if the new site at which the polymerase will land is not accessible (for example, if another RNA polymerase lies in its way)
6. If transcription is complete and the necessary termination factors are available,
   (a) Release the completed transcript
   (b) Transition the RNA polymerase to the free state
   (c) Increment the RNA count

3.26 Transcriptional Regulation

Biology

Transcription factors are one of many mechanisms cells employ to respond to external signals and maintain homeostasis. Transcription factors regulate the synthesis rates of RNA by modulating the affinity of RNA polymerase for promoters. Transcriptional enhancers stabilize RNA polymerase-promoter complexes by contributing negative free energy to the complex, for example by providing additional surfaces for RNA polymerase binding. Transcriptional repressors destabilize RNA polymerase-promoter complexes, for example by sterically blocking promoters. This process models the binding of transcriptional regulators to promoters and the fold-change effect of transcriptional regulators on the affinity of RNA polymerase for individual promoters.

Reconstruction

The *M. genitalium* transcriptional regulatory network was reconstructed based on an extensive review of the primary literature[110,112,186,196,395,411,418,420,433–438,505] and the proteomic database DBTBS[419]. *M. genitalium* has few homologs to reported transcription factors. The reconstructed network contains five regulators which regulate 54 genes through 29 regulatory interactions, including one regulator (Spx, MG127) which interacts directly with RNA polymerase. Table S3P lists the five reconstructed transcriptional regulators and the binding site, motif, and affinity, and fold-change effect of each regulatory interaction.

Computational Representation

Mathematical Model

This process models the binding of transcriptional regulators to promoters and the fold-change effect of transcriptional regulators on RNA polymerase-promoter binding. Because *M. genitalium* transcriptional regulation is not well characterized, this process makes several simplifying assumptions. First, this process assumes that transcriptional regulator-DNA binding is kinetically fast and energetically favorable, and therefore proceeds to completion within the 1 s simulation time step. Second, because transcriptional regulator-promoter affinities have not been systematically characterized, this process assumes that transcriptional regulators bind promoters with affinity proportional to their fold change effect. Third, this process assumes that only one copy of each transcriptional regulator can bind at each promoter. Fourth, this process assumes that transcriptional regulators stably bind DNA. Consequently, transcriptional regulator dissociation is ignored except displacement by other DNA-binding proteins which is modeled by the `Transcription` and `Replication` processes. Finally, this process assumes that transcriptional regulators independently affect RNA polymerase, and thus their fold change effects add multiplicatively. Algorithm S26 outlines the implementation of the transcriptional regulation model.

Integration

The `Protein Monomer` and `Protein Complex` states represent the free and DNA-bound copy numbers of each transcriptional regulator. The `Chromosome` state represents the exact chromosomal location of each DNA-bound transcriptional regulator. The `RNA Polymerase` state represents the fold change effect of transcriptional regulation on the affinity of RNA polymerase for each promoter.

Several processes including `Translation` model protein synthesis (see Section 2.10). The `Transcription` process models (1) transcription initiation: RNA polymerase-promoter binding including the fold-change effects of transcriptional regulation, (2) transcript elongation, and 3) transcription termination.

Initial Conditions

Section 1.4 outlines the cell state initialization algorithm. Briefly, after the `Protein Monomer` and `Protein Complex` states initialize the total copy number of each transcriptional regulator, the `Transcriptional Regulation` process initializes the status – free or DNA-bound – and chromosomal location of the transcriptional regulators to a steady-state of the transcriptional regulatory network by iteratively evaluating Algorithm S26 until convergence. Because the transcriptional regulatory model assumes that transcriptional regulator-promoter binding proceeds to completion within the 1 s simulation time step and is stable, Algorithm S26 converges in one iteration.

Dynamic Computation

Algorithm S26 outlines the implementation of the `Transcriptional Regulation` model.

Algorithm S26 | Transcriptional regulation simulation.

```
Input: $n_{ik}$ is true if promoter $i$ is expressed in chromosome $k$
Input: $p_{c,i}$ free cytosolic copy number of transcriptional regulator $i$
Input: $p_{b,i}$ DNA-bound copy number of transcriptional regulator $i$
Input: $x_{ij}$ Binding site of transcriptional regulator $i$ at promoter $j$
Input: $F_{ij}$ fold-change effect of transcriptional regulator $i$ on promoter $j$
Input: $b^m_{ijkl}$, $b^c_{ijkl}$ chromosomal protein occupancy as defined in List S2
Output: $f_i$ fold-change effect of transcriptional regulation on RNA polymerase affinity for promoter $i$

Calculate the relative rate, $r_{ijk}$, transcriptional regulator $i$ binds promoter $j$ of chromosome $k$:
foreach DNA-binding transcriptional regulator $i$ in promoter $j$ of chromosome $k = \{1..2\}$ do
    $r_{ijk} \leftarrow n_{jk} p_{c,i} F_{ij}$

Bind transcriptional regulators to the chromosome:
repeat
    Select regulator $i$, promoter $j$, and chromosome $k \sim$ multinomialRand(1, $r_{ijk} / \sum_{ijk} r_{ijk}$)
    if regulator $i$ expressed ($p_{c,i} > 0$) and isRegionAccessible(promoter $j$ of chromosome $k$ to regulator $i$) then
        Bind protein to chromosome: $b^z_{y \bullet ki} = 1 \; \forall \; y \in \{x_{ij}..x_{ij} + l_i - 1\}$, where $z = m$ for monomers and $c$ for complexes
        Update free and bound copy numbers: $p_{c,i} \leftarrow p_{c,i} - 1$, $p_{b,i} \leftarrow p_{b,i} + 1$
    Update binding rate: $r_{ijk} \leftarrow 0$
until no additional transcriptional regulator can bind DNA ($r_{ij} = 0 \; \forall \; i, \; j$)

Calculate the fold-change effect of transcriptional regulators on the affinity of RNA polymerase for each promoter
Initialize fold-change effects: $f_i \leftarrow 1 \; \forall \; i$
foreach promoter $j$ of chromosome $k = \{1..2\}$ bound by DNA-binding transcriptional regulator $i$ do
    Add fold-change effects multiplicatively: $f_j \leftarrow f_j F_{ij}$

foreach promoter $j$ regulated by an expressed non-DNA-binding transcriptional regulator $i$ ($p_{c,i} > 0$) do
    $f_j \leftarrow f_j F_{ij}$
```

3.27 Translation

Biology

Translation is the process whereby the ribosome, accessory enzymes, and tRNAs transcode mRNAs and produce amino acid polymers. Translation begins with the recruitment of the 30S and 50S ribosomal particles and initiation factor 3 (IF3) to an mRNA molecule. Next, the ribosomal particles assemble into a 70S ribosome on the mRNA molecule with the help of initiation factors. Third, the ribosome polymerizes amino acids presented by aminoacylated tRNAs with the help of elongation factors. Finally, a release factor recognizes the stop codon UAG or UAA, hydrolyzes the peptidyl tRNA bond, and dissociates. A ribosome recycling factor dissociates the E-site tRNA, an elongation factor G releases the release factor and 50S ribosome, and an initiation factor 3 dissociates the 30S ribosome, P-site tRNA, and mRNA[393].

A ribosome may stall for various reasons including collisions with other proteins and limited resources required of translation. When a ribosome stalls, its last tRNA is expelled and replaced by the tRNA-like domain of a tmRNA molecule. Next, the mRNA-like domain of the tmRNA expels the bound mRNA. Third, the ribosome resumes polymerization, now using the tmRNA's mRNA-like domain as its template. This results in the production of an amino acid polymer containing a C-terminal proteolysis tag. Finally, the proteolysis tag will be recognized by the protein degradation machinery, and the amino acid polymer will be degraded into its individual component amino acids[226].

