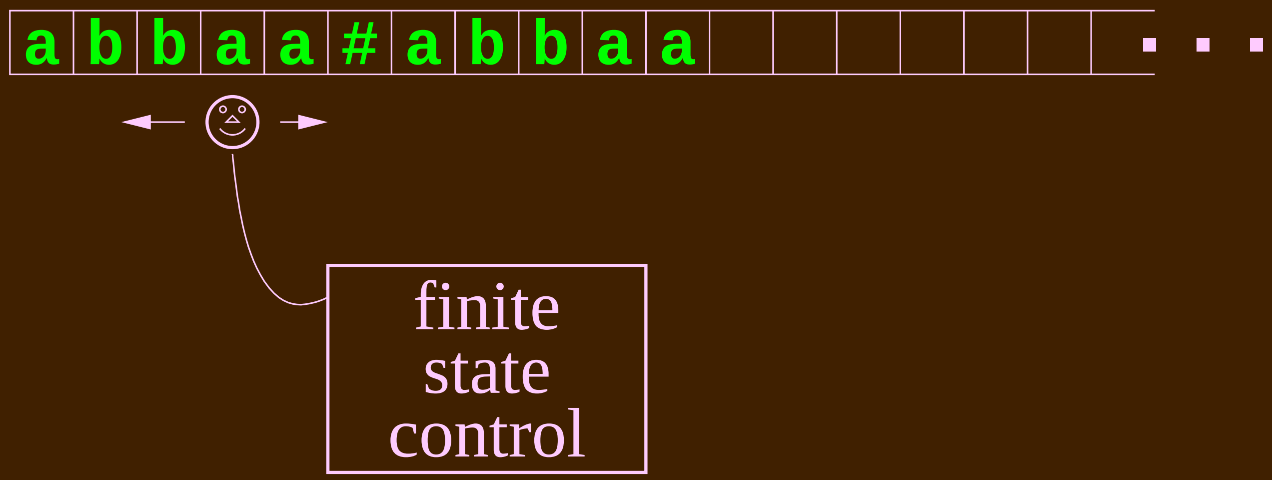
Turing Machines

Mark Greenstreet, CpSc 421, Term 1, 2006/07

- A Turing Machine Example
- Formal Definition
- More Examples

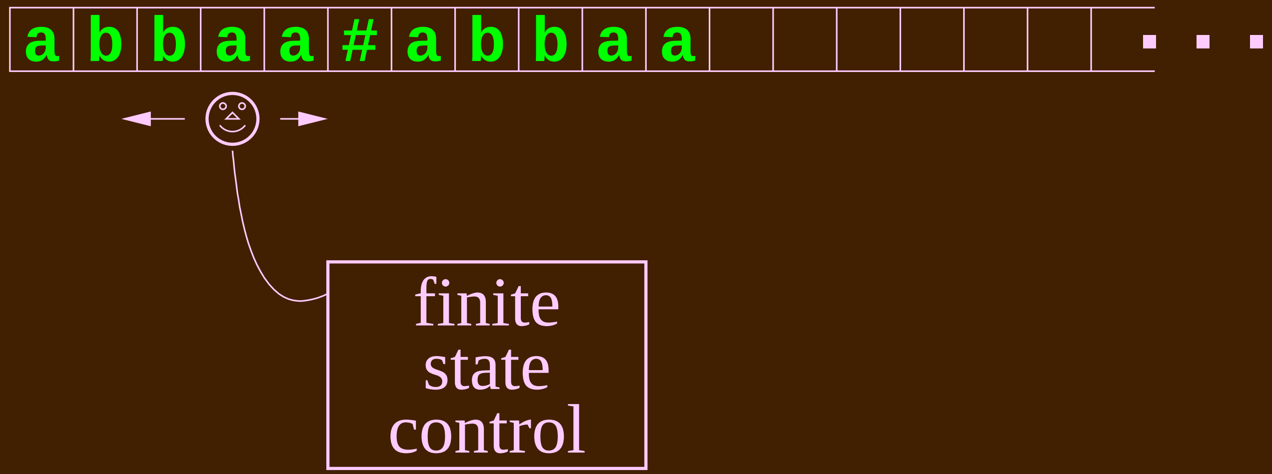
A Turing Machine



A Turing Machine has

- A tape that extends infinitely to the right.
- A finite state control: based on the current state and current input symbol:
- The controller has two special states:

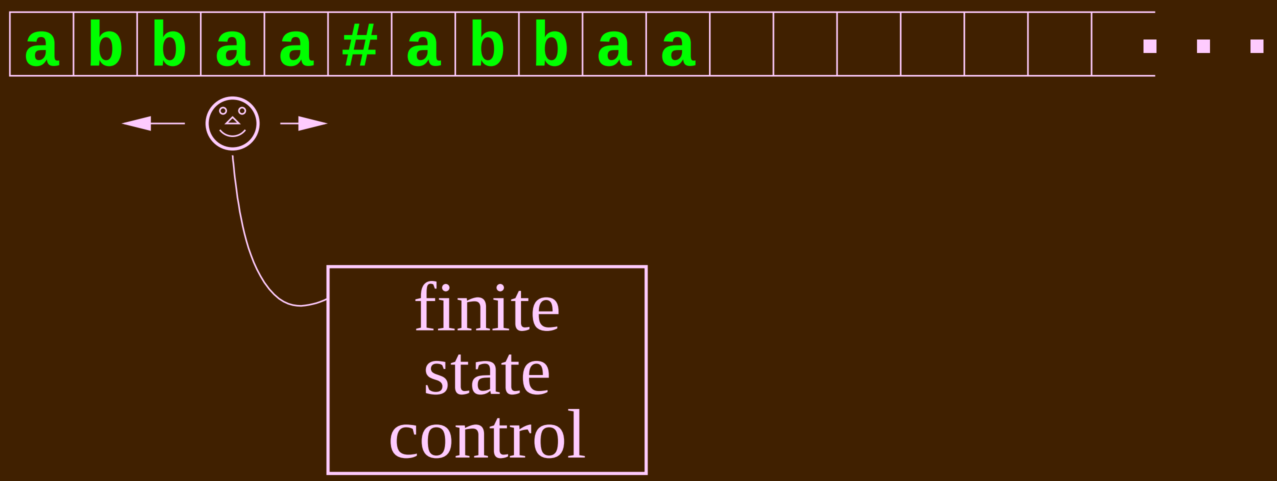
A Turing Machine



A Turing Machine has

- A tape that extends infinitely to the right.
  - The input string is written on the left end of the tape.
  - The rest of the tape is filled with blanks.
- A finite state control: based on the current state and current input symbol:
- The controller has two special states:

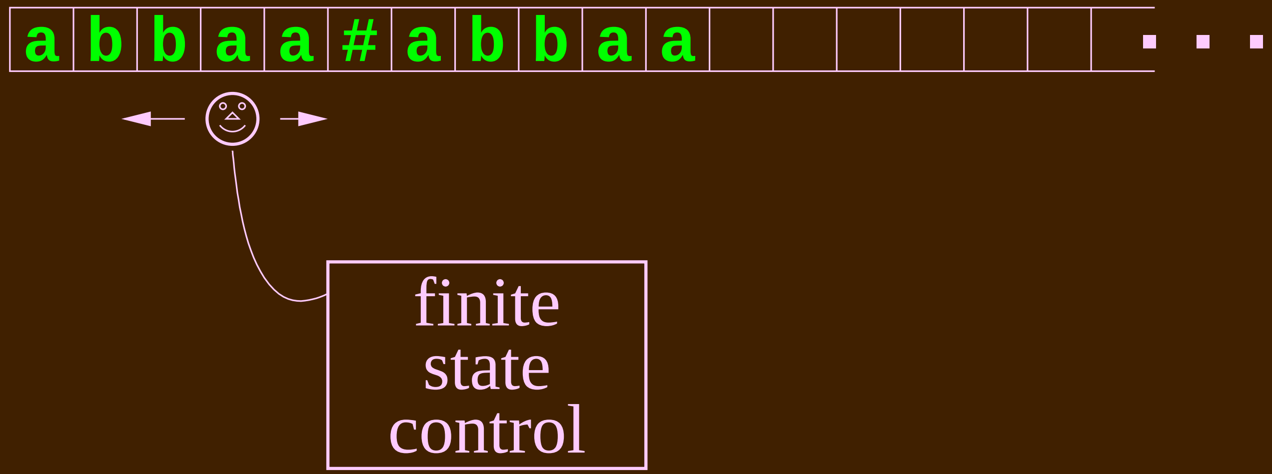
A Turing Machine



A Turing Machine has

- A tape that extends infinitely to the right.
- A finite state control: based on the current state and current input symbol:
  - The machine writes a new symbol on the current tape square (replacing the symbol it just read).
  - Moves the head one square to the left or right.
  - Enters a new state.
  - The rest of the tape is filled with blanks.
- The controller has two special states:

A Turing Machine



A Turing Machine has

- A tape that extends infinitely to the right.
- A finite state control: based on the current state and current input symbol:
- The controller has two special states:
  - `accept`: If the machine ever enters this state, it halts and the input string is accepted.
  - `reject`: If the machine ever enters this state, it halts and the input string is rejected.

$A = w\#w$ is not a CFL

- $\Sigma = \{0, 1, \#\}$.
- Let p be a proposed pumping lemma constant for A.
- Let $s = 0^p1^p\#0^p1^p$.
- Let u, v, x, y, and z be any strings such that: $s = uvxyz$;
- $|vy| > 0$; and
- $|vxy| \leq p$.
- Consider four cases:
- There is no way to break up s that satisfies the conditions of the pumping lemma.
- $\therefore A$ is not a CFL.

$A = w\#w$ is not a CFL

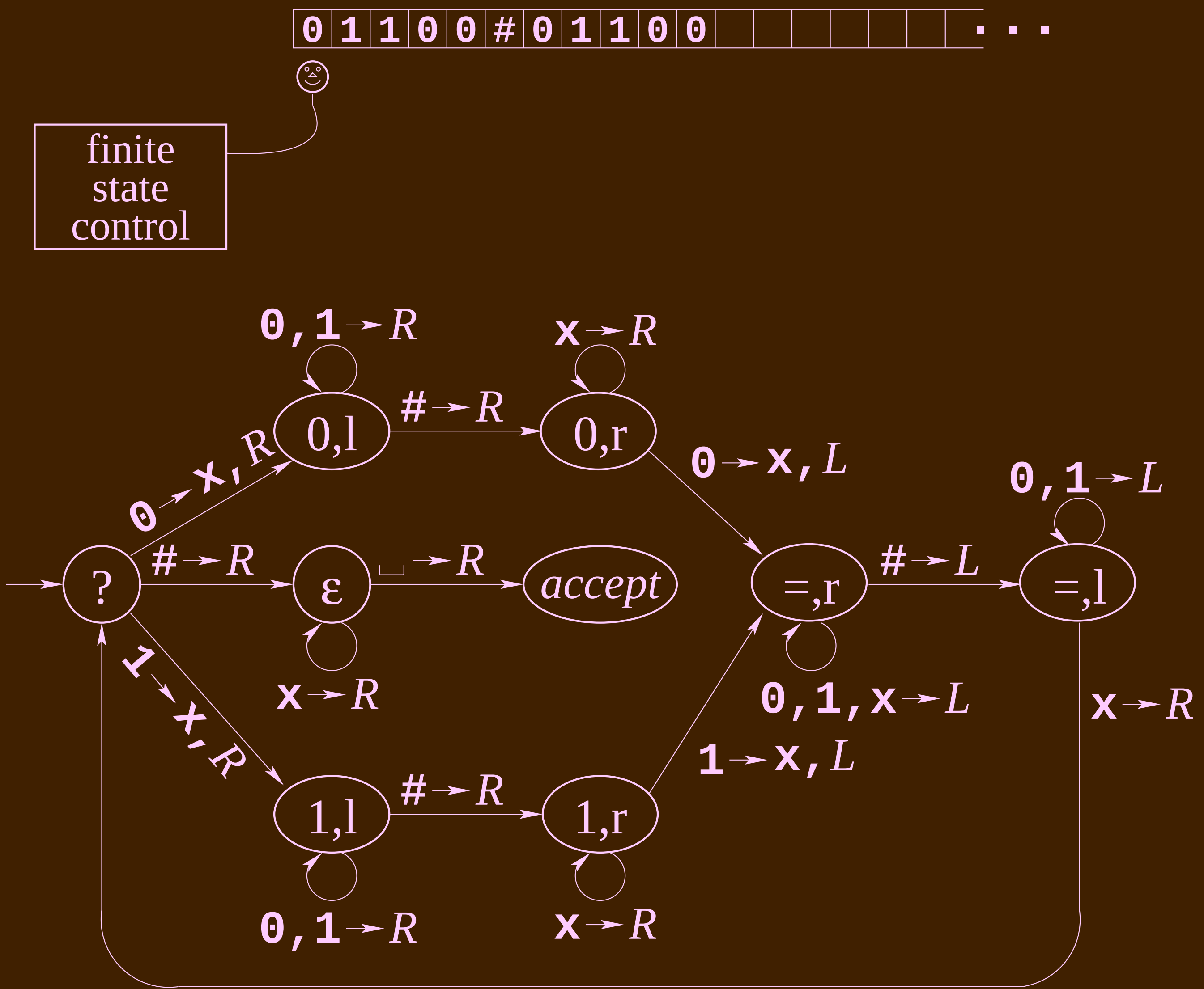
- $\Sigma = \{0, 1, \#\}$.
- Let p be a proposed pumping lemma constant for A.
- Let $s = 0^p 1^p \# 0^p 1^p$.
- Let u, v, x, y, and z be any strings such that: $s = uvxyz$;
- $|vy| > 0$; and
- $|vxy| \leq p$.
- Consider four cases:
- There is no way to break up s that satisfies the conditions of the pumping lemma.
- $\therefore A$ is not a CFL.

