Stochastic Process Notation

- Random variables S_i, $S_i = s \in \mathcal{A}$.
- Infinite sequence of random variables: $\overleftrightarrow{S} = \ldots S_{-1} S_0 S_1 S_2 \ldots$
- Block of L consecutive variables: $S^L = S_1, \ldots, S_L$.
- $\Pr(s_i, s_{i+1}, \ldots, s_{i+L-1}) = \Pr(s^L)$
- Assume translation invariance or stationarity:

$$\Pr(\mathrm{s_i}, \mathrm{s_{i+1}}, \cdots, \mathrm{s_{i+L-1}}) = \Pr(\mathrm{s_1}, \mathrm{s_2}, \cdots, \mathrm{s_L}) .$$

- Left half ("past"): $\overleftarrow{S} \equiv \cdots S_{-3} S_{-2} S_{-1}$
- Right half ("future"): $\overrightarrow{S} \equiv S_0 S_1 S_2 \cdots$

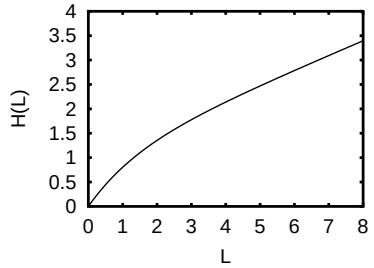
$$\cdots 1101010010110101010100100101001 0 \cdots$$

Entropy Growth

- Entropy of L-block:

$$H(L) \equiv -\sum_{s^L \in \mathcal{A}^L} \Pr(s^L) \log_2 \Pr(s^L) .$$

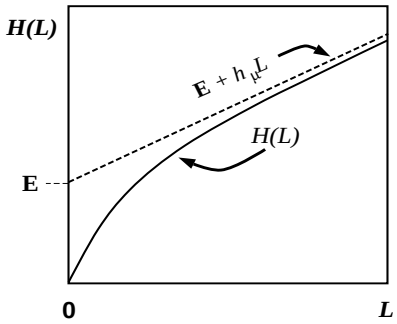
- $H(L) =$ average uncertainty about the outcome of L consecutive variables.



- $H(L)$ increases monotonically and asymptotes to a line
- We can learn a lot from the shape of $H(L)$.

Entropy Rate

- Let's first look at the slope of the line:



- Slope of $H(L)$: $h_\mu(L) \equiv H(L) - H(L-1)$
- Slope of the line to which $H(L)$ asymptotes is known as the *entropy rate:*

$$h_\mu = \lim_{L \to \infty} h_\mu(L).$$

Entropy Rate, continued

- Slope of the line to which $H(L)$ asymptotes is known as the *entropy rate:*

$$h_\mu = \lim_{L \to \infty} h_\mu(L).$$

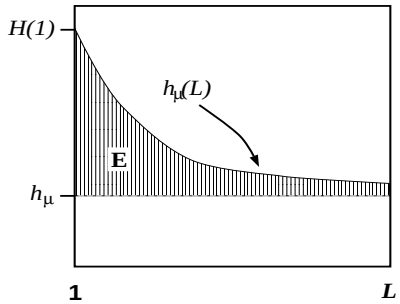
- $h_\mu(L) = H[S_L | S_1 S_1 \ldots S_{L-1}]$
- I.e., $h_\mu(L)$ is the average uncertainty of the next symbol, given that the previous L symbols have been observed.

Interpretations of Entropy Rate

- Uncertainty per symbol.
- Irreducible randomness: the randomness that persists even after accounting for correlations over arbitrarily large blocks of variables.
- The randomness that cannot be "explained away".
- Entropy rate is also known as the Entropy Density or the Metric Entropy.
- $h_\mu =$ Lyapunov exponent for many classes of 1D maps.
- The entropy rate may also be written: $h_\mu = \lim_{L\to\infty} \frac{H(L)}{L}$.
- h_μ is equivalent to thermodynamic entropy.
- These limits exist for all stationary processes.

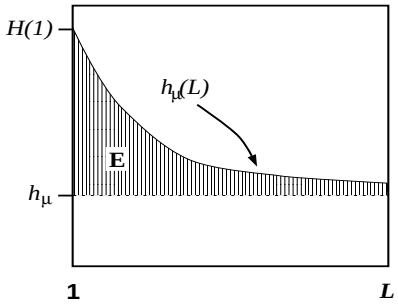
How does $h_\mu(L)$ approach h_μ?

- For finite L , $h_\mu(L) \geq h_\mu$. Thus, the system appears more random than it is.



- We can learn about the complexity of the system by looking at *how* the entropy density converges to h_μ.

The Excess Entropy



- The **excess entropy** captures the nature of the convergence and is defined as the shaded area above:

$$\mathbf{E} \equiv \sum_{L=1}^{\infty} [h_\mu(L) - h_\mu] \ .$$

- $\mathbf{E}$ is thus the total amount of randomness that is "explained away" by considering larger blocks of variables.