Reconstruction

All of the enzymes required for translation are detailed in List S25, and all of the parameters are detailed in List S26.

List S25. Enzymes and complexes used in the `Translation` process class.

Enzymes/Complexes	Composition	Gene Name(s)
Initiation factor IF-1	(1) MG173	infA
Initiation factor IF-2	(1) MG142	infB
Initiation factor IF-3	(1) MG196	infC
Elongation factor G	(2) MG089	fusA
Elongation factor P	(1) MG026	efp
Elongation factor Tu	(2) MG451	tuf
Elongation factor Ts	(2) MG433	tsf
Peptide chain release factor 1	(1) MG258	prfA
Ribosome recycling factor	(1) MG435	frr
30S ribosomal subunit	(1 each) MGrrna16S, MG070, MG087, MG088, MG090, MG092, MG150, MG155, MG157, MG160, MG164, MG165, MG168, MG175, MG176, MG311, MG417, MG424, MG446, MG481, MG522	16S rRNA, rpsB, rpsL, rpsG, -, rpsR, rpsJ, rpsS, rpsC, rpsQ, rpsN, rpsH, rpsE, rpsM, rpsK, rpsD,, rpsI, , rpsO, , rpsP, -, -
30S ribosomal subunit - initiation factor IF-3 complex	(1) Ribosome_30S, (1) MG196	30S ribosomal subunit, infC
50S ribosomal subunit	(1 each) MGrrna23S, MGrrna5S, MG081, MG082, MG093, MG151, MG152, MG153, MG154, MG156, MG158, MG159, MG161, MG162, MG163, MG166, MG1676, MG169, MG174, MG178, MG197, MG198, MG257, MG325,MG361, MG362, MG363, MG418, MG426, MG444, MG466, MG473	23S rRNA, 5S rRNA, rplK, rplA, -, rplC, rplD, rplW, rplB, rplV, rplP, rplC, rplN, rplX, rplE, rplF, rplR, rplO, rpmJ,, rplQ, rpmI, rplT, rpmE, rpmG, -, rplL, rpmF, rplM, rpmB, rplS, rpL34, rpmG-2
70S ribosome	(1) Ribosome_50S, (1) Ribosome_30S	50S ribosomal subunit, 30S ribosomal subunit
tmRNA, MCS6 (10sa RNA)	(1) MG0004	ssrA
SsrA binding protein	(1) MG059	smpB
peptidyl-tRNA hydrolase	(1) MG083	pth

List S26. Fixed parameters used in the `Translation` process class.

Parameter	Value	Symbol	Source
Ribosome elongation rate (amino acids/second)	16	k_{elong}	[564]
Probability that stationary ribosome is moved to a stalled state (mRNA replaced by tmRNA)	1×10^{-6}	$P_{stalled}$	See "Parameter Fitting"
tRNA sequences of the protein monomers			Determined from genome sequence, tRNA code [247]
tRNA sequences of the proteolysis tags			Determined from genome sequence, tRNA code [247]

Parameter Assignment

Most of the parameters required for this process class were obtained from the literature or derived from the *M. genitalium* genome sequence. The probability that a ribosome stalled is an uncharacterized value, and therefore we set this probability to be very low, such that ribosome stalling is a rare event which does not typically limit the simulation.

Computational Representation

This process simulates protein translation by ribosomes and accessory initiation, elongation, and termination factors. Ribosomes transition from a free to active state if the necessary factors are present, and bind to an mRNA. The selection of a specific mRNA to bind to a ribosome is random and weighted by mRNA abundances. As the sequence of each gene is known, the codons presented by the mRNAs are translated using aminoacylated tRNAs and elongation factors at a rate of up to 16 amino acids per second (measured by radioactive labeling in *E. coli*)[564]. Once a stop codon has been reached, translation is terminated and the polypeptide will undergo further modifications by other processes in our simulation. The process class also simulates the identification of stalled ribosomes by the tmRNA, the replacement of the tRNA and mRNA with the tmRNA, and the synthesis of the proteolysis tag encoded by the tmRNA's mRNA-like domain.

Integration

List S27. State classes connected to the `Translation` process class.

Connected State	Read from state	Written to state
`Polypeptide`	▪ List of ribosome-bound mRNAs ▪ Lengths of nascent polypeptides (proteolysis tags) ▪ tRNA sequences of protein monomers and proteolysis tags	▪ Updated list of ribosome-bound mRNAs ▪ Updated lengths of nascent polypeptides (proteolysis tags)
`Ribosome`	▪ List of ribosome-bound mRNAs ▪ Lengths of nascent polypeptides (proteolysis tags) ▪ State of each ribosome: free, actively translating, or stalled	▪ Updated list of ribosome-bound mRNAs ▪ Updated lengths of nascent polypeptides (proteolysis tags) ▪ Updated state of each ribosome: free, actively translating, or stalled
`Rna`	▪ Counts of mRNA, aminoacylated tRNA, and aminoacylated tmRNA species	▪ Updated counts of aminoacylated tRNA and aminoacylated tmRNA species

Initial Conditions
The simulation begins in a state in which ribosomes are already bound to mRNAs and in the process of elongating. Each ribosome is randomly assigned (without replacement) to an mRNA species, weighted by the current expression of the mRNAs. Then, each ribosome is assigned to positions within the assigned mRNA with uniform probability. No ribosomes are initialized to the stalled state, since the expected probability of stalling is negligible. No tmRNAs are initiated to the bound state.

Dynamic Computation
Initiation
Each ribosome exists in one of two states, free or actively transcribing. Ribosomes are created in the free state and may transition to the actively transcribing state as follows:

- If ribosome factor A is present, initiation factors (IF-1, IF-2, and IF-3) are present, and one unit of energy (GTP) is available
  - Randomly select free ribosomes to initiate up to the limits of ribosome factor A, initiation factors, and GTP
  - Randomly select mRNA species for each initiating ribosome to bind to, weighted by the counts of each mRNA species
  - Update the state of each ribosome and mRNA, and the amount of available substrates

Elongation
Next, we elongate polypeptides. We assume that one of each elongation factor (EF-tu, TS, and G) is sufficient for each ribosome, but require a separate set of factors for each ribosome. Elongation proceeds as follows:

- If elongation factors (EF-tu, TS, and G), aminoacylated tRNAs, and energy (GTP) are available,
  - Randomly select actively transcribing ribosomes to elongate up to the limits of elongation factors. Available amino acids and energy are allocated among actively translating ribosomes.
    - If the ribosome is newly initiated,
      - Release the initiation factors
      - Associate a tRNA for f-methionine to bind the first amino acid, and release a free tRNA
    - Else,

        - Derive the amino acid sequence of the translating gene by converting the genome sequence into a tRNA sequence using the amino acid code
        - Associate aminoacylated tRNAs to bind amino acids to the growing polypeptide up to the aminoacylated tRNA limit, energy limit, and elongation limit, k_{elong}. Release free tRNAs
    - Update the state of each ribosome, the progress of all actively translating ribosomes, and the amount of available substrates

    Note, for cases in which translation of a peptide finishes partway through the time step, the elongation factors are released, but only available at the timestep. For simplicity, we do not explicitly model the transition between P an E sites.

