Supporting Information

Sonification-enhanced lattice model animations for teaching the protein folding reaction

Carla Scaletti,[1*] Meredith M. Rickard,[2] Kurt J. Hebel,[1] Taras V. Pogorelov,[2,3,4,5,6] Stephen A. Taylor,[7] and Martin Gruebele[2,3,5,8*]

[1]Symbolic Sound Corporation, Champaign, IL 61820, United States; [2]Department of Chemistry, University of Illinois at Urbana-Champaign, IL 61801, United States; [3]Center for Biophysics and Quantitative Biology, University of Illinois at Urbana-Champaign, IL 61801, United States; [4]School of Chemical Science, University of Illinois at Urbana-Champaign, IL 61801, United States; [5]Beckman Institute for Advanced Science and Technology, University of Illinois at Urbana-Champaign, IL 61801, United States; [6]National Center for Supercomputer Applications, University of Illinois at Urbana-Champaign, IL 61801, United States; [7]School of Music, University of Illinois at Urbana-Champaign, IL 61801, United States; [8]Department of Physics, and University of Illinois at Urbana-Champaign, IL 61801, United States

Corresponding author emails: **Martin Gruebele** *mgruebel@illinois.edu*; **Carla Scaletti** *carla@symbolicsound.com*

Contents of this PDF:

Sample lecture slides are provided in a separate PDF file.

1. Videos: lattice model animations with data-driven sound

For best results, please select video quality HD1080 in the YouTube settings, and listen with stereo headphones or speakers.

- Video S1.1 is a sonification/visualization of the 4-bead lattice model shown in Figure 2 whose transition matrix is given in Figure S2.1.
- Video S1.2 An unevolved 6-bead chain where 2 and 6 are hydrophobic shows a 'glassy' or unevolved energy landscape that cannot fold into a single native state (See Figure 5 and "Unevolved sequence" in the Results section).
- Video S1.3 Beta hairpin, a 6-bead chain where 2 and 5 are hydrophobic, shows a 4-level energy landscape with a single, lowest energy state (See Figure 6A and "Beta hairpin" in the Results section).
- Video S1.4 Alpha helix, an 8-bead chain with hydrophobic elements in positions 2 and 7 shows a smoother energy landscape funnel. (See Figure 6B and "Alpha helix" in the Results).
- Video S1.5 WW domain, a 9-bead chain with hydrophobic residues in positions 2, 5, and 8, shows a 10-level energy landscape with multiple 'traps' or local minima.
- Video S1.6 WW domain animation and sonification where the radius of gyration (R_g) is mapped to frequency: the more compact the conformation, the lower the frequency. (See Figure 7 and discussion in Results).
- Playlist S1.7 is a collection of videos that can be used in conjunction with the sample lecture slides (see SI External File 6 **Sample lecture slides with comments**).
- Playlist S1.8 contains the videos that our students used in order to answer the homework questions (see SI Section 4 **Example homework and solutions**)

- Video S1.9 Using sound to track the formation of native vs non-native bonds in the 9-bead WW lattice model
- Video S1.10 WW domain model showing a temperature-dependent bias as to which states are more likely to be visited
- Video S1.11 WW domain model's traps and intermediate states
- Video S1.12 Beta hairpin model's traps and intermediate states. (Starts at a low temperature, then jumps up to a high temperature about halfway through the video).
- Video S1.13 Sonification of a simple coin-toss that is either fair, biased toward Heads, or biased toward Tails
- Video S1.14 Coin-toss with an unknown bias
- Video S1.15 Beta hairpin model running at a high temperature (and using a sound-mapping similar to the one used in coin toss to facilitate comparisons)
- Video S1.16 Beta hairpin model running at a low temperature
- Video S1.17 Beta hairpin model at the 'folding temperature', Tm where the temperature at which the likelihood of being in the folded state at the bottom of the funnel equals the likelihood of being in an unfolded state (the ensemble of unfolded structures).
- Video S1.18 Beta hairpin model at an unknown temperature. Is the temperature above or below Tm?
- Video S1.19 A visualization/sonification of R_g vs. time (vertical axis units are Å, horizontal axis units are μs.) The complete folding process of WW domain was calculated by M. Rickard et al., using the CHARMM22* force field in an all-atom fully solvated molecular dynamics simulation (see citation to Rickard, M. M.; Zhang, Y.; Pogorelov, T. V.; Gruebele, M. Crowding, Sticking, and Partial Folding of GTT WW Domain in a Small Cytoplasm Model. *J. Phys. Chem. B* **2020**, *124* (23), 4732–4740. https://doi.org/10.1021/acs.jpcb.0c02536 used in sample homework problem f, described in the main text). WW structures are shown on the right. The pitch you hear corresponds to R_g (higher pitch = higher R_g). R_g is also mapped to pan position. The standard deviation, σ_{Rg}, is mapped to the timbre of the sound and to the color of the trace on the left. VMD visualization (on the right) rendered by M. Rickard.
- Playlist S1.20 The same sound mapping (described in SI section 3) applied to several different lattice models

2. Method for implementing the lattice model in software

2.1 Representation of proteins on a lattice

A conformation is represented in Kyma as a string of directions, starting with the first bond pointing 'up', and successively labeling each joint (angle between two bonds) as 90° to the left (L), 90° to the right (R), or straight ahead (S). Transitions between conformations can be represented by a transition probability matrix (Figure S1). For example, if a flip from 'SL' to 'LL' is randomly chosen in Figure 2, it is accepted with probability 1 because it is downhill in energy.