Excess Entropy: Other expressions and interpretations

Mutual information

- One can show that $\mathbf{E}$ is equal to the mutual information between the "past" and the "future":

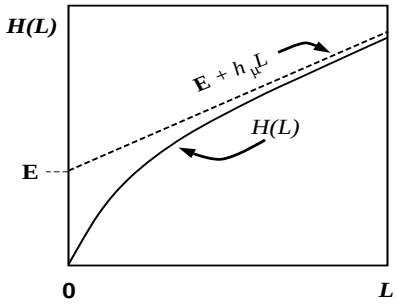
$$\mathbf{E} = I(\overleftarrow{S}; \overrightarrow{S}) \equiv \sum_{\{\overleftrightarrow{s}\}} \Pr(\overleftrightarrow{s}) \log_2 \left[\frac{\Pr(\overleftrightarrow{s})}{\Pr(\overleftarrow{s})\Pr(\overrightarrow{s})} \right] \ .$$

- $\mathbf{E}$ is thus the amount one half "remembers" about the other, the reduction in uncertainty about the future given knowledge of the past.
- Equivalently, $\mathbf{E}$ is the "cost of amnesia:" how much more random the future appears if all historical information is suddenly lost.

Excess Entropy: Other expressions and interpretations

Geometric View

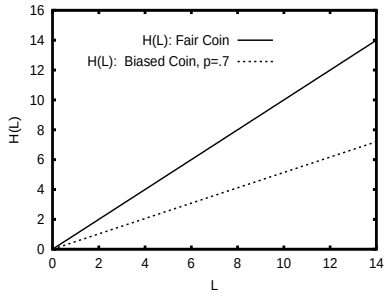
- $\mathbf{E}$ is the y-intercept of the straight line to which $H(L)$ asymptotes.
- $\mathbf{E} = \lim_{L\to\infty} \left[H(L) \, - \, h_\mu L\right]$.



Excess Entropy Summary

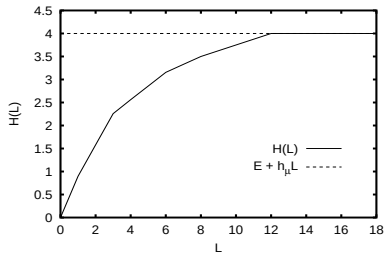
- Is a structural property of the system — measures a feature complementary to entropy.
- Measures memory or spatial structure.
- Lower bound for statistical complexity, minimum amount of information needed for minimal stochastic model of system

Example I: Fair Coin



- For fair coin, $h_\mu = 1$.
- For the biased coin, $h_\mu \approx 0.8831$.
- For both coins, $\mathbf{E} = 0$.
- Note that two systems with different entropy rates have the same excess entropy.

Example II: Periodic Sequence



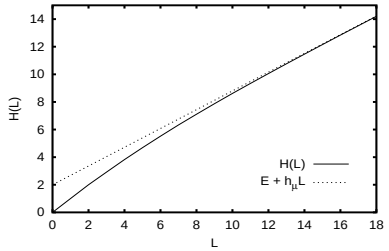
- Sequence: $\ldots 1010111011101110 \ldots$

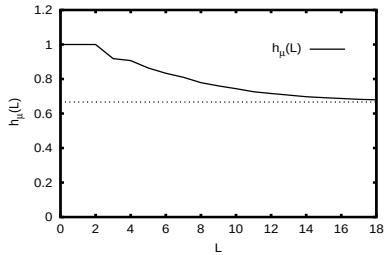
Example II, continued

- Sequence: $\ldots 1010111011101110 \ldots$
- $h_\mu \approx 0$; the sequence is perfectly predictable.
- $\mathbf{E} = \log_2 16 = 4$: four bits of phase information
- For any period-p sequence, $h_\mu = 0$ and $\mathbf{E} = \log_2 p$.

For more than you probably ever wanted to know about periodic sequences, see Feldman and Crutchfield, Synchronizing to Periodicity: The Transient Information and Synchronization Time of Periodic Sequences. *Advances in Complex Systems*. **7**(3-4): 329-355, 2004.

Example III: Random, Random, XOR





- Sequence: two random symbols, followed by the XOR of those symbols.

Example III, continued

- Sequence: two random symbols, followed by the XOR of those symbols.
- $h_\mu = \frac{2}{3}$; two-thirds of the symbols are unpredictable.
- $\mathbf{E} = \log_2 4 = 2$: two bits of phase information.
- For many more examples, see Crutchfield and Feldman, Chaos, 15: 25-54, 2003.

Excess Entropy: Notes on Terminology

All of the following terms refer to essentially the same quantity.