Termination

Once all amino acids of a protein have been translated, one release factor, one recycling factor, one elongation factor G, and one GTP molecule are sufficient to terminate the polypeptide, as follows:

- If at release factors, recycling factors, elongation factors G, and energy (GTP) are available,
    - Randomly select completed polypeptide-mRNA-ribosome complexes to dissociate
    - Update the state of each ribosome and mRNA, monomer counts, and the amount of available substrates. The ribosome will be available to bind mRNA at the following iteration

Translation Stalling

The stalling of ribosomes is dealt with as follows:

- If in a timestep an elongating ribosome hasn't advanced,
    - Then with a small probability, $P_{stalled}$,
        - Transition the ribosome to the stalled state
        - Expel the mRNA and replace it with a tmRNA
        - Encode a proteolysis tag and mark the amino acid chain for degradation by the `Protein Decay` process class.

3.28 tRNA Aminoacylation

Biology

The `tRNA Aminoacylation` process class simulates the conjugation of amino acids to the tRNAs. tRNAs serve as mediators between the ribosome and the amino acids which form polypeptides. tRNAs are composed of short RNA sequences which recognize specific codons (triplets of bases) on mRNAs. Each tRNA binds to a specific amino acid, and then interacts with a ribosome to deliver this amino acid to an elongating polypeptide chain, according to the mRNA code.

tmRNAs are short RNA structures that add a proteolysis tag to the end of incomplete polypeptides upon ribosomal stalling, signaling the polypeptides for degradation. The `tRNA Aminoacylation` process class also simulates the aminoacylation of the tmRNA which delivers the amino acid alanine to stalled ribosomes.

Reconstruction

These aminoacylation reactions are both enzyme and energy dependent. *M.genitalium* is believed to have 36 tRNA aminoacylation reactions and 1 tmRNA aminoacylation reaction, and all of the reactions require 1 ATP[53,100]. There are also two tRNA modification reactions that must occur. Since there is no glutaminyl-tRNA synthetase (to add a glutamine to a tRNA), glutamyl-tRNA synthetase first adds a glutamate to tRNA MG502, and then Glu-tRNA Gln amidotransferase converts the glutamate into a glutamine. Also, a methionylformyltransferase is used to add the formyl group onto the methionine on tRNA MG488 (formylmethionine is used as the start codon).

All of the enzymes required for tRNA aminoacylation are detailed in List S28, and all of the parameters are detailed in List S29.

List S28. Enzymes and complexes used in the `tRNA Aminoacylation` process class.

Enzymes/Complexes	Composition	Gene Name(s)
Alanyl-tRNA synthetase	(4) MG292	alaS
Arginyl-tRNA synthetase	(1) MG378	argS
Aspartyl-tRNA synthetase	(2) MG036	aspS
Asparaginyl-tRNA synthetase	(2) MG113	asnS
Cysteinyl-tRNA synthetase	(1) MG253	cysS
Glutamyl-tRNA synthetase	(1) MG462	gltX
Glycyl-tRNA synthetase	(2) MG251	glyS
Histidyl-tRNA synthetase	(2) MG035	hisS
Isoleucyl-tRNA synthetase	(1) MG345	ileS
Leucyl-tRNA synthetase	(1) MG266	leuS
Lysyl-tRNA synthetase	(2) MG136	lysS
Methionyl-tRNA synthetase	(2) MG021	metG
Phenylalanyl-tRNA synthetase	(2) MG194, (2) MG195	pheS, -
Prolyl-tRNA synthetase	(2) MG283	proS
Seryl-tRNA synthetase	(2) MG005	serS
Threonyl-tRNA synthetase	(2) MG375	thrS
Tryptophanyl-tRNA synthetase	(2) MG126	trpS
Tyrosyl-tRNA synthetase	(2) MG455	tyrS
Valyl-tRNA synthetase	(1) MG334	valS
Glutamyl-tRNA(Gln) amidotransferase	(1) MG098, (1) MG099, (1) MG100	-, -, gatB
Methionyl-tRNA formyltransferase	(1) MG365	

List S29. Fixed parameters used in the `tRNA Aminoacylation` process class.

Parameters	Source
Free metabolites required for each reaction	[53, 100]
tRNA aminoacylated by each reaction	[53, 100]
Enzymes required to aminoacylate each tRNA	[100]
k_{cat} of the catalyzing enzyme of each aminoacylation reaction (s^{-1})	[100]

Computational Representation

The `tRNA Aminoacylation` process maximizes the number of aminoacylation reactions (tRNA aminoacylations, tmRNA aminoacylations, and tRNA modifications) up to the limits of available RNAs, enzymes, and metabolites. The enzymatic bounds are calculated from the catalytic turnover constant (k_{cat}) of each enzyme. The required metabolites include, among others, amino acids and ATP. Since multiple reactions require the same metabolites and enzymes, reactions to occur within a given timestep are randomly selected using a probability distribution that is weighted by the abundances of the reaction requirements. That is, the limits of each reaction are calculated assuming that all of the available required resources would be allocated to the given reaction. The probability distribution for selecting given reactions is weighted by these calculated limits. Reactions are performed one by one until insufficient resources exist to perform any additional reactions. Intermediate steps in the aminoacylation of tRNAs are not represented.

Integration

The `tRNA Aminoacylation` process class reads from and writes to the `Rna` state class. It reads in the counts of free and aminoacylated tRNAs and tmRNA, and writes back the updated counts.

Initial Conditions

All of the tRNAs are initialized to an aminoacylated state.

Dynamic Computation

At each timestep,

1. For each of the 39 possible reactions (36 tRNA aminoacylations, 1 tmRNA aminoacylation, 1 Glu-Gln

amidotransfer, 1 methionylformyltransfer), i, calculate the maximum number of reactions that can occur:

$$\text{Reaction Limit } i = \min \begin{cases} \text{Number of available required metabolites (amino acid, ATP, ...)} \\ \text{Number of available required enzyme} \times k_{cat} \\ \text{Number of free tRNA or tmRNA} \end{cases}$$

2. The available ATP, amino acids, and enzyme activities used in the limits calculation in Step 1, may be double counted as multiple reactions may require the same metabolites or enzymes. Therefore, the selection of which reactions occur, within the calculated limits, is randomly determined.
   While resources are available:
   (a) Randomly select a reaction to perform, weighing the probability of selecting a given reaction by its calculated reaction limit
   (b) Decrement the used metabolites, enzyme activities, free tRNAs, free tmRNAs
   (c) Increment the produced metabolites, and modified RNA
   (d) Recalculate the reaction limits as in Step 1, based on the new availabilities

Chapter 4

Experimental Procedures

4.1 Media Composition

Each liter of SP-4 is comprised of broth base 600 ml (Mycoplasma Broth Base 3.5 g (BD 211458), Bacto Tryptone 10 g (BD 211705), Bacto Peptone 5.3 g (BD 211677), distilled water 598 ml, pH 7.5 by KOH and autoclaved), 20% glucose 25 ml (CalBioChem 346351), CMRL 1066 10X 50 ml (ATCC20-2207), 7.5% sodium bicarbonate 5 ml (EMDSX-0320-1), 200 mM L-glutamine 5 ml, 2% yeast extract solution 35 ml (BD 210933), 2% autoclaved TC Yeastolate 100 ml (BD 255772), fetal bovine serum albumin (heat inactivated at 55°C for 2 hours) 170 ml (Gibco 26140-079), penicillin G (100,000 $\mathrm{U\,ml^{-1}}$) 2.5 ml (Sigma P7794), and 0.5% phenol red 4 ml (Sigma P0290).

4.2 Frozen Stocks

Cells were harvested for storage as a frozen stock when the media in the 10 cm petri dish cultures was yellow (pH 6.3-6.7). The media from the 10 cm plate cultures was aspirated. Cells were collected by scraping the bottom of the plates, resuspended in 3 ml of FBS, and serial filtered through 1.2, 0.8, 0.45, and 0.2 µm polyethersulfone filters to sterilize and de-clump the cells. Stocks were stored at -80°C.