$A = w\#w$ is not a CFL

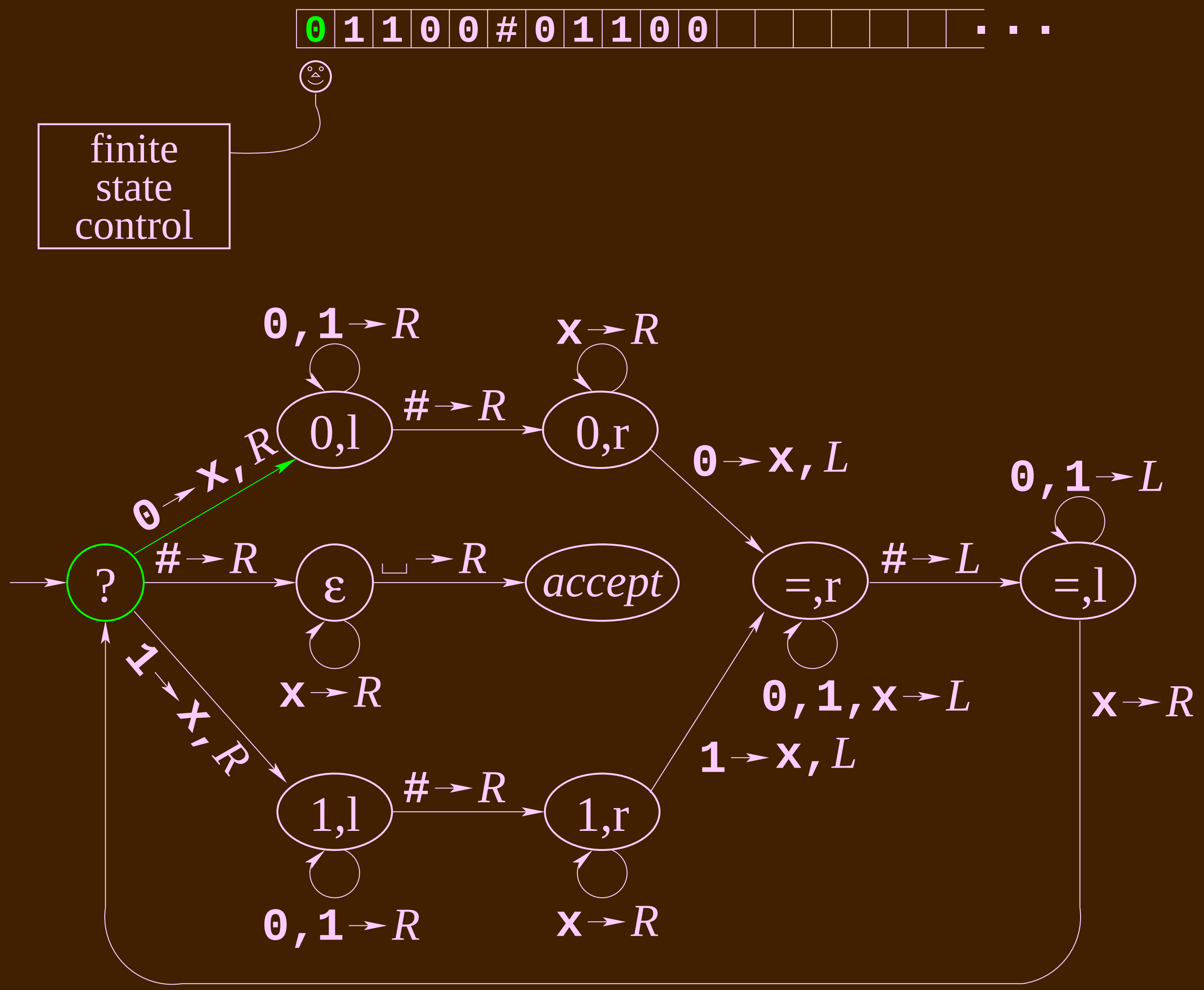
- $\Sigma = \{0, 1, \#\}$.
- Let p be a proposed pumping lemma constant for A.
- Let $s = 0^p1^p\#0^p1^p$.
- Let u, v, x, y, and z be any strings such that: $s = uvxyz$;
- $|vy| > 0$; and
- $|vxy| \leq p$.
- Consider four cases:
  - If vxy is contained in the initial 0^p1^p of s, then pumping it will create a string that is not in A.
  - Likewise, vxy can't be in the final 0^p1^p.
  - If vxy straddles the two halves of s, then it must be contained in the $1^p\#0^p$ part because $|vxy| \leq p$. This means that pumping p will change the number of 1's in the first half and the number of 1's in the second half. Again this produces strings that are not in A.
  - If $vxy = \#$ then pumping the string changes the number of $\#$'s and produces strings that are not in A.

- There is no way to break up s that satisfies the conditions of the pumping lemma.