Conformations that can interconvert via 2D rotation are not double-counted, as shown by the crossed-out configuration in Figure 2. In a more stringent version of the model, configurations accessible via 3D rotations can also be excluded, (this would eliminate one of the L-shaped conformations in Figure 2), and the bead at one end of the chain (e.g., the 'N-terminus' in a real protein) can be distinguished from the other end of the chain (see discussion of Figure 6).

This simple model, with only two amino acid types, is suitable only for small proteins: for large proteins, there is a high probability of contacts not being consistent with a single minimum energy state. At least 6 different amino acids (beads) are needed to define real protein structures uniquely,[32] but we stick with 2 here for simplicity.

2.2 Implementing the lattice model of protein dynamics as a state machine

- A lattice protein object (written in the Smalltalk programming language) which, given the number of beads, positions of the hydrophobic beads, and the energy (ε or ε') of each bond type, enumerates all allowed conformations, removes redundant conformations, and computes the allowable, energy-weighted transitions between conformations. Once initialized, the protein lattice object can create a state machine (specified as a formal grammar) and generate a CSV file where each row corresponds to one conformation's "observables" (see **Definitions of observables** below). A protein lattice object can also generate an image for each conformation and for each state of the energy funnel, as well as respond to other queries (for example, it can return a list of the traps and the shortest path(s) from any state to the folded state).
- Given the grammar generated by a lattice protein object, a `PushDownAutomaton` object in Kyma can generate a (non-terminating) sequence of conformations in real time, allowing for user interaction while the state machine is running; for example, you can change the temperature, change the sound parameters, pause the simulation, or set a different initial state.
- The values of the calculated observables are mapped, in real time, to the parameters of a sound synthesis or processing algorithm, by an `IndexedEventsFromData` object which also displays the animation frame for that conformation.
- The conformations of the 4-bead example (Figure 2 in the main text) define a 'finite state machine' coded in Kyma. The transitions that take the model protein from one conformation to the next can also be described by a transition probability matrix (Figure S1) to highlight the equivalence of the finite state machine to a Markov chain: a Markov chain is a series of events at time t, $t+\Delta t$ … such that the next state depends only on the current state and a randomly generated number, and not on the history of previous states.
- Visually, the state machine is represented as a graph G = (V, E) such as that shown in Figure 2 of the main text, where each vertex or node, V ∈ {LS, SL, RL, SS, LR, LL}, corresponds to one conformation, and each edge is an allowable, weighted transition from one shape to another in the state space. The transition probability matrix is not symmetrical, so the weight is different depending on the direction of the transition. For example, the transition from LL to LS has probability B/(2B+1) where B = exp(-ΔE/RT), whereas from LS to LL, the probability is ¼.

	SS	SL	RL	LS	LR	LL
SS	1	2	0	2	0	0
SL	1	1	1	0	0	1
RL	0	2	1	0	0	0
LS	1	0	0	1	1	1
LR	0	0	0	2	1	0
LL	0	B	0	B	0	1

Figure S1. The transition probability matrix for the four-bead system in Figure 2. Note that if two states can interconvert in two different ways (e.g. SS can go to SL by kinking the upper bond left, or the lower bond right plus a rotation by 180°), a weight of 2 is assigned. If a move increases the energy, then a Boltzmann weight B=exp(-ΔE/RT) is assigned, where ΔE is the positive energy difference in kJ/mole, and RT is temperature converted from Kelvin to units of energy. The actual probability is obtained by dividing the entry by the sum of its row. For example, the probability of going from LL (folded state) to SL (one kink at the end) is B/(0+B+0+B+0+1)=B/(2B+1).

On each time step, a new conformation (corresponding to a unique row of the CSV file) is selected by the `PushDownAutomaton` according to the current conformation, the likelihood of a transition from the current conformation to all other allowable conformations, the current temperature, and a random number.

Definitions of observables O = (E, R_g, d_{ee}, Q):

- The total energy E of the conformation, which depends on the number and type of bonds present
- The radius of gyration R_g, defined as the square root of the sum of the distances squared between each bead and the center of the shape; the 'mass' of each bead is assumed to be the same.
- The end-to-end distance d_{ee}, also labeled 'E2E' in some videos, from the first bead to the last bead
- The fraction of native contacts Q, ranging from 0 (unfolded) to 1 (folded) defined as the ratio of the number of native contacts in a conformation divided by the number of native contacts in the native state. In the lattice model, a 'native contact' is when two dark beads (hydrophobic amino acids) or two light beads (hydrophilic amino acids) are adjacent to one another as they should be in the native (lowest energy) structure.

Beta hairpin model: further details. The state space for this 6-bead beta hairpin model, like the 4-bead example, is generated by taking the space of all $3^4 = 81$ possible conformations, removing any conformations that overlay beads, that are rotations of another conformation in the 2D plane, or that are equivalent under order-reversal. One can also enforce a preference for chirality (e.g., every chain begins as either a straight or as a right turn), resulting in a state space of 22 shapes and a 4-level energy landscape.