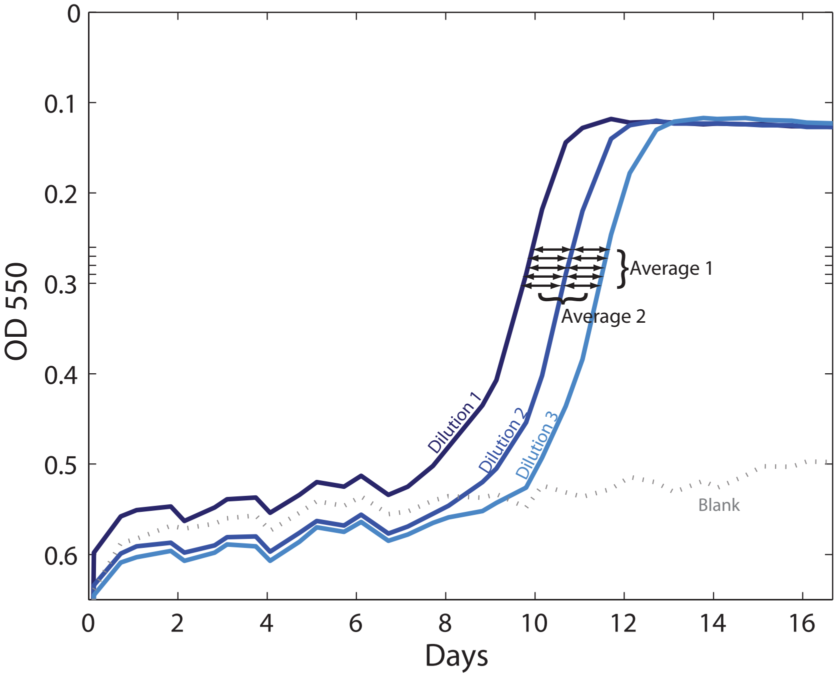
4.3 Colorimetric Growth Assay Serial Dilutions

Cells were harvested for serial dilutions when the media in the 10 cm petri dish culture was yellow (pH 6.3-6.7), and were harvested as described above for frozen stocks. The filtered suspension was used to make a serial dilution plate. The initial solution in the serial dilution plate is comprised of 50 ul of culture in 450 ul of SP-4. (For strains that were especially difficult to culture, 100 ul culture was added to 400 ul SP-4.) From this concentration we made three 5-fold serial dilutions. Dilutions were performed in triplicate for each culture, and 200 ul of each diluted sample was plated in a 96-well plate. The 96-well plates were stored at 37°C and 5% CO_2. Optical density readings were taken at 550 nm twice a day and used to make growth curves.

4.4 Colorimetric Growth Assay Calculations

The growth rate constant and doubling time for each strain was calculated using the growth curves of serially diluted bacterial cultures measured by the colorimetric assay and the serial dilution factor.

Assuming exponential growth, the cell concentration (C) in dilution1 can be defined as:



Schematic S16. Comparison of consecutive dilutions.

$$C_{\text{dilution1,time}_x} = C_{\text{dilution1,time}_0} \exp(\text{growth rate} \times \text{time}_x) \tag{S46}$$

And the concentration of dilution2 as:

$$C_{\text{dilution2,time}_y} = C_{\text{dilution2,time}_0} \exp(\text{growth rate} \times \text{time}_y) \tag{S47}$$

Because the concentration of a dilution at the start of the experiment is five times less than that of the previous dilution:

$$C_{\text{dilution1,time}_0} = 5C_{\text{dilution2,time}_0} \tag{S48}$$

Furthermore, the two dilutions reach a given OD550 at different times x and y (see horizontal arrows in Schematic S16):

$$C_{\text{dilution1,time}_x} = C_{\text{dilution2,time}_y} \tag{S49}$$

(This calculation was done at 5 OD550 values:0.26, 0.27, 0.28, 0.29, and 0.30)

By substitution, the relation for growth rate constant (for a specific pair of cultures and OD500) and is defined:

$$\text{growth rate constant} = \frac{\ln(5)}{\text{time}_y - \text{time}_x} \tag{S50}$$

The doubling time is then calculated as:

$$\text{doubling time} = \frac{\ln(2)}{\text{growth rate constant}} \tag{S51}$$

4.5 Quantitative PCR to Measure Cell Growth

Some strains grew so slowly that the colorimetric assay was inadequate to determine a growth rate. In these cases, a DNA quantitfication method was used to estimate the growth rate. This method calculates the number of chromosomes in a sample, and assumes that changes in chromosome count is correlated with changes in the *M. genitalium* cell count. The strains were cultured in 4 mL SP-4 and incubated at 37°C with 5% CO_2. Three replicate cultures were made for each of 14 days of harvest, resulting in 42 cultures per strain. Each day, cells were harvested by scraping the bottom of the plate, spun down at 4575 xg for 15 minutes, and resuspended in 20-400 µl of TE with 1% SDS. Samples were run on a 0.8% electrophoresis gel, and the DNA band was quantified using a Typhoon scanner and ImageQuant. The slope of the exponential region of the resulting growth curves was used to calculate a growth rate constant and doubling time (see Table S1). Comparing the relative growth of wild-type and the *tkt* gene deletion strain, we see that the results of the PCR method match those of the colorimetric assay.

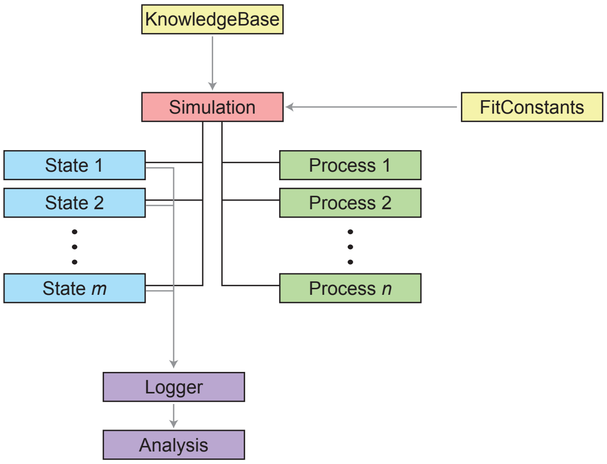
Appendix A

Computational Implementation

A.1 Whole-Cell Model Architecture

This chapter outlines the computational implementation of the whole-cell model and briefly summarizes the most important whole-cell model classes and functions. All of the source code for the whole-cell model, as well as comprehensive documentation of each class and function is freely available at SimTK: `http://www.simtk.org/home/wholecell`.

As illustrated in Schematic S17, the `Simulation` class coordinates the entire whole-cell model and is the primary class users interact with to execute the whole-cell model.



Schematic S17. Whole-cell model architecture.

The `Simulation` class performs several functions to coordinate the whole-cell model. First, the `Simulation` class instantiates all of the states and processes. Second, the `Simulation` class constructs an object graph of the states and processes. Specifically, the `Simulation` class triggers the states and process to store references to the other states and processes with which they will interact during the simulation. Third, the `Simulation` class initializes the structural and quantitative parameters of each state and process. Specifically, the `Simulation` class passes the `KnowledgeBase` to the states and processes and triggers the states and process to retrieve data from the knowledge base. Fourth, the `Simulation` class manages the calculation of the initial cell state by triggering initialization methods of the states and processes. Fifth, the `Simulation` class oversees the simulation time line. Sixth, at each time step of the simulation the `Simulation` class orchestrates the calculation of the temporal evolution of the cell state by allocating shared resources among the processes and triggering the temporal evolution method of each of the processes. Finally, at each time step of the simulation the `Simulation` class triggers loggers to store the predicted values of the states.