- **Excess Entropy:** Crutchfield, Packard, Feldman
- **Stored Information:** Shaw
- **Effective Measure Complexity:** Grassberger, Lindgren, Nordahl
- **Reduced (Rényi) Information:** Szépfalusy, Györgyi, Csordás
- **Complexity:** Li, Arnold
- **Predictive Information:** Nemenman, Bialek, Tishby

Excess Entropy: Selected References and Applications

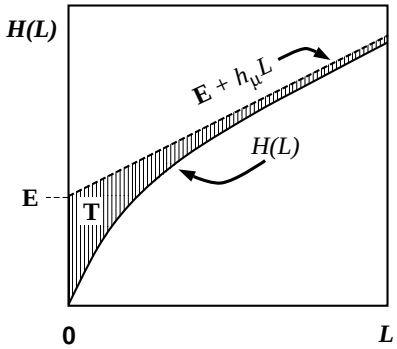
- Crutchfield and Packard, *Intl. J. Theo. Phys*, 21:433-466. (1982); *Physica D*, 7:201-223, 1983. [Dynamical systems]
- Shaw, "The Dripping Faucet ..., " Aerial Press, 1984. [A dripping faucet]
- Grassberger, *Intl. J. Theo. Phys*, 25:907-938, 1986. [Cellular automata (CAs), dynamical systems]
- Szépfalusy and Györgyi, *Phys. Rev. A*, 33:2852-2855, 1986. [Dynamical systems]
- Lindgren and Nordahl, *Complex Systems*, 2:409-440. (1988). [CAs, dynamical systems]
- Csordás and Szépfalusy, *Phys. Rev. A*, 39:4767-4777. 1989. [Dynamical Systems]
- Li, *Complex Systems*, 5:381-399, 1991.
- Freund, Ebeling, and Rateitschak, *Phys. Rev. E*, 54:5561-5566, 1996.
- Feldman and Crutchfield, SFI:98-04-026, 1998. Crutchfield and Feldman, *Phys. Rev. E* 55:R1239-42. 1997. [One-dimensional Ising models]

Excess Entropy: Selected References and Applications, continued

- Feldman and Crutchfield. *Physical Review E*, 67:051104. 2003. [Two-dimensional Ising models]
- Feixas, et al, *Eurographics*, Computer Graphics Forum, 18(3):95-106, 1999. [Image processing]
- Ebeling. *Physica D*, 1090:42-52. 1997. [Dynamical systems, written texts, music]
- Bialek, et al, *Neur. Comp.*, 13:2409-2463. 2001. [Long-range 1D Ising models, machine learning]

Transient Information $\mathbf{T}$

- $\mathbf{T} \equiv \sum_{L=1}^{\infty} [\mathbf{E} + h_\mu L - H(L)]$.
- $\mathbf{T}$ is related to the total uncertainty experienced while synchronizing to a process.



- The shaded area is the transient information T.
- $\mathbf{T}$ measures how difficult it is to synchronize to a sequence.

Some Applications in Agent-Based Modeling Settings

1. If an agent doesn't have sufficient memory, its environment will appear more random. In a quantitative sense, regularities that are missed (as measured by the excess entropy) are converted into randomness (as measured by the entropy rate).
   - Crutchfield and Feldman, Synchronizing to the Environment: Information Theoretic Constraints on Agent Learning. *Advances in Complex Systems.* 4. 251–264. 2001.
2. The average-case difficulty for an agent to synchronize to a periodic environment is measured by the transient information.
   - Feldman and Crutchfield. Synchronizing to a Periodic Signal: The Transient Information and Synchronization Time of Periodic Sequences. *Advances in Complex Systems.* 7. 329–355. 2004.

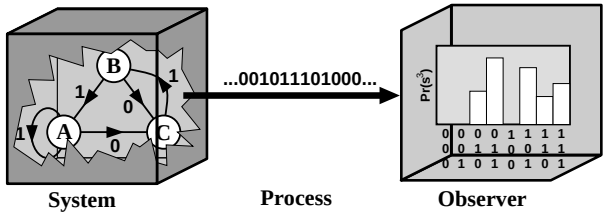
   More on this in Part XII.

Some Applications in Agent-Based Modeling Settings, continued

3. More generally it seems likely that the entropy and mutual information are useful tools for quantifying
   (a) properties of agents: e.g., how much memory they have
   (b) the behavior of agents: e.g, how unpredictably they act
   (c) properties of the environment: e.g., how structured it is

Estimating Probabilities

- $\mathbf{E}$ and h_μ can be estimated empirically by observing a process.