Alpha helix model: further details. As with the 4-bead and beta hairpin models, we start by taking the space of all $3^6 = 729$ possible combinations of left, right, or straight angles at each joint, remove any conformations that cross themselves or that are rotations of another conformation in the 2D plane. To mimic chirality in actual proteins, we allow only the right-handed variant of each shape. To mimic the N- and C- termini of actual proteins, in this model, we do not consider the first and last beads to be equivalent, so we do not remove shapes that would be equivalent under order-reversal of the beads. After this pruning, there are 272 shapes left. Since alpha-helical peptides form near-optimal linear hydrogen bonds, we give only the helical hydrogen bonds ε=1 kJ/mole.

WW domain: further details. There are 740 shapes in the state space and 10 discrete energy levels in the energy landscape of the WW domain in our lattice model.

3. Step-by-step sound-mapping examples

This section describes some of the ways in which the observables of a lattice model were mapped to sound parameters.

3.1 WW 9-bead example Figure S2 shows an overview of the Kyma signal flow graph that generates the sound for Video S1.5, a 9-bead lattice model of the WW domain. (You can hear examples of this sound mapping applied to several different lattice models in Playlist S1.20.)

In a Kyma signal flow graph, the audio signal flows from left to right, and each icon represents a signal synthesis or processing algorithm or a meta-module that constructs a more complex graph.

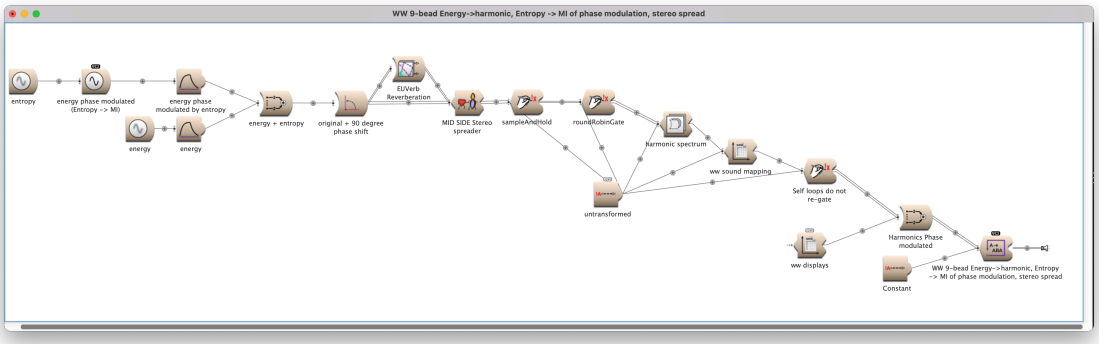


Figure S2. A signal flow graph for the 9-bead WW model.

The left half of the signal flow is shown in Figure S3: a phase-modulated oscillator whose frequency, modulation index, amplitude, stereo spread and reverberation can be controlled in real time by data.

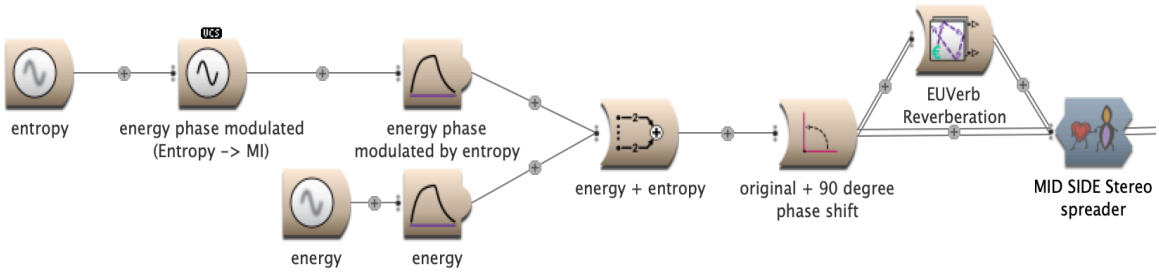


Figure S3. Sound synthesis and processing signal chain for two amplitude-enveloped oscillators, one of which is phase-modulated, followed by further processing to control the stereo spread.

In Figure S3, the module labeled ‘energy + entropy’ computes the sum of two sinusoidal oscillators whose frequency is controlled by the same free variable, `!Harmonic`. The oscillator labeled ‘energy phase modulated (Entropy ->MI)’ on the upper branch is phase-modulated by the oscillator labeled ‘entropy’; the modulator’s frequency is one octave below the carrier oscillator’s frequency

```
!Harmonic * 0.5
```

and its amplitude is `!Brightness`.

An exponential `AR` (an amplitude envelope shape multiplied by its input) is applied to the phase-modulated oscillator and a linear `AR` is applied to the unmodulated oscillator. Both amplitude envelopes have free variables `!Attack` and `!Decay` (time in units of seconds) and `!Gate`, a momentary switch which, when it changes from 0 to 1, triggers the attack portion of the envelope and holds it at unity gain until it changes from 1 to 0, when it enters the release portion of the envelope. The sum of the two oscillators is attenuated by the expression

```
0.5 * !Amp
```

A pseudo-stereo signal is created by phase-shifting the right channel by 90 degrees (relative to the left channel) using a *HilbertTransform*. The module labeled 'MID SIDE Stereo Spreader' separates its stereo input into two components: Mid (the part that appears in the middle of the stereo image) and Side (the part that appears on the sides of the stereo image). With this module you can apply processing to the Side, the Mid, or both (with independent controls on Mid and Side processing) prior to recombining Mid and Side to form the left and right channels of a standard stereo signal. In this example, reverberation is applied only to the Side signal, and the amount of Side signal added to the mix is controlled by a free variable, `!Spread`.