The role of cell state allocation is most clearly illustrated by counterexample. If processes were executed serially and each process was passed the total count of each metabolite at the beginning of the time step, then there would be no constraint which prevents the processes from together using more than the total amount of a metabolite, resulting in a negative and unphysical metabolite count. If alternatively the processes were executed serially in the same order at each time step and each process updated the cell state following its execution, then there would be no constraint which prevents the earliest executed processes from using all of the copies of a metabolite needed by a later executed process. If processes were executed as in the previous example, except that processes were executed in a random order at each time step, then early processes could outcompete later processes as before, resulting in high frequency artifacts where processes randomly oscillate between metabolically on and off states on the same time scale as the simulation time step. In summary, input allocation has the benefit of reducing the process evaluation order dependence of the model without introducing high frequency artifacts.

The `KnowledgeBase` class represents the reconstructed *M. genitalium* knowledge base and provides this data to the states and processes. The `KnowledgeBase` is constructed outside the `Simulation` class, and is passed to the `Simulation` class to set the values of the parameters of the states and processes. The *M. genitalium* knowledge base was implemented in MySQL. The knowledge base web-interface was implemented in PHP.

Each cellular process sub-model is implemented as a subclass of `Process` and each cellular state is implemented as a subclass of `CellState`. This standardizes the implementation of the processes and states, and in particular, ensures that each process and state exposes a common interface. Specifically, each state class (1) implements a common method to retrieve structural and quantitative data from the `KnowledgeBase`, (2) implements a common method to initialize its value at the beginning of each simulation, (3) implements a common method to calculate its contribution to the total cell mass, and (4) sets the value of a common property which tells the loggers which properties to store at each time step.

Each process class implements common methods that (1) retrieve structural and quantitative data from the `KnowledgeBase`, (2) model each process' contribution to the initialization and temporal evolution of the cell state, and (3) calculate the instantaneous and total life cycle metabolic and enzymatic requirements of each process. In addition to implementing the `Process` interface, each process satisfied a contract with the cell state classes to conserve mass and synchronize the redundantly represented parts of the cell state. For example, the process classes maintained synchrony between the DNA-bound copy number of each protein complex represented by the `Protein Complex` state and the location of each DNA-bound protein complex represented by the `Chromosome` state. Unit testing was used to verify that this contract between the states and processes was upheld.

The role of the `FitConstants` class is to fit the whole-cell model predictions to match the experimentally observed properties of *M. genitalium* including the mass doubling time, chemical composition, and RNA decay and expression. The `FitConstants` class implements an algorithm which computationally refines the values of the parameters of the states and processes. See Section 1.3 for further discussion of the simulation fitting algorithm.

The role of the `Logger` classes is to store the simulated dynamics represented by the cell states, and to later retrieve this information for computational analysis. Logged dynamics are plotted and analyzed by the several classes in the `analysis` package, including the `PaperFigures` class which produces Figure 2-6 of the accompanying manuscript.

A.2 Test-Driven Development

The whole-cell model was extensively tested throughout the development process to ensure correctness. Using the `MATLAB` xUnit framework[894], we developed 1,058 tests of the whole-cell model `MATLAB` code, including 100% of all state and process class methods and 92% of all state and process class lines. First, we developed 791 tests of each individual state and process class. For example, we developed several tests of the `Metab-`

olism process which check that the predicted and observed growth rates are approximately equal, that the predicted growth rate responds correctly to the external environment and enzyme copy numbers, and that the Metabolism process conserves mass. Second, we developed 41 tests of collections of related processes and states. For example, we developed several tests of the process classes responsible for modeling translation and protein maturation. These tests check that these processes receive adequate amino acids from the Metabolism process, translate mRNA, and produce mature proteins. Third, we developed five tests of the entire simulation which check that the model predicts exponential growth with growth rates approximately equal to the experimentally observed rate, that all of the mass fractions – DNA, RNA, protein, metabolite, etc. – of the cell are approximately equal to their experimentally observed sizes, that the simulated and observed gene expression are approximately equal, and that the simulation obeys mass conservation. Stochastic functions were tested by checking the mean of their outputs. Additionally, we developed tests of the KnowledgeBase, analysis, and utility classes.

The whole-cell model code was maintained using Subversion[896]. All tests were run at each code revision using the freely available Hudson continuous integration server[895], providing rapid feedback throughout the development process.

Separately, the model was experimentally validated as discussed in the accompanying manuscript and Section 1.6.

A.3 Distributed Simulation

The whole-cell model was implemented in object-oriented MATLAB (R2010b) and simulated on a 128 core Rocks (v5.4)[897] Linux cluster. Simulation code was compiled using the MATLAB compiler[900] and executed on the cluster nodes using the freely distributable MATLAB component runtime[900]. The open-source resource manager Torque[899] and cluster scheduler Maui[898] were used to distribute simulation jobs among the cluster nodes.

A.4 Computational Analysis

The whole-cell model simulations were analyzed using the MATLAB code in the edu.stanford.covert.cell.sim.analysis package.

A.5 Knowledge Base

The *M. genitalium* knowledge base was stored using a modified version of the BioWarehouse schema[918] in a MySQL relational database. Several tables and columns were added to the BioWarehouse schema primarily to represent additional functional genomic data. The knowledge base was viewed and edited using a web-interface implemented in PHP. The KnowledgeBase classes represented the knowledge base MySQL relational database in MATLAB.

A.6 Source Code Organization

The whole-cell model source code is available at SimTK, http://www.simtk.org/home/wholecell. The model source code is contained in the simulation/src directory and organized into several packages using the Java package naming convention:

- edu.stanford.covert.cell.sim: Simulation class and cellular process and state superclasses
- edu.stanford.covert.cell.sim.analysis: simulation analysis code
- edu.stanford.covert.cell.sim.constant: classes which represent constants such gene names
- edu.stanford.covert.cell.sim.process: implementations of all cellular processes
- edu.stanford.covert.cell.sim.state: implementations of all cell states
- edu.stanford.covert.cell.sim.util: utility classes for fitting, logging, plotting, printing, and more

- edu.stanford.covert.cell.kb: classes which represent the *M. genitalium* knowledge base
- edu.stanford.covert.db: classes for interacting with MySQL relational databases
- edu.stanford.covert.io: classes for reading and writing MAT- and JSON-formatted data
- edu.stanford.covert.test: classes for unit test logging, assessing code coverage, and mocking
- edu.stanford.covert.util: various utility functions

The simulation directory contains several unpackaged MATLAB, Perl, PHP, and Shell scripts for running simulations and unit tests on individual machines and compute clusters. The whole-cell model test code is similarly organized to the source code in the simulation/src_test directory. The knowledge base source code is located in the knowledgebase directory.

A.7 Key Classes & Functions

A.7.1 CellState

edu.stanford.covert.cell.sim.CellState is the base class from which all cell state classes are derived. CellState defines the interface each state exposes to Simulation. This interface enables Simulation to construct each state using data from the knowledge base, initialize the value of each state, account for the contribution of each state to the total cell mass, and log the dynamics of each state to disk. MoleculeCountState extends CellState by defining properties which represent the identity and copy numbers of molecules, and by providing a function which calculates the state's contribution to the total cell mass.