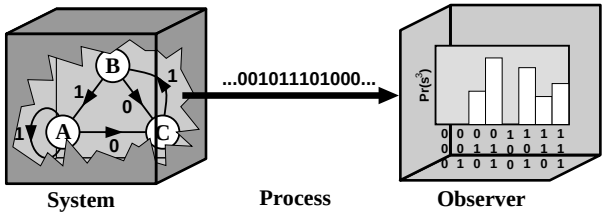
- One simply forms histograms of occurrences of particular sequences and uses these to estimate $\Pr(s^L)$, from which $\mathbf{E}$ and h_μ may be readily calculated.

However, this will lead to a biased under-estimate for h_μ. For more sophisticated and accurate ways of inferring h_μ, see, e.g.,

- Schürmann and Grassberger. Chaos 6:414-427. 1996.
- Nemenman. `http://arXiv.org/physics/0207009`. 2002.

A look ahead

- Note that the observer sees measurement symbols: 0's and 1's.



- It doesn't see inside the "black box" of the system.
- In particular, it doesn't see the internal, hidden states of the system, A, B, and C.
- Is there a way an observer can infer these hidden states?
- What is the meaning of *state*?

Part VI

Extensions to Shannon Entropy, Rényi Entropyies, Multifractals, Tsallis Entropies

Extensions to Shannon Entropy

- One of the requirements on the Shannon entropy H that is used to derive it is that H is independent of the way we group probabilities.
- Let's state this more precisely. We'll do so via an example.
- Consider the random variable X that can take on three outcomes, a, b, and c:
- $\Pr(a) = 1/2$, $\Pr(b) = 1/2$, and $\Pr(c) = 1/4$.
- It turns out that $H[X] = 3/2$.
- We can also view this as follows: Y can be a or Z, each with probability $1/2$. And Z can be b or c with probability $1/2$.
- $H[Y] = 1$, and $H[Z] = 1$.
- $H[X] = H[Y] + \frac{1}{2}H[Z]$.
- This last condition is an example of requiring H be independent of the way we group probabilities.

Rényi Entropy

- Let's relax the condition that H be independent of grouping.
- But still require that the entropy of independent variables be additive:

$$\Pr(X,Y) = \Pr(X)\Pr(Y) \implies$$
$$H[X,Y] = H[X] + H[Y] \quad .$$

- The result is a one-parameter family, the Rényi entropies:

$$H_q \equiv \frac{1}{q-1}\log_2 \sum_i p_i^q \ . \tag{5}$$

- This can be rewritten in the following, slightly less odd-looking way.

$$H_q = \frac{1}{q-1}\log_2 \sum_i p_i p_i^{q-1} \ . \tag{6}$$

$$H_q = \frac{1}{q-1}\log_2 \langle p_i^{q-1} \rangle \ . \tag{7}$$

Rényi Entropies: Properties and Comments

- H_q is a non-increasing function of q.
- H_1 is the Shannon entropy.
- H_0 is the topological entropy, the log of the number of states.
- There are coding theorems for Rényi entropy. Campbell. *Information and Control.* 8:423. 1966; Aggarwal and Bansal, `arXiv.cs.IT/0607029`. 2006.

Rényi and Thermodynamics

- The Rényi entropy allows one to apply the formalism of thermodynamics to any probability distribution. q plays a role similar to inverse temperature.

Escort Distributions

- Given a set of probabilities, we can always make a new set of probabilities as follows:

$$p_i \longrightarrow \frac{p_i^\beta}{Z} \ .$$

- β is a number that acts like $1/\mathrm{Temperature}$.
- $\beta = 1$: initial distribution
- $\beta = 0$: all states equally likely $\Rightarrow T = \infty$.
- $\beta = \infty$: only most probable state remains. This is the $T = 0$ "ground state."
- $\beta = -\infty$: only least probable state remains. This is the $T = 0^-$ "anti-ground" state.
- Loosely speaking, the Rényi entropy can be thought of the average surprise of the escort distribution with $\beta = q - 1$.
- The parameter β allows one to probe different regions of the distribution.

Thermodynamic Formalism

- The ideas on the previous slide can be extended in an elegant and fun way to apply thermodynamics to any probability distribution.
- This goes by many names; thermodynamic formalism, $S(u)$, $f(\alpha)$, multifractals, fluctuation spectrum, large deviation theory.
- This is a well developed, well understood approach. It is very enticing and very cool.
- In my experience, this approach doesn't speak directly to complexity or pattern, largely because thermodynamics doesn't have direct measures of complexity.
- For example, a biased coin (i.e. no correlations), has a "multifractal spectrum."
- This doesn't mean the multifractals are uninteresting. They are a natural way of quantifying the frequency with which deviations from typical behavior occurs.

Thermodynamic Formalism References

There are many confusing things written about the thermodynamic formalism. Some clear references:

- Young and Crutchfield. *Chaos, Solitons, and Fractals*. **4**:5. 1993.
- Beck and Schlögl, *Thermodynamics of Chaotic Systems*. Cambridge University Press. 1993.