At this point in the signal flow graph, we have seven real-time controls that can affect the sound: `!Spread`, `!Amp`, `!Gate`, `!Attack`, `!Release`, `!Brightness`, and `!Harmonic`. These controls can be mapped to widgets on a virtual or physical control surface, or they can be mapped to values read from a data file.

Using a module called *IndexedEventsFromData*, you can specify which data variable values should be mapped to which of the live controls; any Realtime controls that are not mapped to data variables are still controllable from a virtual or physical control surface. The blue-highlighted module in Figure S4 is an *IndexedEventsFromData* with its parameter fields shown on the right.

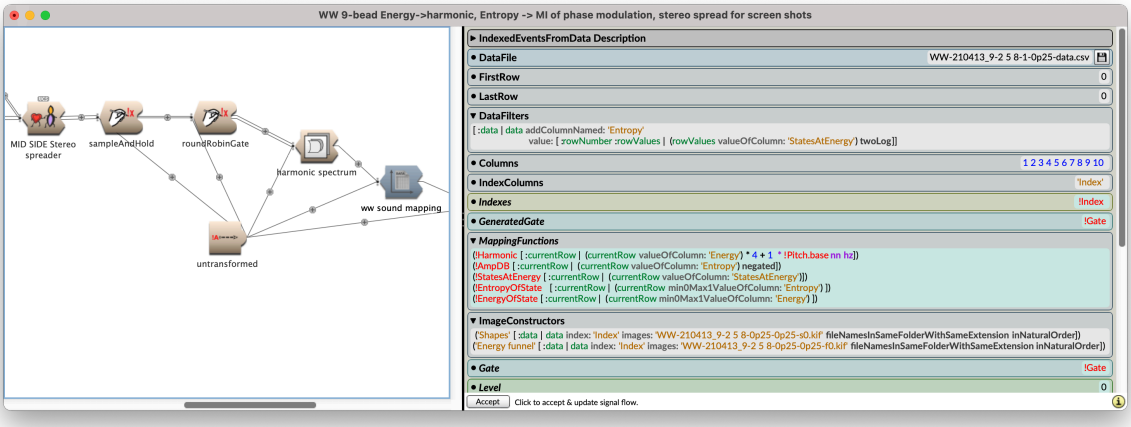


Figure S4. An `IndexedEventsFromData` accesses a data file row by number (the index) and can read the value from each column of that row by name or by column number. These values can be mapped to the realtime controls of the modules to the left of this one in the chain.

In DataFilters, you can create a modified virtual copy of the file specified in DataFile and use it as part of the mapping; in this example, we are adding a new column, 'Entropy' defined as the log base 2 of the value read from another column in that same row: 'StatesAtEnergy'.

A row of the file is selected using `!Index` and each time this module receives a `!Gate`, it also generates the real-time control named in GeneratedGate. (The same name was used for the incoming and the generated gate variables, but they are independent of one another).

In MappingFunctions you can list the name of a Realtime control followed by a block of code specifying how to map the value from a column in the currently selected row of the file to that real-time control. For example, the first line of the MappingFunctions is:

```
(!Harmonic [ :currentRow | (currentRow valueOfColumn: 'Energy') * 4 + 1 * !Pitch.base nn hz])
```

which means: read the value of column 'Energy' in the current row, multiply it by 4, add 1 and multiply the result by another real-time control, `!Pitch.base`, in units of linear frequency (Hz), and use the resulting value for the real-time control `!Harmonic` everywhere it appears in the modules to the left.

The mapping:

```
(!AmpDB [ :currentRow | (currentRow valueOfColumn: 'Entropy') negated])
```

takes the value of the newly created column 'Entropy', negates it, and uses that as an amplitude value.

The value in the column 'Entropy', normalized to [0,1], is mapped to a realtime control `!EntropyOfState` by the statement:

```
(!EntropyOfState [ :currentRow | (currentRow min0Max1ValueOfColumn: 'Entropy')])
```

In the ImageConstructors field, the first statement creates a "movie" of the changing conformations by using the current index to select and display a precomputed frame of the animation. The folding funnel graph animation is created in the same way, by indexing into a list of pre-computed frames of the animation.

The current index (`!Index`) controls which row of the data file to map to sound parameters and which frame of the animation to display, and each time it changes, it also emits a `!Gate` for the modules to the left of the `IndexedEventsFromData`. But where does `!Index` come from?

The rightmost module in Figure S2, an instance of *`PushDownAutomaton`*, generates a sequence of state numbers (`!Index`) in real time, allowing for user interaction with the sound and model parameters while the state machine is running. The state transition rules for this lattice model are encoded as a production rule specifying how you can leave a source state, emit a token, and transition to a destination state:

```
StateVariables -> (terminal StateVariabled) {weight1}.
```

Since each micro state can transition to itself, the first few rules specify self loops; for example, the rule:

```
SSSSRLR -> (t_9_1111020 SSSSRLR) {1}.
```

specifies that, if you are in state SSSSRLR, you can output the terminal token t_9, and take the self-loop transition back to state SSSSRLR with probability {1}. Other rules (not visible) give the probability of SSSSRLR transitioning to other states, and the probability of transition to each of the possible destination states is the weight of that rule divided by the sum of all possible transition weights from state SSSSRLR (See Figure S5 for an example).