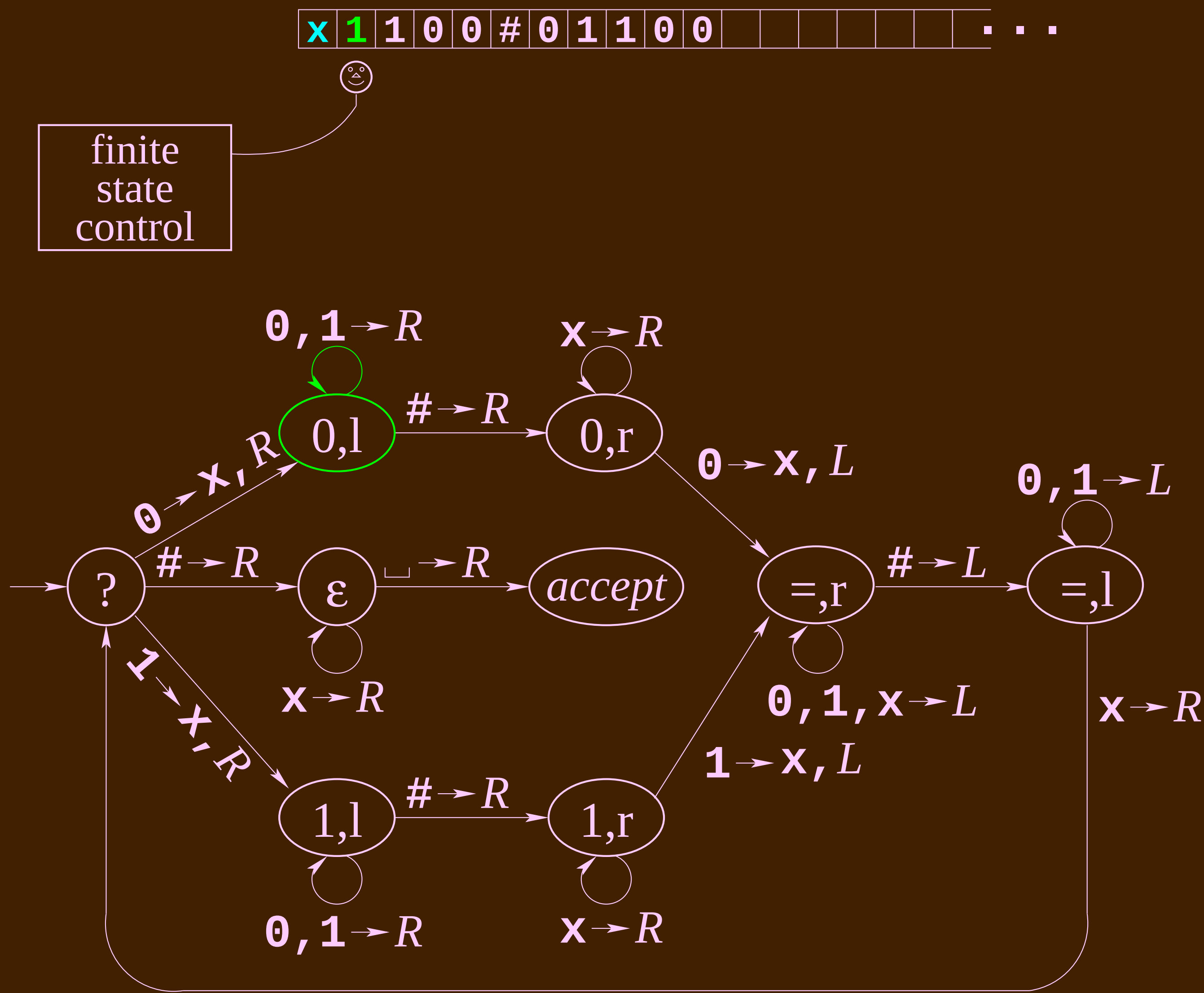
A Machine for $w\#w$



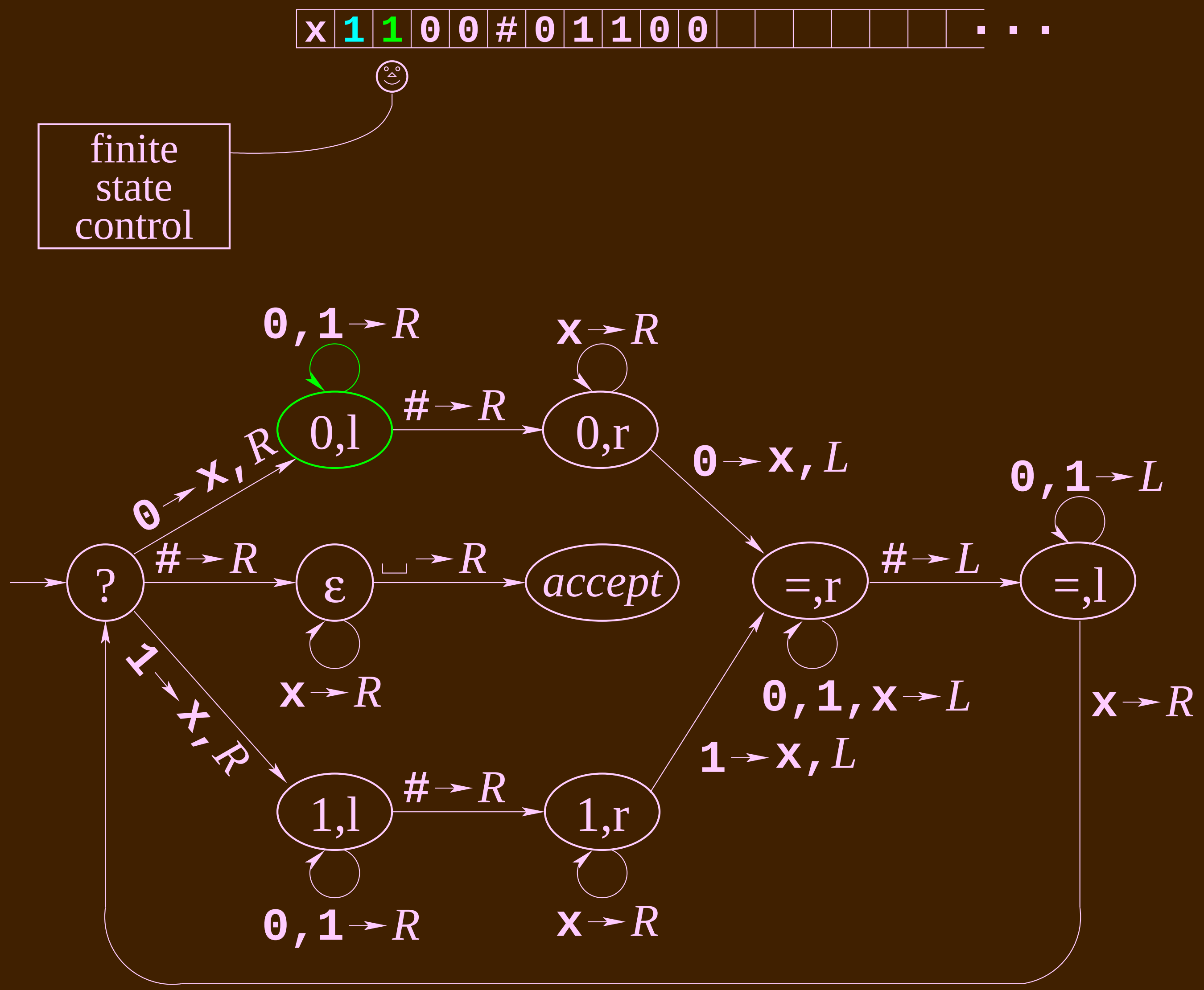
A Machine for $w\#w$



