What is Computation?
\[ \rho \left( \frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} \right) = -\nabla p + \nabla \cdot \mathbf{T} + \mathbf{f}, \]
Figure 1.24: Examples of plant-like structures generated by bracketed OL-systems. L-systems (a), (b) and (c) are edge-rewriting, while (d), (e) and (f) are node-rewriting.

n=5, δ=25°

F → F+[F]F[-F]F

n=5, δ=20°

F → F+[F]F[-F][F]

n=4, δ=22.5°

F → F-[F-X]+[+F-F-F]

n=7, δ=20°

F → F+[X]F[-X]+X

n=7, δ=25.7°

X → F[+X]F[-X]FX

n=5, δ=22.5°

X → F-[X]+X+F[+FX]-X

F → FF
Here parameters \( h \) and \( W \) describe the somatic impact on the firing rates from connecting neurons within and ing post-synaptic potentials (PSPs), which in turn depend on the current membrane potential; for instance, if \( \text{exc} \) and \( \text{inh} \) refers to excitatory and inhibitory neurons, respectively.

The dynamics of \( h \) depends on the current membrane potential; for instance, if \( h_t = \Psi [h(t)] \cdot I_t \).

\[
\frac{\hat{h}}{\partial t} = h_e - h_t + \sum_{l=e,i} \Psi [h(t)] \cdot I_l(t)
\]

or

“What do I want for lunch?”
Languages are *all about* computation!
The Chomsky Hierarchy, equally familiar to linguists and theoretical computer scientists.
\[ \tau_{xy} \leftarrow (1 - \rho) \tau_{xy} + \sum_k \Delta \tau_{xy}^k \]

\[ \Delta \tau_{xy}^k = \begin{cases} 
Q/L_k & \text{if ant } k \text{ uses curve } xy \text{ in its tour} \\
0 & \text{otherwise} 
\end{cases} \]
Definition 0.1  A frameshift machine is a five-tuple \( M = (\Sigma, \Gamma, \tau, E, \delta) \) where

\[ \begin{align*}
\Sigma & \text{ is the finite input alphabet}, \\
\Gamma & \text{ is the finite output alphabet}, \\
\tau & \in \mathbb{N} \text{ is the frame size}, \\
E & \subseteq \Sigma^\tau \text{ is the set of end frames}, \\
\delta & \subseteq \Sigma^* \times \Sigma^\tau \times \Sigma^* \times \Gamma \times \mathbb{Z}(\tau - 1), \\
\delta & \text{ finite, is the transition relation.}
\end{align*} \]
So what?
If we model processes (like translation) as computations...

**Proposition 0.1**  Let $x = ax'b, x \in \Gamma_{AA}^+$ with $a, b \in \Gamma$. Then

$$(w, \lambda, 1) \vdash_{M_{GEN}}^* (w, ax'b, k),$$

$$(v, \lambda, 1) \vdash_{M_{GEN}}^* (v, ax'b, l),$$

implies $w(2) \cdots w(k + 1) = v(2) \cdots v(l + 1)$.

...then we can prove theorems about them.
Algorithm 1  Determine prefix and dual RNA regular expressions

input: $\alpha = (a_1, b_1) \cdots (a_n, b_n), a_i, b_j \in \Gamma_{AA} \cup \{\lambda, \$, \}, M_{\text{duo}} = (Q, \hat{\Gamma}_{AA}, F, q_0, \delta)$.
output: $i$, the longest prefix such that $(a_1, b_1) \cdots (a_i, b_i) \in L_{\text{duo}}$, regular expressions $\bar{w}, \bar{v}$ for all possible $w, v$ of minimal length corresponding to $(a_1, b_1) \cdots (a_i, b_i)$ in equations (1), (2), (3), (4) of Proposition 0.3.