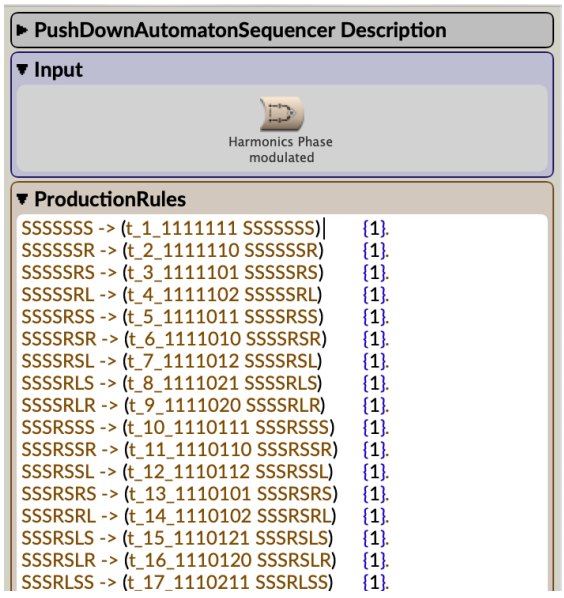


Figure S5. A *`PushDownAutomaton`* generates a sequence of values for `!Index` based on state transition rules encoded as a grammar.

Hiding the contents of the ProductionRules field (Figure S6) reveals more of the *PushDownAutomaton*'s parameter fields, including the duration of each time step (here, it is the inverse of the `!BPM` control converted to units of seconds); a list of GeneratedValues (here, we are generating the `!Index` used by the *IndexedEventsFromData* downstream), and the GeneratedGate control, also used by the downstream *IndexedEventsFromData*.

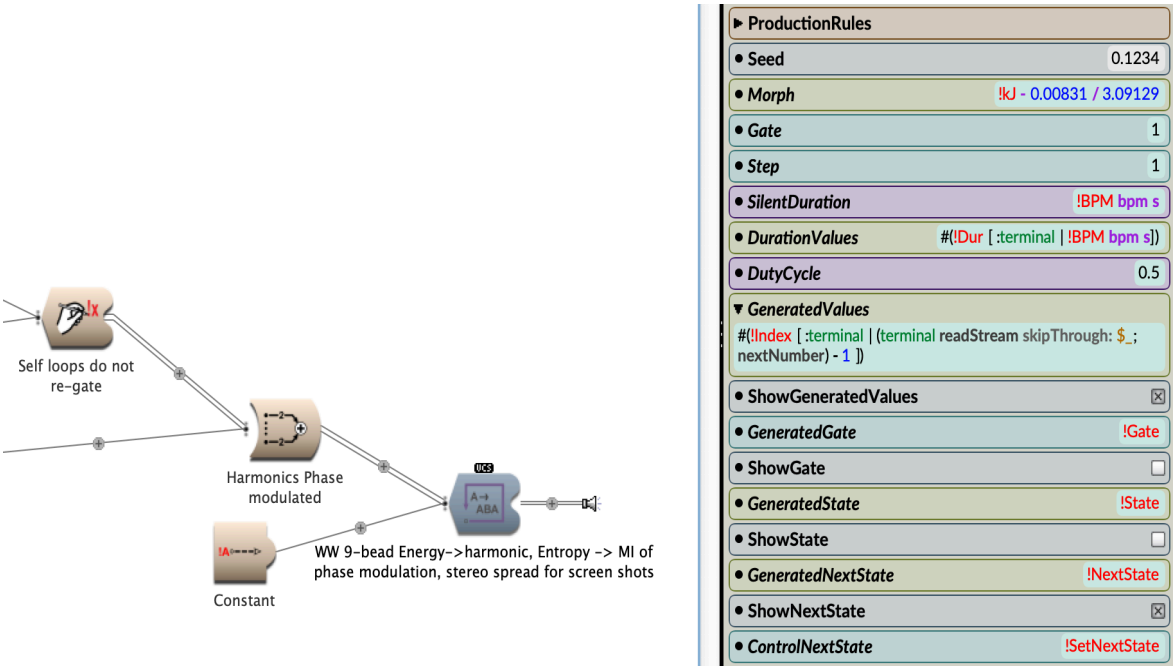


Figure S6. Other parameters of the *PushDownAutomaton* include generated control values (for example `!Index`), the duration of each time step, a control to morph between sets of probabilities, and means for selecting the start state by hand or free-running the state machine.

The module labeled 'Self loops do not re-gate' in Figure S6 is responsible for the rhythmic variations in the sound mappings; a new sound event is triggered only when the state machine enters a new state, not on time steps where it remains in the same state.