A.7.2 Chromosome

edu.stanford.covert.cell.sim.state.Chromosome implements the Chromosome state and provides an API which cellular process classes use to modify the chromosome object. To efficiently represent chromosomes, each property of the Chromosome class, except catenation, is implemented as an instance of the CircularSparseMat class of size $L \times 4$ where the first two columns represent the base-paired strands of the first chromosome and the second two columns represent the second chromosome. List S31 summarizes the properties of the Chromosome class. polymerizedRegions(i, j) sparsely represents the length in nt of a contiguous DNA strand starting at nucleotide i of strand/chromosome j. linkingNumbers(i, j) sparsely represents the linking number of a contiguous double-stranded DNA region starting at nucleotide i of strand/chromosome j. gapSites(i, j) is true if the sugar-phosphate of nucleotide i of strand/chromosome j is absent, and false otherwise. abasicSites(i, j) is true if the base of nucleotide i of strand/chromosome j is absent, and false otherwise. damagedBases(i, j) and damagedSugarPhosphates(i, j) are enumerations indicating the Metabolite identity of the base and sugar-phosphate of nucleotide i of strand/chromosome j. intrastrandCrossLinks(i, j) is true if nucleotide i of strand/chromosome j is cross linked to nucleotide $i + 1$ of the same strand and chromosome, and false otherwise. strandBreaks(i, j) is true if the phosphodiester bond between nucleotides i and $i + 1$ of strand/chromosome j is broken, and false otherwise. hollidayJunctions(i, j) is true if nucleotide i of strand/chromosome j forms a Holliday junction with the opposite strand, and false otherwise. monomerBoundSites(i, j) and complexBoundSites(i, j) are enumerations indicating the Protein Monomer or Protein Complex identity of the protein bound starting at nucleotide i of strand/chromosome j.

Due its complexity, the Chromosome class provides an API which process classes use to access and modify its properties. First, the API allows the Chromosome class to prevent processes from specifying invalid configurations of the chromosome state such as single-stranded binding proteins binding to double-stranded DNA or multiple proteins occupying the same nucleotide. Second, the API ensures that the Chromosome properties remains synchronized with the rest of the cell state. In particular, the API ensures that the monomerBoundSites and complexBoundSites properties remain synchronized with the copy numbers of DNA-bound proteins stored in the Protein Monomer and Protein Complex classes. Third, the API provides an abstraction layer over the chromosome representation, allowing process classes to focus on modeling biology. In particular, the API provides process classes methods for calculating the chromosomal regions accessible to each protein species and binding and releasing proteins to and from the chromosome. Furthermore, the API centralizes the modeled interactions among DNA-binding proteins, and specifically the reconstructed rules which describe the bound proteins each protein species is able to displace from the chromosome (see Table S3O). Fourth, the API eliminates the need to redundantly implement similar codes in each of the chromosome-interacting process classes. Finally, the API allows the `Chromosome` class to implement caching and other strategies to optimize simulation run-time performance. List S30 summarizes the public methods of the `Chromosome` class.

Each chromosome-interacting process class inherits from `ChromosomeProcessAspect`. This class provides several additional methods to help process classes interact with the `Chromosome` class. See Section A.7.3 for further discussion of the implementation of `ChromosomeProcessAspect`.

List S30. `Chromosome` **class public methods.**

Function	Description
Accessors	
`isRegionSingleStranded`	Checks if region is single-stranded.
`isRegionDoubleStranded`	Checks if region is double-stranded.
`isRegionPolymerized`	Checks if region is polymerized.
`isRegionNotPolymerized`	Checks if region is not polymerized.
`isRegionProteinFree`	Checks if region is not occupied by proteins.
`isRegionUndamaged`	Checks if region is unmodified (except methylated MunI R/M sites).
`isRegionAccessible`	Checks if a protein species can bind a region. Checks if region is polymerized, protein free or the query protein is capable of displacing all bound proteins, and undamaged.
`getAccessibleRegions`	Returns all regions accessible (polymerized, protein free or the query protein is capable of displacing all bound proteins, and undamaged) to a protein species.
`sampleAccessibleSites`	Efficiently returns a short list of regions accessible (polymerized, protein free or the query protein is capable of displacing all bound proteins, and undamaged) to a protein species.
`sampleAccessibleRegions`	Efficiently returns a short list of sites vulnerable to a specific type of DNA damage.
Modifiers	
`setRegionPolymerized`	Marks a region polymerized.
`setRegionUnwound`	Moves a region of strand 2 of chromosome 1 to strand 2 of chromosome 2. Conserves the linking number of chromosome 1 by increasing the superhelical density of the downstream double-stranded region of chromosome 1.
`setSiteProteinBound`	If the region is accessible to the query protein, binds query protein to region. Displaces any previously bound proteins. Updates `Protein Monomer` and `Protein Complex` states.
`setRegionProteinUnbound`	Releases all proteins bound to a region. Updates `Protein Monomer` and `Protein Complex` states.
`setSiteDamaged`	Introduces a gap site, abasic site, modified base or sugar phosphate, cross link, strand break, or Holliday junction at the query nucleotide.
`stochasticallySetProteinUnbound`	Stochastically releases all copies of a protein species at a specified probability. Updates `Protein Monomer` and `Protein Complex` states.
`modifyBoundProtein`	If the query protein and previous bound protein have the same footprint, changes the identity of the protein bound at a site. Updates `Protein Monomer` and `Protein Complex` states.

A.7.3 `ChromosomeProcessAspect`

`edu.stanford.covert.cell.sim.ChromosomeProcessAspect` provides several convenience methods which cellular process classes use to access and modify the properties of the `Chromosome` class. S32 summarizes the public methods of the `ChromosomeProcessAspect` class.

List S31. Computational representation of the `Chromosome` class.

Physical Property	Name	Size	Type
Polymerization	`polymerizedRegions`	$L \times 4$	`CircularSparseMat<int>`
Winding	`linkingNumbers`	$L \times 4$	`CircularSparseMat<double>`
Modification			
Gap site	`gapSites`	$L \times 4$	`CircularSparseMat<logical>`
Abasic site	`abasicSites`	$L \times 4$	`CircularSparseMat<logical>`
Sugar-phosphate	`damagedSugarPhosphates`	$L \times 4$	`CircularSparseMat<int>`
Base	`damagedBases`	$L \times 4$	`CircularSparseMat<int>`
Intrastrand cross link	`intrastrandCrossLinks`	$L \times 4$	`CircularSparseMat<logical>`
Strand break	`strandBreaks`	$L \times 4$	`CircularSparseMat<logical>`
Holliday junction	`hollidayJunctions`	$L \times 4$	`CircularSparseMat<logical>`
Protein occupancy			
Monomer	`monomerBoundSites`	$L \times 4$	`CircularSparseMat<int>`
Complex	`complexBoundSites`	$L \times 4$	`CircularSparseMat<int>`
Catenation	`segregated`	1×1	`logical`

List S32. `ChromosomeProcessAspect` class public methods.

Function	Description
Accessors	
`isDnaBound`	Checks if a site is bound by a specified protein species.
`findProteinInRegion`	Returns positions of a protein species within a specified region.
Modifiers	
`bindProteinToChromosome`	Convenience function for `setSiteProteinBound`. Updates local enzyme state and global `Protein Monomer` and `Protein Complex` states.
`bindProteinToChromosomeStochastically`	Binds multiple copies of a protein species to specified positions each with a specified probability. Updates local enzyme state and global `Protein Monomer` and `Protein Complex` states.
`modifyProteinOnChromosome`	Convenience function for `modifyBoundProtein`. Updates local enzyme state and global `Protein Monomer` and `Protein Complex` states.
`releaseProteinFromChromosome`	Convenience function for `stochasticallySetProteinUnbound`. Updates local enzyme state and global `Protein Monomer` and `Protein Complex` states.
`releaseProteinFromSites`	Convenience function for `setRegionProteinUnbound`. Updates local enzyme state and global `Protein Monomer` and `Protein Complex` states.