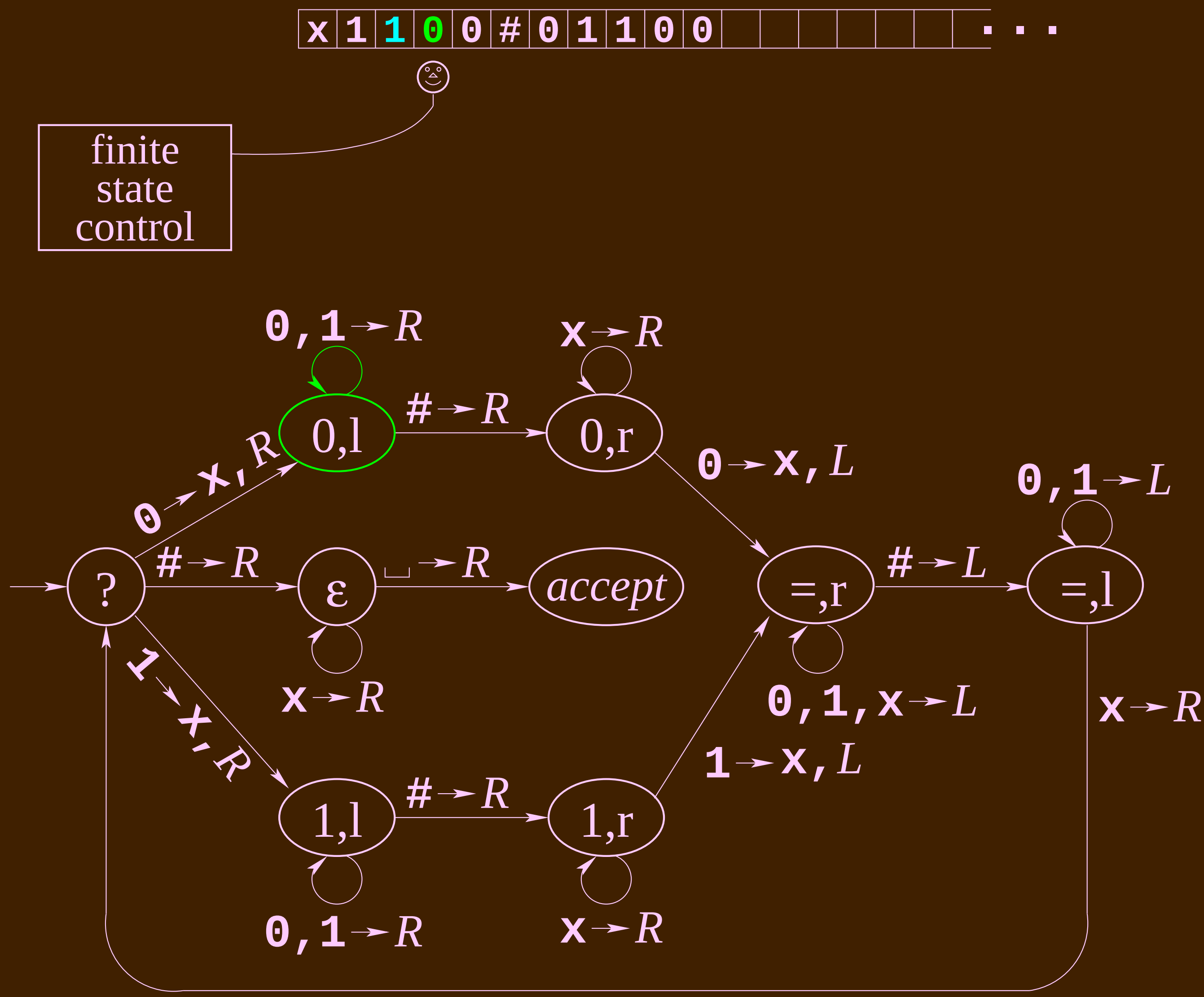
A Machine for $w\#w$



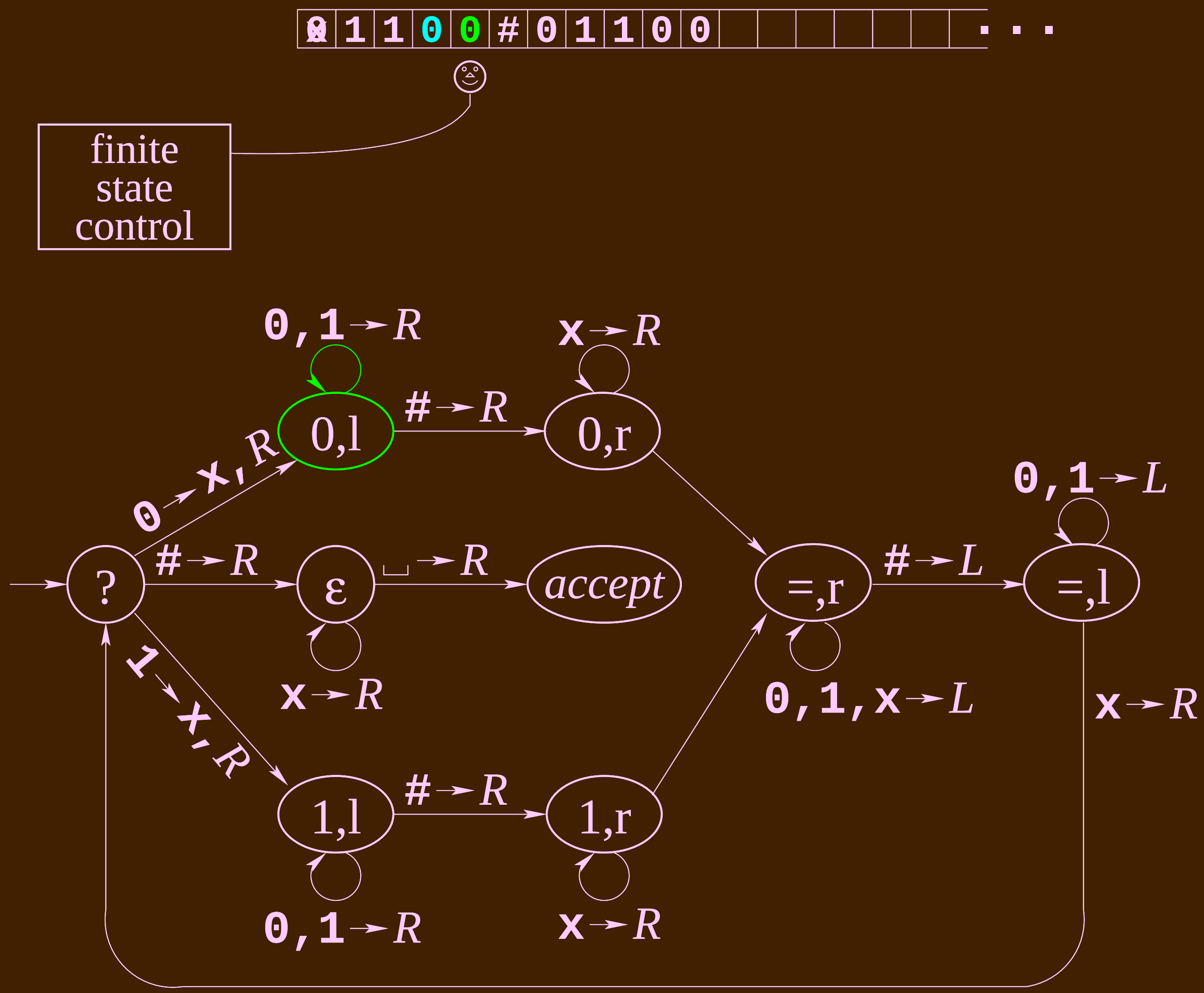
A Machine for $w\#w$