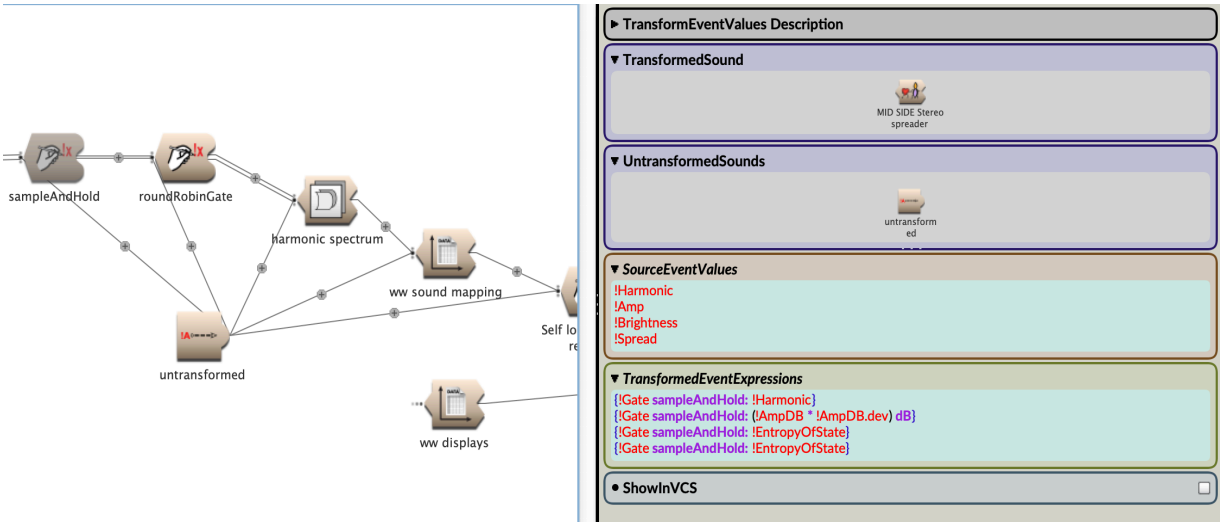


Figure S7. A *`TransformEventValues`* showing additional mappings and a sample-and-hold to hold the parameter values stable if a new `!Gate` arrives prior to the expiration of the current sound event.

Additional variable-to-sound-parameter mappings are specified in the *`TransformEventValues`* labeled 'sampleAndHold' in Figure S7. For example, `!Amp` (which controls the loudness of the sound) is rewritten to depend on `!AmpDB` (which in turn, depends on the computed Entropy column negated):

```
{!Gate sampleAndHold: (!AmpDB * !AmpDB.dev) dB}
```

The stereo `!Spread` and `!Brightness` (modulation index) controls are both mapped to the [0,1]-normalized value of Entropy.

The module labeled 'harmonic spectrum' in Figure S7 is a meta-module called a *`Replicator`* that creates a mix of multiple copies of the signal flow chain to its left. In effect, this is creating a pool of sound generators, each of which can be started before the previous sound event has expired, allowing the decay of the previous sound event to overlap with the attack of a new sound event.

The lower branch, labeled 'ww displays' in Figure S7, is purely for graphic display generation. This is where the normalized Entropy is plotted against the normalized Energy and composited with the mirror of the Entropy to create the gray and black, funnel-shaped trace in the display.

3.2 Creation of the Coin Toss Sonifications The sonification for the coin-toss question in the homework was produced using the *Noise* object in Kyma to generate a stream of random numbers. On each coin toss, the current output of the noise generator is sampled and held, and its sign {-1, 1} is used to modulate the frequency of an amplitude-enveloped oscillator whose envelope is triggered with each coin toss. To create an unfair coin, we shift the mean of the white noise up or down. For the homework video, we used discrete mean values from 1 to -1 in steps of 0.2: {1, 0.8, 0.6, … 0, … -0.2, -0.4, … -1}.

3.3 Using sound to track the formation of native vs. non-native bonds In Video S1.9, a sound event is triggered each time the state changes. States corresponding to conformations where no bonds are formed are heard as filtered noise. If a bond can form, then each bonded pair is mapped to pair of pitches (where the positions of the beads in the chain are mapped to scale steps). A native bond triggers a pair of synthesized plucked strings in the right channel. A non-native bond triggers a pair of violin pizzicato

samples one octave higher on the left channel. The background sound is narrow-band-pass filtered noise centered at a pitch that corresponds to the energy level of the current state:

```
!energyBand * 12 nn + 55 nn
```

where `!energyBand` is in the range of [0, 2.5], 55 nn corresponds to the pitch 3rd octave g, and 12 nn specifies that the range is 12 half steps (one octave).

The bead's position in the chain corresponds to the scale step of a major scale based on G, so the scale steps 1 through 7 correspond to pitch classes: g a b c d e f#. Native bonds 1-6, 2-5, 4-9, 5-8 correspond to the pitch dyads: g-e, a-d, c-a, d-g, and non-native bonds 1-4, 3-6, 4-7, 6-9 correspond to pitch dyads: g-c, b-e, c-f#, e-a

3.4 Mapping each state to a percussive sample with pitch offset In Video 1.12, entering a new state triggers a percussive sample whose relative pitch is determined by the energy level, offset to assign a unique pitch to each state at the same energy level.

3.5 Others For more sound examples, see the videos in Playlist S1.7, Playlist S1.8, and Playlist S1.20.

4. Example homework and solutions

Sample homework assignment: Protein Folding Sonification
Answers are shown after the "A:"

On this homework, you'll watch animations of Monte Carlo dynamics as an approximation of folding dynamics to answer questions. Each YouTube video [see SI section 1] is a data-driven visualization accompanied by a data-driven sonification (soundtrack) of an underlying lattice model.