$i \leftarrow 1$
\text{inDuo} \leftarrow \text{true} \; //\text{this will be true if the prefix of length } i \text{ is in } L_{\text{duo}}$
$q \leftarrow 1 \; //\text{start state of } M_{\text{duo}}$
while $i \leq n$ and $\text{inDuo} = \text{true}$
  $q \leftarrow \delta(q, (a_i, b_i))$
  if $q$ is defined //true if $(a_1, b_1) \cdots (a_i, b_i) \in L_{\text{duo}}$
    if $(a_i, b_i) \leftarrow (r, e)$ and ($i = 1$ or $i - 1 \in \{(s, l), (l, s), (h, t), (c, v), (y, i), (a, r), (r, a), (f, l), (p, p), (g, g)\})$ then $\bar{w}(3(i - 1) + 1) \leftarrow \{A, \bar{C}\}$.
      //as discussed above.
    else let $\bar{w}(3(i - 1) + 1)$ be the unique first character of words in $\rho(a_i, b_i)$.
    if $(a_i, b_i) = (g, \$)$ and ($i = 1$ or $i - 1 \in \{(s, l), (l, s), (t, h), (v, c), (i, y), (a, r), (r, a), (e, f), (p, p), (g, g)\})$ then $\bar{v}(3(i - 1) + 1) \leftarrow \{A, C\}$.
      else let $\bar{v}(3(i - 1) + 1)$ be the unique first character of words in $\rho(b_i, a_i)$.
      let $\bar{w}(3(i - 1) + 2)$ and $\bar{w}(3(i - 1) + 3)$ be unique 2nd, 3rd chars of words in $\rho(a_i, b_i)$.
      let $\bar{v}(3(i - 1) + 2)$ and $\bar{v}(3(i - 1) + 3)$ be unique 2nd, 3rd chars of words in $\rho(b_i, a_i)$.
      $i++$
    else $\text{inDuo} \leftarrow \text{false}$.
  \text{let } \bar{w}(3(i - 1) + 4) \text{ be the set of all last ribonucleotides of strings in } \rho(a_i, b_i)$.
  \text{let } \bar{v}(3(i - 1) + 4) \text{ be the set of all last ribonucleotides of strings in } \rho(b_i, a_i)$.
output $i - 1, \bar{w}, \bar{v}$

... and develop new algorithms.
What is Computer Science?
Every process is a computation.

Computer Scientists study computation, not necessarily computers.
“Computer Science is no more about computers than astronomy is about telescopes”
I love that quote, because it’s true for me.

But for all of CS? It’s complete bullshit.

Some of us study *computation*, but some of us really do study telescopes. I mean “computers”.
What is a Computer?
Even if we restrict ourselves to human-made, “artificial” computers, there’s more than you might think...
\[ a |000\rangle + b |001\rangle + c |010\rangle + d |011\rangle + e |100\rangle + f |101\rangle + g |110\rangle + h |111\rangle \]
A. Explanation of state and symbol encoding

<table>
<thead>
<tr>
<th>Symbol</th>
<th>a</th>
<th>b</th>
<th>terminator (t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>encodings &amp; &lt;state, symbol&gt; sticky ends</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;S0, a&gt;</td>
<td>TGGCT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;S0, b&gt;</td>
<td>CGAGG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;S1, a&gt;</td>
<td>GGC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;S1, b&gt;</td>
<td>CTTAC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

B. Hardware

9 nt

13 nt

*FokI enzyme & recognition site*

C. Software

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>T1: SO→S0</td>
<td>T2: SO→S1</td>
<td>T3: SO→S0</td>
</tr>
<tr>
<td>T4: SO→S1</td>
<td>T5: S1→S0</td>
<td>T6: S1→S1</td>
</tr>
<tr>
<td>T7: S1→S0</td>
<td>T8: S1→S1</td>
<td></td>
</tr>
</tbody>
</table>

D. Input

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;S0, a&gt;</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>&lt;S0, b&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>terminator</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

E. Computation through symbol cleavage and scatter

Hybridization of the Hardware-Software complex to the input and cleavage of next symbol exposing next state-symbol combination

Hardware-Software complex

Input

Hardware-Software-Input complex

Recycled HW-SW complex

Remaining input

Cleaved and scattered input symbol

Output

The cascade proceeds until the terminator is cleaved and an output is formed or until suspension.