Note: If you wish to estimate the fluctuation spectrum $S(u)$, a good way to do it is to first estimate the ϵ-machine and then calculate $S(u)$. Numerically calculating $S(u)$ directly can be inaccurate. See Young and Crutchfield.

Tsallis Entropy

- Define the following generalized entropy

$$S_q \equiv \frac{1 - \sum_i p^q}{q - 1} \ . \tag{8}$$

- This q is not the same as Rényi's q.
- S_q has the property that if $\Pr(X, Y) = \Pr(X)\Pr(Y)$, then:

$$S_q[X, Y] = S_q[X] + S_q[Y] + (1-q)S_q[X]S_q[Y] \ . \tag{9}$$

- I.e., S_q is not additive for independent events.
- One can generate a statistical mechanics and thermodynamics using Eq. (8) as a starting point.

Tsallis Entropy: Doubts and Concerns

- However, it is hard to see how a non-additive entropy can be physical.
- It has been claimed that S_q works well for systems with strong correlations. But it seems to me that the non-additivity creates spurious correlations rather than measuring correlations that are really there.
- My sense is that q is basically a fitting parameter. I don't know that it has a clear physical or mathematical meaning.
- Overall, I don't understand why Tsallis entropy is a big deal. I think this is the position of most, but by no means all, physicists.

Tsallis Entropy, references

But... you should read the papers and decide for yourself.

Some reviews:

- Tsallis. *Physica D*, **193**:3. 2004. http://arxiv.org/cond-mat/0403012.
- Tsallis, et al. arXiv.org/cond-mat/0309093. 2003.
- Tsallis and Brigatti, *Continuum Mechanics & Thermodynamics*, **16**:223 arXiv.org/cond-mat/0305606. 2004.

Some strenuous (and entertaining) critiques, and responses:

- Grassberger. *Physical Review Letters*, 95. 140601. 2005. http://arxiv.org/cond-mat/0508110
- Responses to Grassberger:
  - Robledo. arxiv.org/cond-mat/0510293
  - Tsallis. arxiv.org/cond-mat/0511213

- Nauenberg. *Physical Review E*. **67**:036114. 2003. arxiv.org/cond-mat/0210561
- Responses to Nauenberg and discussion:
  - Tsallis. *Physical Review E*. **69**:038102. 2004. arxiv.org/cond-mat/0304696
  - Nauenberg. *Physical Review E*. **69**:038102. 2004 arxiv.org/cond-mat/0305365

See also:

- www.cbpf.br/GrupPesq/StatisticalPhys/biblio.htm for an extensive bibliography on Tsallis entropy.

Part VII

A Sketch of Computation Theory: Machines, Languagues, and the Computation Hierarchy

A (Mostly) Informal Introduction to Computation Theory

- Computation theory is a different, more structural and less statistical approach to complexity, emergence, organization.
- Computation theory can be very elegant, rigorous, and mathematical.
- But I'll present little of the formalism. I think the math can obscure some of the basic ideas, which are really quite simple.

We'll begin with some examples in the form of a game:

- I'll give you the specification for a set
- I'll then show you an object, and you need to tell me if it's in the set or not

Example 1

The set $\mathcal{L}$ consists of all sequences of 0's and 1's of any length, except for those that have two 00's in a row.

Accept all sequences of 1's and 0's except for those which have two or more 0's in a row.