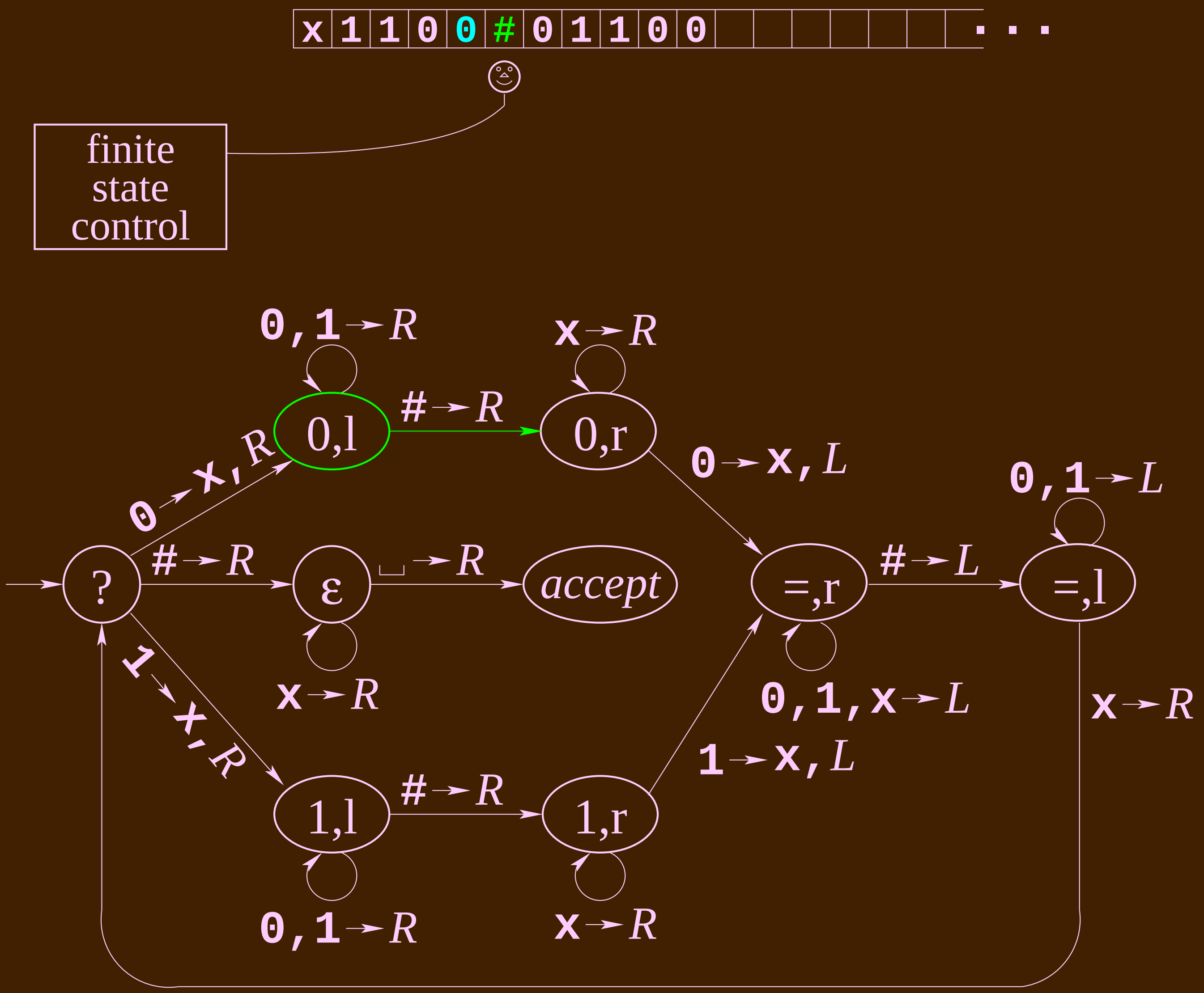
A Machine for $w\#w$



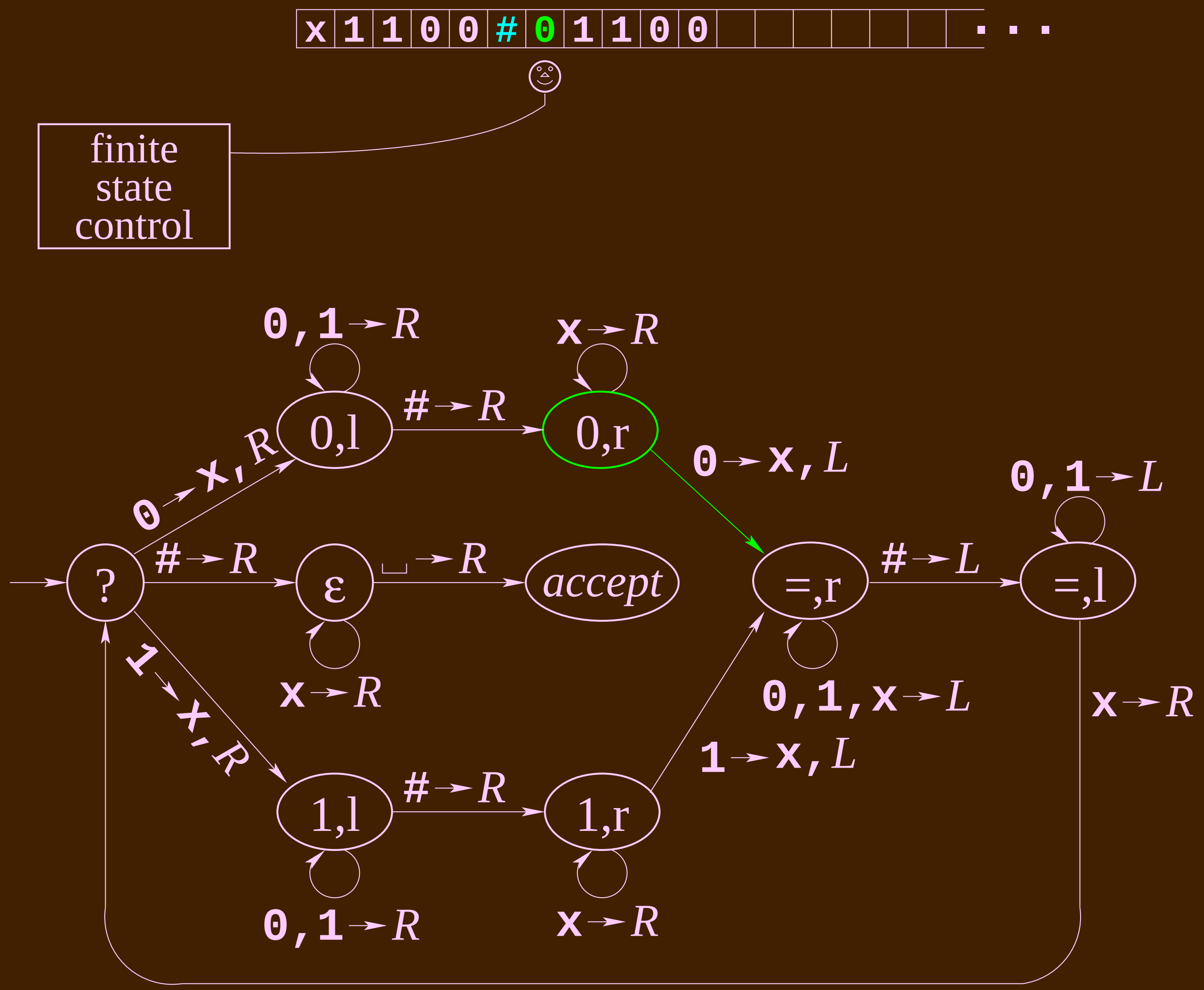
A Machine for $w\#w$