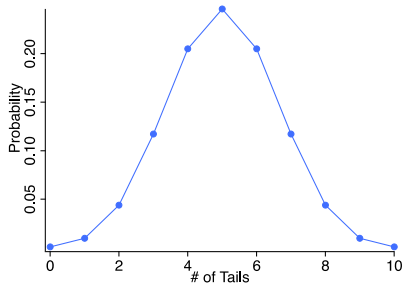
Note: On YouTube, watch the sonified videos 'full screen' using the ⛶ icon. Set to "HD" and control "playback speed" using the ⚙ icon to get a sharp video at the speed you want. Use headphones for best sound quality. The videos are usually longer than needed to find the right answer, so watch until you think you have the answer.

a. (2 pts) As a simple example, watch and listen to sonification of a coin toss at Fair and unfair coin tosses. Each toss is a tone whose pitch is high for Heads and low for Tails, a Fair coin is 5:5. Then listen to 'UnknownCoin', and **answer**: Is it 'Fair', biased towards 'Tails', or biased towards 'Heads'?

A: Biased towards Tails [The bias is 2:8 towards low pitch]

b. (4+4 pts) If a coin is tossed only 10 times, **histogram the probability** that it came out 10:0, 9:1, … 5:5, … 1:9, or 0:10. With just 10 tosses, **how many** excess tails or heads are needed before you can say with ≥90% certainty that a coin is unfair?

A: There's only 1 way of making it come out 10:0; there are 10 ways of making it come out 9:1 (e.g. THHHHHHHHHH, HTHHHHHHHHH etc.); there are 10·9/2! ways of distributing 8:2 (There are 10 places to put the first T, 9 places for the second, and the 2! removes the degeneracy of e.g. flipping the two T in HTHTHHHHHH, which is the same toss sequence); 10·9·8/3! for 7:3, and generally 10!/(10-T)!/T!, with a similar formula with H replacing T past the 5:5 point. So you get for the number of distinct arrangements 1:10:45:120:210:252: and back to 1. Normalizing by the sum $=2^{10}=1024$ of all possible arrangements to get probability, and histogramming.



Now to the certainty: you sum the probability at the edges (excess tails or heads), until you get >10% total; the step just before that means you can't be ≥90% sure that random chance did not give you the asymmetry. So, the probability of 10:0 and 0:10 adds up to 2·0.000976562≈0.20%. For 9:1 and 1:9, the probability is 2*0.00976562≈1.95%, totaling

2.15% so far. For 8:2 and 2:8, it's ≈8.79%, bringing the total to ≈10.94%. That would be already <90% certain. So to be ≥90% sure (in fact, about 97.85% sure), you need to see a 10:0, 0:10, 1:9 or 9:1. Anything less asymmetrical in 10 tosses could be random with <90% certainty that it's not random. So don't get too excited if you toss a coin 10 times and get 8 tails.

c. (1+2 pts) Now we are ready to watch and hear dynamics in a folding funnel. Watch and play the following three examples, discussed in class: 'HairpinHighT,' 'HairpinTm' and 'HairpinLowT.' Here an RNA or peptide hairpin can form by making a hydrophobic contact between beads 2 and 5 and a hydrophilic contact between beads 1 and 6. T_m is the folding temperature where the concentration of the folded state at the bottom of the funnel equals the concentration of the unfolded state = ensemble of unfolded structures. Visually, **does this system have traps** (i.e. one must go *up* in enthalpy to escape the trap before one can fold)? **How many**?

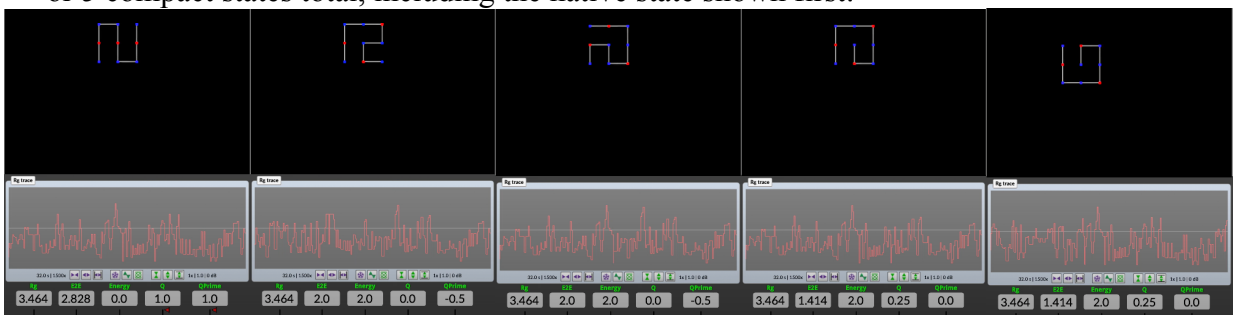
A: Yes. Two. [The two microstates that lie two enthalpy layers up from the bottom of the funnel can only be escaped going upwards in enthalpy; the right one was not visited very often, the left one was - random statistics!]

d. (2 pts) Now listen to 'HairpinUnknown' (without the visualization). **Is T>T_m, T≈T_m or T<T_m** in this example?