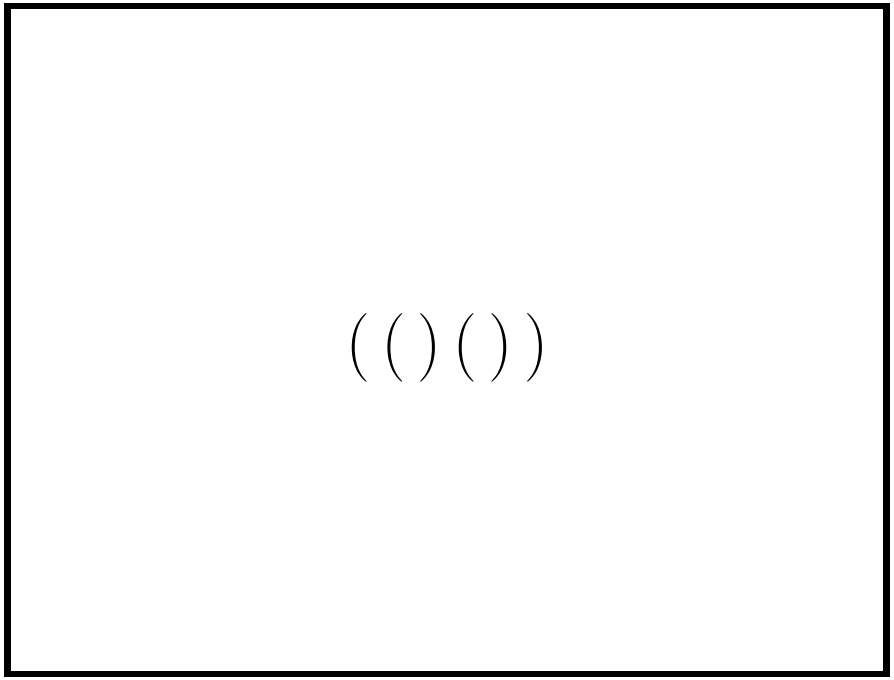
1110101101

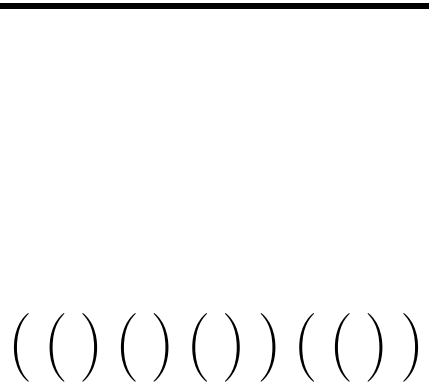
1101101001

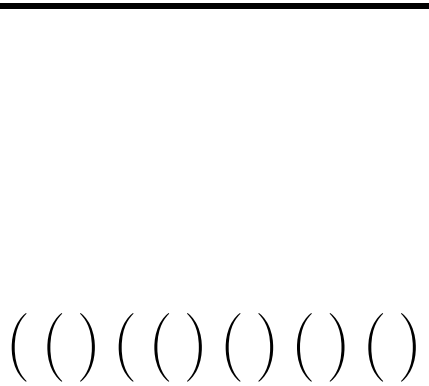
110110101011

Example 2:

The set $\mathcal{L}$ consists of all sequences of correctly balanced parentheses.







Example 3:

The set $\mathcal{L}$ consists of all sequences of 0's and 1's, except for those that contain a **prime** number of consecutive 0's!

$$1100011000001$$

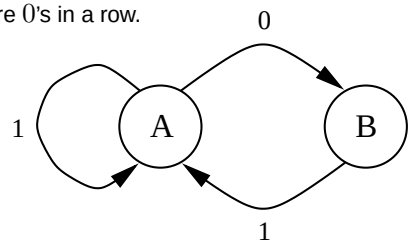
$$11000011$$

$$11100000000001$$

$$11 \overbrace{00\cdots00}^{1031 \text{ elements}} 11$$

What to learn from the examples

- There are qualitative differences between the procedures you just used to identify the strings on the previous slides.
- These distinctions lie at the heart of computation theory.
- We'll start by focusing on example 1.
- Your task was to accept all sequences of 1's and 0's except for those which have two or more 0's in a row.



- Sequence is OK if there exists a path through this machine
- Example: 1011001 is not in the set.

Finite State Machines

- The mathematical object on the previous page is known as a **Finite State Machine** or a **Finite Automaton**.
- Note that this two-state machine can correctly identify arbitrarily long sequences.
- The machine is a finite representation of the infinite set $\mathcal{L}$.

Some terminology and definitions

- A **Language** $\mathcal{L}$ is a set of words (symbol strings) formed from an **Alphabet** $\mathcal{A}$.
- We'll always assume a binary alphabet, $\mathcal{A} = \{0, 1\}$.

Big Idea: There is a correspondence between the rules needed to generate or describe a language, and the type of machine needed to recognize it.

Regular Expressions

- A **Regular Expression** is a way of writing down rules that generate a language.
- To generate a regexp, start with the symbols in $\mathcal{A}$.
- You can make new expressions via the following operations: grouping, concatenating, logical OR (denoted $+$), and closure $*$.
- Closure means 0 or more concatenations.
- Examples:
  1. $(0 + 1) = \{0, 1\}$
  2. $(0 + 1)^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$
  3. $(01)^* = \{\epsilon, 01, 0101, 010101, \ldots\}$
- (ϵ is the empty symbol.)

Regular Languages and FSM

- A language $\mathcal{L}$ is a **Regular Language** if and only if it can be generated by a regular expression.
- A puzzle: what is the regular expression that generates the language of example 1?

Two important results:

1. For any regular language, there is an FSM that recognizes it.
2. Any language generated by an FSM is regular.

Notes on terminology:

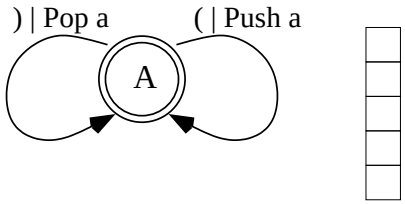
- A regular expression is a **rule**.
- A regular language is a **set**.
- A FSM is a **machine**.

Regular languages $\leftrightarrow$ FSM's is the first example of the correspondence between sets and the procedures or machines needed to recognize them.