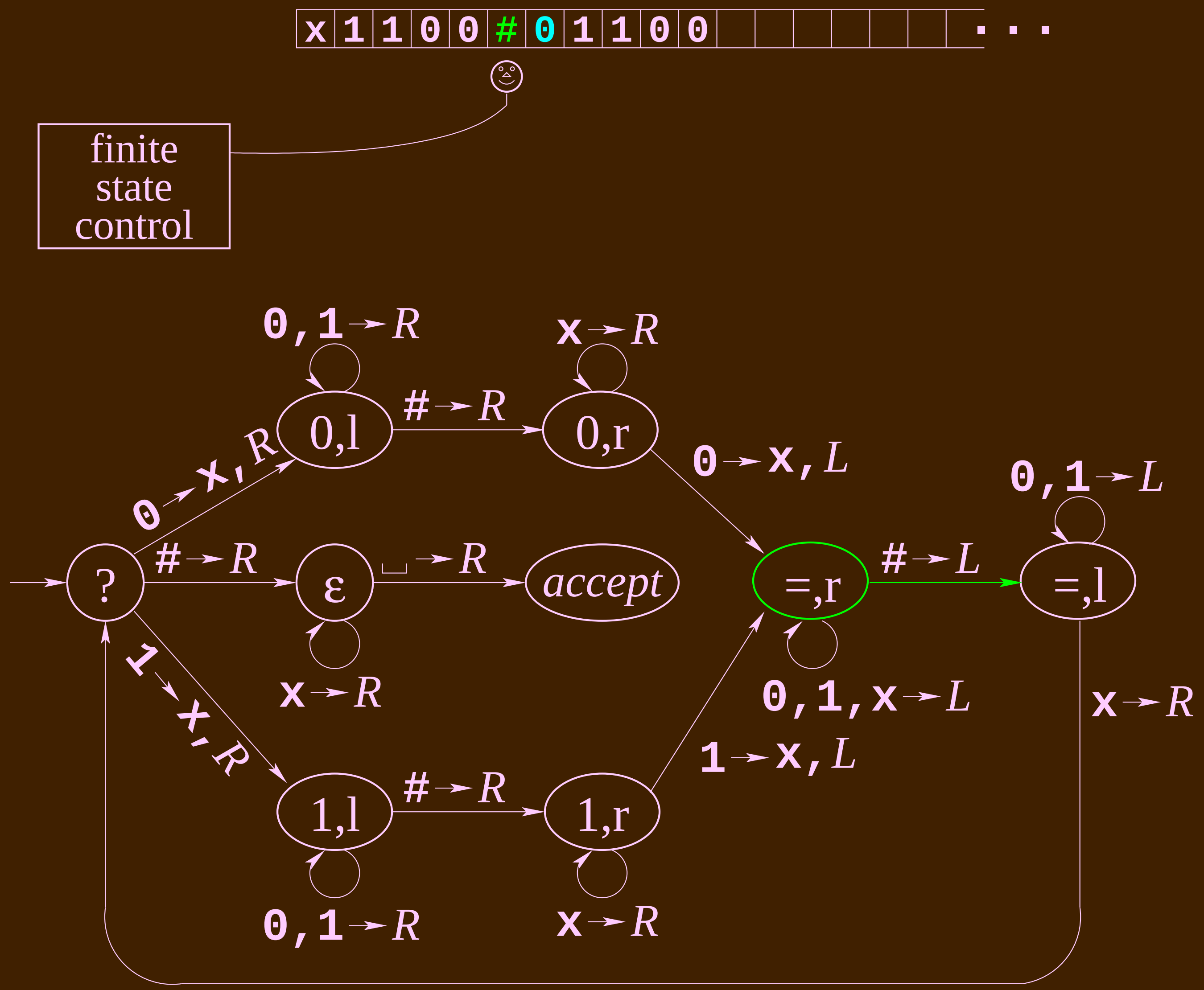
A Machine for $w\#w$



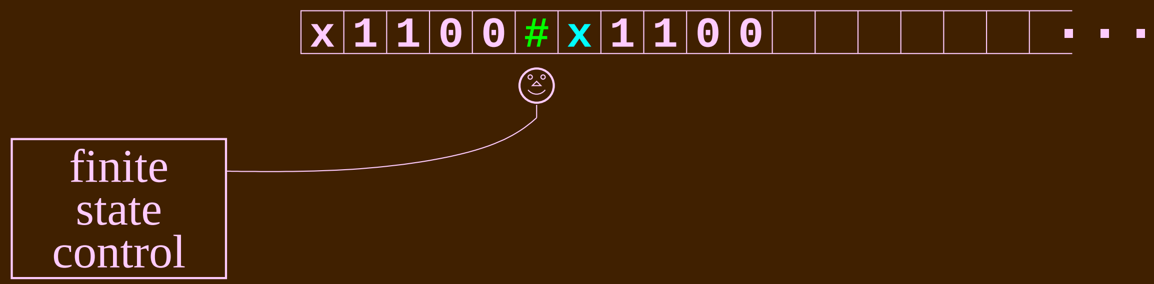
A Machine for $w\#w$