A.7.4 CircularSparseMat

edu.stanford.covert.util.CircularSparseMat efficiently represents sparse, circular, multidimensional matrices. CircularSparseMat inherits from SparseMat, adding support for circular matrix reference and assignment. Several properties of the Chromosome class are implemented as instances of CircularSparseMat.

A.7.5 Compartment

edu.stanford.covert.cell.sim.constant.Compartment defines the identifier and name of each of the six modeled compartments: cytosol, extracellular space, membrane, nucleoid, terminal organelle cytosol, and terminal organelle membrane. State classes which derive from MoleculeCountState represent the copy number of each molecule in each of these six compartments.

A.7.6 DatabaseLogger

edu.stanford.covert.cell.sim.util.DatabaseLogger provides methods to store and retrieve simulated single cell dynamics and simulation meta data to/from a relational database. DatabaseLogger implements the logger interface defined by Logger.

A.7.7 DiskLogger

edu.stanford.covert.cell.sim.util.DiskLogger was the primary logger used to store the simulated single cell dynamics. DiskLogger provides methods to store and retrieve the simulated dynamics of individual cells and simulation meta data to/from disk. DiskLogger implements the logger interface defined by Logger. SimulationEnsemble provides methods to retrieve simulated data stored by DiskLogger for multiple cells. SimulationDiskUtil provides methods to retrieve and search simulation meta data.

A.7.8 FitConstants

edu.stanford.covert.cell.sim.util.FitConstants provides several methods for fitting the simulated dynamics of cellular populations to experimental observations as well as computationally aligning the quantitative parameters of the cellular process sub-models such that the sub-models are mutually consistent. See Section 1.3 for further discussion of simulation fitting.

A.7.9 Gene

edu.stanford.covert.cell.sim.constant.Gene defines the identifier, name, type (m-, r-, s-, or t-RNA), location, length, and directionality of each gene.

A.7.10 Logger

DatabaseLogger, DiskLogger, and SummaryLogger define three ways of storing simulated single cell dynamics. DatabaseLogger provides methods to store and retrieve simulated data to/from a relational database. DiskLogger provides methods to store and retrieve simulated data directly to/from disk. SummaryLogger provides methods to store a small amount of key simulated data to/from disk. DatabaseLogger, DiskLogger, and SummaryLogger derive from edu.stanford.covert.cell.sim.util.Logger. Logger defines the common interface which Simulation uses to initialize, append, and finalize each simulation log.

A.7.11 KnowledgeBase

The primary function of edu.stanford.covert.cell.kb.KnowledgeBase is to represent the curated knowledge base of *M. genitalium* physiology, and to provide this information to the states and processes. In this way, KnowledgeBase serves as an abstraction layer between the knowledge base relational database and the whole-cell MATLAB model. Specifically, the knowledge base represents several reconstructed properties of *M. genitalium* including the genomic sequence; the location, length, and direction and observed expression of half-life of each gene; the transcription unit organization of the chromosome; the subunit composition of each macromolecular complex; the chemical composition of *M. genitalium* and its typical external environment; and the stoichiometry, kinetics, energetics, and catalysis of each chemical reaction.

A.7.12 KnowledgeBaseObject

edu.stanford.covert.cell.kb.KnowledgeBaseObject is the base class from which all MATLAB classes which represent objects contained the *M. genitalium* knowledge base are derived, including KnowledgeBase. See Section A.7.11 for further discussion of the role of the *M. genitalium* knowledge base.

A.7.13 MoleculeCountState

edu.stanford.covert.cell.sim.MoleculeCountState is the superclass for all cell state classes which represent the copy numbers of individual molecules including the Metabolite, Rna, Protein Monomer, and Protein Complex classes. MoleculeCountState extends CellState by defining properties which represent the identifier, name, and molecular weight of each molecular species and the copy number of each species in each of the six compartments defined by Compartment.

A.7.14 PaperFigures

The run method of edu.stanford.covert.cell.sim.analysis.PaperFigures produces Figure 2-6 presented in the accompanying manuscript.

A.7.15 polymerize

edu.stanford.covert.cell.sim.util.polymerize implements a model of nutrient and energy allocation among nascent polymers. The Replication, Transcription, and Translation processes use polymerize to model the allocation of dNTPs, NTPs, and amino acids among active DNA polymerases, RNA polymerases, and ribosomes. See Section 3.18, 3.25, or 3.27 for further discussion of the mathematical model implemented by the polymerize function.

A.7.16 Process

edu.stanford.covert.cell.sim.Process is the base class from which all process sub-model classes are derived. Process defines the public interface that each process exposes to Simulation. This interface enables Simulation to construct each sub-model using data from the knowledge base, evaluate each process' contributions to the initialization and temporal evolution of the cell's state, determine the parts of the cell state accessed and modified by each sub-model, and to calculate the life cycle and instantaneous metabolic demands of each process. ReactionProcess extends Process by adding several properties which represent the structure of the modeled biological network as well as one method which initializes the values of those properties using the knowledge base.

A.7.17 RandStream

The MATLAB built in RandStream class provides four low-level methods – rand, randi, randn, and randperm – for generating random numbers from a specific random stream. edu.stanford.covert.util.RandStream extends the functionality of the built in RandStream class by (1) providing methods to execute four statistics toolbox functions – multinomially-distributed random number generation (mnrnd), Poisson-distributed random number generation (poissrnd), a wrapper over several random number generation methods (random), and multinomially-distributed random number generation with and without replacement (randsample) – on a specific random stream, and (2) adding four methods randCounts, stochasticRound, randomlySelect-Rows, and randomlySelectNRows. randCounts returns the number of times each of N objects with counts m_i for $i = 1..N$ is selected without replacement. stochasticRound stochastically rounds elements of a matrix M. With probability M_{ij} mod 1 stochasticRound replaces M_{ij} with is ceiling, and otherwise replaces M_{ij} with its floor. randomlySelectRows randomly returns each row of a matrix with a specified probability. randomlySelectNRows randomly returns N rows of a matrix, each with equal probability.

A.7.18 ReactionProcess

edu.stanford.covert.cell.sim.ReactionProcess is the base class from which several process classes representing large network models including Metabolism are derived. ReactionProcess extends Process by defining several properties which represent the structure of the modeled biological network and by providing one method which initializes the values of those properties using the knowledge base.

A.7.19 SparseMat

edu.stanford.covert.util.SparseMat efficiently represents sparse multidimensional matrices. Externally, SparseMat behaves similarly to matrices created using the built in MATLAB sparse function with two exceptions: (1) SparseMat supports multidimensional matrices, whereas the built in sparse function supports only one- and two-dimensional matrices, and (2) SparseMat redefines linear indexing in reference and assignment operations as syntactic sugar for the composition of linear indexing with sub2ind. SparseMat supports many common matrix operations including addition, subtraction, multiplication, and division. Internally, SparseMat represents an N-dimensional matrix containing n non-zero elements as an $n \times (N+1)$ two-dimensional matrix containing the indices and values of each non-zero element. CircularSparseMat extends SparseMat by enabling circular matrix reference and assignment. Several properties of the Chromosome class are implemented as instances of CircularSparseMat.