Revisiting Parentheses

- This example is different than the last—you can't scan left to right unless you remember stuff.
- There is no FSM that can recognize this language. The problem is that as the sequence grows in length, the number of states necessary also grows.
- This task requires infinite memory. However, the memory only needs to be organized in a simple way.
- The parentheses language can be recognized by a device known as a **Pushdown Automata**.
- Put an object on the stack if you see a left paren $($ and take it off if you see a right paren $)$.
- If the stack is empty after scanning the sequence, then it is ok.

Pushdown Automata



- This is the PDA for the parentheses example
- If you see a "(", write (push) a symbol to the stack.
- If you see a ")", erase (pop) a symbol from the stack.
- The machine can only write to the top of the stack.
- This PDA can recognize balanced parentheses of any length.

Context-Free Languages

- The languages recognized by PDA are **context-free languages.**
- Regular languages are generated sequentiality—one symbol after the next.
- CFL's are generated by writing rules applied in parallel.
- For example, to generate the parentheses language, apply the following:

$$\begin{aligned} W &\rightarrow (V \\ V &\rightarrow (VV \text{ or }) \end{aligned}$$

- Start with W. The set of all possible applications of the above rules give you the set of all possible balanced parentheses.
- For example:

$$W,\ (V,\ ((VV,\ (()V\ (()(VV\ (()()V\ (()())$$

CFL Terminology

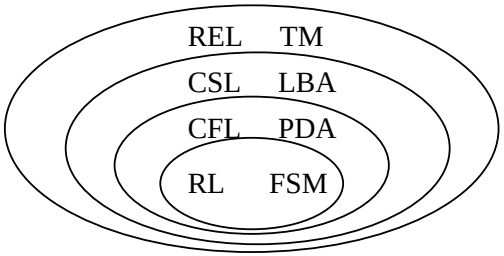
- $(,)$ are **terminals**, symbols in the alphabet $\mathcal{A}$.
- W, V are **variables**, symbols not in $\mathcal{A}$, to be eventually replaced by terminals.
- CFL's are context free in the sense that the production rule depends only on the variable, not on where the variable is in the string.

CFL Summary

- Every CFL can be recognized by a PDA, and every PDA produces a CFL.
- Also, FSM's are a proper subset of PDAs, and
- Regular Languages are a proper subset of CFL's
- We can thus divide languages into two classes, one of which is strictly more complex than the other.
- Are there even more complex languages? Yes ...

Chomsky Hierarchy

- The hierarchy continues:



- This hierarchy of languages/machines is known as the **Chomsky Hierarchy**.
- Each level in the hierarchy contains something new, and also contains all the languages at lower levels of the hierarchy.

Chomsky Hierarchy, terminology

- CSL = **Context Sensitive Language**. These are like CFL's, but allow transitions that depend on the position of the variable in the strings.
- LBA = **Linear Bounded Automata**. These are like PDA's, except:
  1. Controller can write anywhere on work tape.
  2. Work tape restricted to be a linear function of input.
- **Recursively Enumerable Languages** are those languages produced by an unrestricted grammar.
- An **Unrestricted Grammar** is like a CSL, but allows substitutions that shrink the length of the string.
- TM = **Turing Machines**. These are LBA's with linear tape restriction removed. These are the most powerful model of computation. (Example 3 requires a TM.) More on these later.

Chomsky Hierarchy, Conclusions

- Order languages (sets) by the type of machine needed to recognize elements of the language.
- There are qualitative difference between machines at different levels of the hierarchy.
- At lower levels of the hierarchy, there are algorithms for minimizing machines. (I.e., remove duplicate nodes.)
- The minimum machine can be viewed as a representation of the pattern contained in the language. The machine is a description of all the regularities.
- The size of the machine may be viewed as a measure of complexity.
- The machine itself reveals the "architecture" of the information processing.

Other computation theory notes

- It is possible to refine the Chomsky hierarchy with different sorts of machines. The result is a rich partial ordering of languages.
- To use computation theory as a basis for measuring complexity or structure, I think it's important to start at the bottom of the hierarchy and work your way up.