A: T>T_m [T is even higher than in the T>T_m example]

e. (1+2+4 pts) Next, watch the visualization and sonification 'WWlattice,' as the WW domain folds and unfolds to/from a triple-stranded beta sheet at the bottom of the folding funnel. Next to the funnel you will see the structures displayed as they are sampled by the Monte Carlo pseudo-kinetics, as well as an "oscilloscope display" of the radius of gyration R_g; deeper ''dongs in the sonification mean smaller R_g. **Is the ground state** (folded state) the only one with small R_g (3.464 in reduced units)? If not, **how many** equally compact 'traps' or 'intermediates' are there? **Draw them** out. [Hint: listening for the same deep 'dong' as the folded state, pausing YouTube and using the ',' (<) and '.' (>) to move one frame back/forward may make it easier to find traps than using the visuals, although that also works.]

A: No. There are 4 equally compact misfolded traps (all in the third enthalpy level from the top) or 5 compact states total, including the native state shown first:



If you counted traps #2 and #3, or traps #4 and #5 above as the same trap, you get full credit also: the N and C termini in WW domain *are distinct*, but the simulation does not label them, so pairs of traps with the termini exchanged look like rotated versions of one another, although they are different.

f. (3 pts) What could you add to the lattice model to get the simulation closer to real-life? **Give three examples.**

A: Add a third dimension to the lattice; allow for more than 2 types of beads with different pairwise interaction strengths for each bead; allow for 0- or 1-bead sidechains to distinguish small from large residues; make the angles continuous instead of 90° hops; you could include solvent degrees of freedom, for example by Brownian dynamics (Zaida (Zan) Luthey-Schulten's lectures) or by including solvent particles; other improvements are of course possible, and will also receive credit.

Optional: For an extra credit, listen to WW-lattice native vs non-native bonds. What **observations can you make** about the dynamics of this folding kinetics model based on sonification in this clip? (One extra point for each of up to three observations.)

A: Some examples: energy is sonified by a deeper tone for lower energy; fully extended or solvate structures have a swishy sound to identify them.

5. Survey instruments used in the PHY 498 and MUS 208 courses to assess student response

a) PHYS 498 anonymous student survey:

1. I am a... (fill in grad/undergrad and major)

2. How was the speed of the lectures? (Too slow/Just right/Too fast)

3. How was the difficulty of the material presented? (Too hard/Just right/Too easy)

4. What was the most interesting thing you learned from the lectures? (Brief open-ended text answer)

5. What are you still confused about with regard to topics covered in Dr. Gruebele's lectures? (Brief open-ended text answer)

6. Have you 'heard' data (sonification) before this homework, in addition to 'seeing' data (visualization)? (Yes/No)

7. A chain of amino acids always goes through the same pathway when transitioning from the unfolded to the native state? (Yes/No)

8. Which part of problem (3) had the clearest and most informative video? (of abcdefg)

9. Which part of problem (3) had the most difficult to follow video? (of abcdefg)

10. How likely are you to incorporate sonification in your own research? (1=very likely, 5=very unlikely)

11. Do you have any additional comments? (Brief open-ended text answer)

This instrument is to be administered to students who participated in sonification lectures. The results will be used in an educational research publication only in an aggregate manner, such as bar graphs of answer distribution, or brief quotes from open-ended questions where all identifiers have been removed, such as "Sonification was really interesting." or "Sonification did not help me with the visualizations."

b) PHYS 498 homework results:

In addition, we will analyze statistics of the homework results, such as fraction of correct/wrong answers on each question. Only aggregate statistics will be presented, no individual homework information.

c) PHYS 498 student profile questions:

1. Are you undergraduate or graduate student?

2. What is your major (i.e. Physics?), and in what year are you in?

3. If you are an undergraduate, what level of physics courses have you taken? Have you taken a statistical mechanics course?

4. What, if any, biology have you had in high school, college

5. Have you done any biophysics-related research?

6. Anything else relevant to the course, or anything things you want to get out of taking the course?

The answers to these questions will be reported only in aggregate form, that is, "what fraction of students in the class were physics, biophysics or chemistry majors" or "X% of students has a biology class in high school or college."

d) MUS 208 data sonification anonymous questions:

1. Have you 'heard' data (sonification) before this class, in addition to 'seeing' data (visualization)? Y/N

2. Unknown coin: is it fair: Biased towards heads (higher pitch)? Or biased towards tails (lower pitch)?
    10 (heads): 0 (tails)
    9:1
    8:2
    7:3
    6:4
    5:5
    4:6
    3:7
    2:8
    1:0

3. Does a chain of amino acids always to through the same pathway when transitioning from the unfolded to the native state? Y/N

4. Which part of problem (3) had the clearest and most informative video?
    a (Coin toss)
    b (Coin toss unknown)
    c (Hairpin high/low temperature)
    d (Hairpin unknown temperature)
    e (WW lattice)
    f (Rg vs. time)
    g (uRg vs. oRg)

5. Which part of problem (3) had the most difficult to follow video?
    a
    b
    c
    d
    e
    f
    g

6. How likely are you to try sonification (or data-driven music) yourself?
    Not at all
    Probably not
    Maybe
    I'm interested
    Definitely