A.7.20 Simulation

edu.stanford.covert.cell.sim.Simulation is the primary class users interact with to run and analyze whole-cell model simulations. The primary functions of Simulation are several-fold: (1) initializeConstants constructs the states and processes using data contained in the knowledge base, (2) initializeState randomly initializes the cell state, (3) evolveState sequentially executes the process sub-models in a random order, (4) applyOptions, applyParameters, applyPerturbations override the default values of each option and parameter, for example to simulate gene disruption, (5) run executes a complete simulation by initializing the cell state, iteratively executing evolveState, and optionally triggering loggers to store the predicted single cell dynamics.

A.7.21 SimulationEnsemble

edu.stanford.covert.cell.sim.util.SimulationEnsemble provides methods for retrieving simulated single cell dynamics stored by DiskLogger and SummaryLogger for multiple cells. DiskLogger and SummaryLogger provide methods for retrieving the simulated dynamics of individual cells.

A.7.22 SimulationDiskUtil

edu.stanford.covert.cell.sim.util.SimulationDiskUtil provides methods to retrieve and search simulation meta data generated by DiskLogger.

A.7.23 SimulationStateSideEffect

edu.stanford.covert.cell.sim.SimulationStateSideEffect and edu.stanford.covert.cell.sim.SimulationStateSideEffectItem represent the side effects of one process on parts of the cell state outside the direct focus of that process. For example, when a sub-model binds one protein to the chromosome, displacing another protein from the chromosome modeled by a different cellular process, these classes represent the side effect of displacing that second protein on the DNA-bound copy number of that second protein redundantly represented by the Protein Monomer state. During the execution of each cellular process sub-model, cellular states use SimulationStateSideEffect to keep track of their side effects on other parts of the cell state primarily modeled by other sub-models. After the completion of the execution of each sub-model, the Simulation class processes the accumulated side effects and updates the affected states. The principal advantages of this approach are two-fold. First, this approach minimizes direct communication among the cellular states which reduces complexity and run time. Second, this approach outlines the side effects of processes on parts of the cell state outside their direct purview.

Each atomic, mass-balanced modification of a distant part of the cell's state is represented by an instance of the `SimulationStateSideEffect` class of size 1 × 1 whose children of type `SimulationStateSideEffectItem` represent the specific effect of the modification on the other parts of the cell state. For example, each event where a process displaces a foreign protein from the chromosome is represented by an instance of the `SimulationStateSideEffect` class which has two children each of type `SimulationStateSideEffectItem`, one of which represents the decrement of the DNA-bound copy number of the displaced protein and a second which represents the increment of the free copy number of the displaced protein.

A.7.24 `SimulationStateSideEffectItem`

See Section A.7.23 for discussion of the `SimulationStateSideEffectItem` class.

A.7.25 `SummaryLogger`

`edu.stanford.covert.cell.sim.util.SummaryLogger` provides methods to store a key subset of simulated single cell dynamics to disk. `SummaryLogger` implements the common logger interface defined by `Logger`. `SimulationEnsemble` provides methods to retrieve results logged by `SummaryLogger`.

A.8 Third-Party Software

The *M. genitalium* whole-cell model and knowledge base were developed using the software libraries and applications listed below.

Knowledge Base

- BioWarehouse
  biowarehouse.ai.sri.com
- Excel
  office.microsoft.com/en-us/excel
- GeSHi
  qbnz.com/highlighter
- Marvin
  chemaxon.com/products/marvin
- MySQL
  mysql.com
- MySQL connector J
  mysql.com/products/connector
- Pear
  pear.php.net
- PHP
  php.net
- PHPExcel
  phpexcel.codeplex.com

Modeling

- GLPKmex
  glpkmex.sourceforge.net
- `MATLAB`
  mathworks.com/products/matlab
- `MATLAB` Utilities
  home.online.no/ pjacklam/matlab/software/util
- multiprod
  mathworks.com/matlabcentral/fileexchange/8773

Simulation

- CentOS
  centos.org
- JSON Marshaller
  code.google.com/p/jsonmarshaller
- `MATLAB` Compiler
  mathworks.com/products/compiler
- Maui
  clusterresources.com/products/maui
- Torque
  adaptivecomputing.com/products/torque.php
- Perl
  perl.org
- Rocks
  www.rocksclusters.org

Analysis & Visualization

- Apache
  apache.org
- boundedline
  mathworks.com/matlabcentral/fileexchange/27485
- Bullseye
  mathworks.com/matlabcentral/fileexchange/16458
- DHTML Window
  dynamicdrive.com/dynamicindex8
- ffmpeg
  ffmpeg.org
- flowplayer
  flowplayer.org
- Funct_Bezier
  mathworks.com/matlabcentral/fileexchange/6661
- Illustrator
  www.adobe.com/products/illustrator.html
- Inkscape
  inkscape.org
- LaTeX
  www.latex-project.org
- jqGrid
  trirand.com/blog
- pdftk
  pdflabs.com/tools/pdftk-the-pdf-toolkit
- PHP
  php.net
- propertygrid
  mathworks.com/matlabcentral/fileexchange/28732
- rude
  mathworks.com/matlabcentral/fileexchange/6436
- Silk
  famfamfam.com/lab/icons/silk
- spanFigure
  mathworks.com/matlabcentral/fileexchange/31604
- swtest
  mathworks.com/matlabcentral/fileexchange/13964
- tick2text
  mathworks.com/matlabcentral/fileexchange/16003
- Uniform
  uniformjs.com
- uitabpanel
  mathworks.com/matlabcentral/fileexchange/11546

Interactive Visualization

- alivepdf
  alivepdf.bytearray.org
- amfphp
  silexlabs.org/amfphp
- as3gif
  code.google.com/p/as3gif
- AS3FlexDB
  code.google.com/p/as3flexdb
- degrafa
  degrafa.org
- Flash
  adobe.com/products/flashplayer.html
- Flex
  adobe.com/products/flex.html
- flexlib
  code.google.com/p/flexlib
- Flex Menu Accelerators
  rphelan.com/2008/03/17/flex-menu-accelerators
- Flex Multiline Button
  forestandthetrees.com/flex-multiline-button
- MS Visual C#
  microsoft.com/visualstudio
- Premier
  adobe.com/products/premiere.html
- print-as3
  code.google.com/p/printf-as3
- Screen2Video
  viscomsoft.com/products/screen2video
- Tooltip Speech Bubble
  blog.flexmp.com/2008/09/10
- tweener
  code.google.com/p/tweener

Testing & Development

- absolutepath
  mathworks.com/matlabcentral/fileexchange/3857
- Doxygen
  www.stack.nl/ dimitri/doxygen
- Hudson
  hudson-ci.org
- MATLAB xUnit framework
  mathworks.com/matlabcentral/fileexchange/22846
- m2cpp
  mathworks.com/matlabcentral/fileexchange/25925
- m2html
  artefact.tk/software/matlab/m2html
- Subversion
  subversion.apache.org

Appendix B

List of Abbreviations

API	Application programming interface
BER	Base excision repair
DDR	Direct damage reversal
DSBR	Double-strand break repair
FBA	Flux-balance analysis
GAM	Growth-associated maintenance energy
GG	Global genomic
HR	Homologous recombination
MOMA	Minimization of metabolic adjustment
mRNA	Messenger RNA
NER	Nucleotide excision repair
NF-κB	Nuclear factor kappa-light-chain-enhancer of activated B cells
ODE	Ordinary differential equation
R/M	Restriction/modification
rRNA	Ribosomal RNA
SMC	Structural maintenance of chromosome protein
SP-4	*Spiroplasma* medium #4
sRNA	Small non-coding RNA
SRP	signal recognition particle
SSB	Single-stranded binding protein
TLR	Toll-like receptor
tRNA	Transfer RNA
TSS	Transcription start site
UV	Ultraviolet