Computation Theory References

The basic material presented is quite standard and there are many references on it. Here are a few:

- Hopcroft and Ullman. Introduction to Automata Theory, Languages and Computation. Addison-Wesley. 1979. *A standard reference. Not my favorite, though. It's thorough and clear, but rather dense.*
- Brookshear. Theory of Computation: Formal Languages, Automata, and Complexity. Benjamin/Cummings. 1989. *I like this book. I find it much clearer than Hopcroft and Ullman.*

Computation theory applied to physical sequences

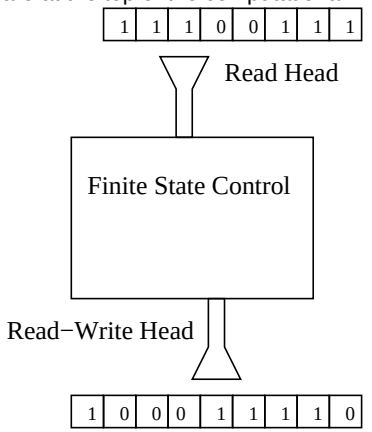
- Badii and Politi. Complexity: Hierarchical Structures and Scaling in Physics. Cambridge. 1997. *Excellent book, geared toward physics grad students. Closest thing to a textbook that covers topics similar to those I've covered throughout these lectures.*
- Bioinformatics textbooks?

Part VIII

A Very Brief and Informal Introduction to Computability and Computational Complexity

A Very Brief and Informal Introduction to Computability and Computational Complexity

- **Turing Machines** are at the top of the computational hierarchy.



- A TM has an input tape, a finite state controller, and a working tape. The finite state controller can read and write symbols from/to the working tape.

More about UTMs

- A universal TM is a TM that can simulate any other TM.
- A UTM is the most general model of computation. It can match the power of any other computational method.
- Thus, UTM's are equivalent to C programs, java programs, etc.
- TM's are also taken as being equivalent to algorithms.

More about UTMs, continued

- Recursively Enumerable Languages are recognized by UTMs
- However, there exists some languages that no machine can recognize! Why?
- The number of UTMs is countably infinite.
- The number of languages is uncountably infinite, since it is the set of all subsets of a countably infinite set.
- Thus # of Languages $>$ # of machines.

This has some profound and important implications. We can use this to show that some algorithms do not exist.

We will do so via a paradox

Paradoxes

Paradoxes are unavoidable with self-referencing systems. Examples:

- "I'm lying"
  1. If I'm lying, I'm telling the truth
  2. If I'm telling the truth, I'm lying
- The shortest integer that can't be described in less than thirteen words.
- Consider the set of all sets that are not members of themselves Should this set be a member of itself?
- etc.

Halting

- One of the problems with UTMs is that sometimes they don't halt. They can get caught in loops.
- Let's see if we can come up with a method for determining if a UTM, with program P and input x will halt.
- This slide and the following is almost directly taken from Randy Wang's lecture on computability: www.cs.princeton.edu/~rywang/99f126/slides.html.
- Call this program $\mathrm{HALT}(P, x)$. It takes as input the program P and the output x.
- $\mathrm{HALT}(P, x)$ returns YES if $P(x)$ halts, and NO if it doesn't.
- Will assume that $\mathrm{HALT}(P, x)$ always halts—it does not go into loops.

Halting Problem, continued

Now, let's construct the strange program $\mathrm{XX}(P)$, as follows:

- $\mathrm{XX}(P)$ calls $\mathrm{HALT}(P, P)$.
- $\mathrm{XX}(P)$ halts if $\mathrm{HALT}(P, P)$ outputs NO.
- $\mathrm{XX}(P)$ infinite loops if $\mathrm{HALT}(P, P)$ outputs YES

In other words:

- If $P(P)$ does not halt, $XX(P)$ halts.
- If $P(P)$ halts, $XX(P)$ does not halt.

Let's call XX with *itself* as input

- If $XX(XX)$ does not halt, $XX(XX)$ halts
- If $XX(XX)$ halts, $XX(XX)$ does not halt

Both lead to a contradiction. Therefore, $\mathrm{HALT}(P, x)$ **cannot exist**.

Consequences of Halting Problem

- There does not exist an algorithm to determine if a UTM running program P with input x will halt.
- We say that the Halting problem is **uncomputable** or unsolvable.
- It turns out that many other problems are uncomputable as well.
- Of particular relevance to us: There does not exist an algorithm that will determine the shortest program that will output a given string.
- I.e., there is no general-purpose algorithm for optimal data compression.
- This means that measures of complexity or randomness based on minimal UTM representations are uncomputable.
- More generally, the existence of uncomputable problems means that we'll never be able to find algorithms for everything.

A Very Brief Discussion of Computational Complexity

- The computational complexity of an algorithm is the run time $T(N)$ needed for the algorithm to run, expressed as a function of the size of the problem N.
- The slower $T(N)$ grows with N, the more tractable (and less complex?) the problem is.
- For applications to physics, see, e.g., the work of Machta `www-unix.oit.umass.edu/~machta/`, and Moore, "Computational Complexity in Physics," `www.santafe.edu/~moore/pubs/nato.html`.
- Computational Complexity is usually concerned with the time scaling of the *worst* cast scenario. The time scaling of the average case may be more relevant. Unlike comp complexity, information theory is concerned with *average* case behavior.
- There has recently been work in applying parallel computational complexity to physical models. This may be more relevant—nature is often parallel.