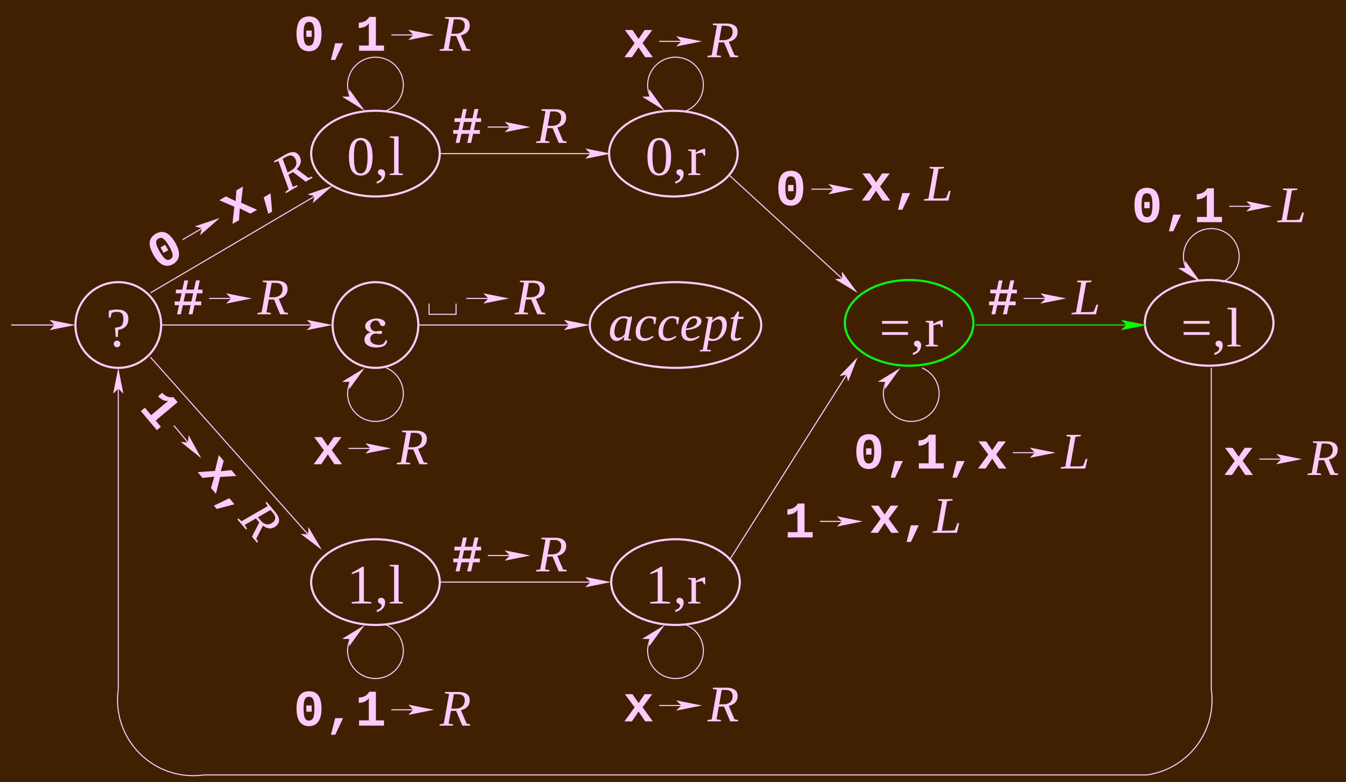


A Machine for $w\#w$

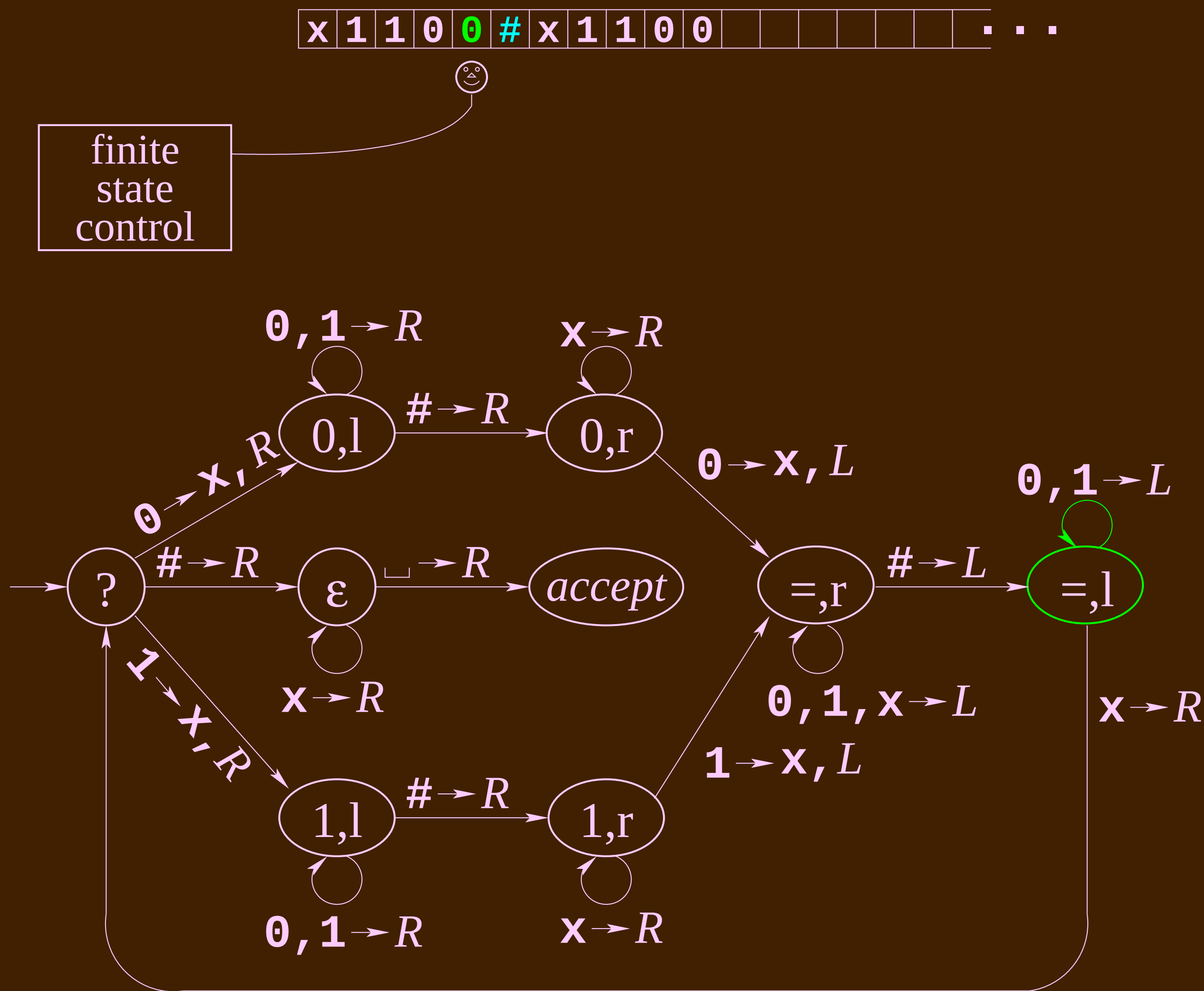


A Machine for $w\#w$

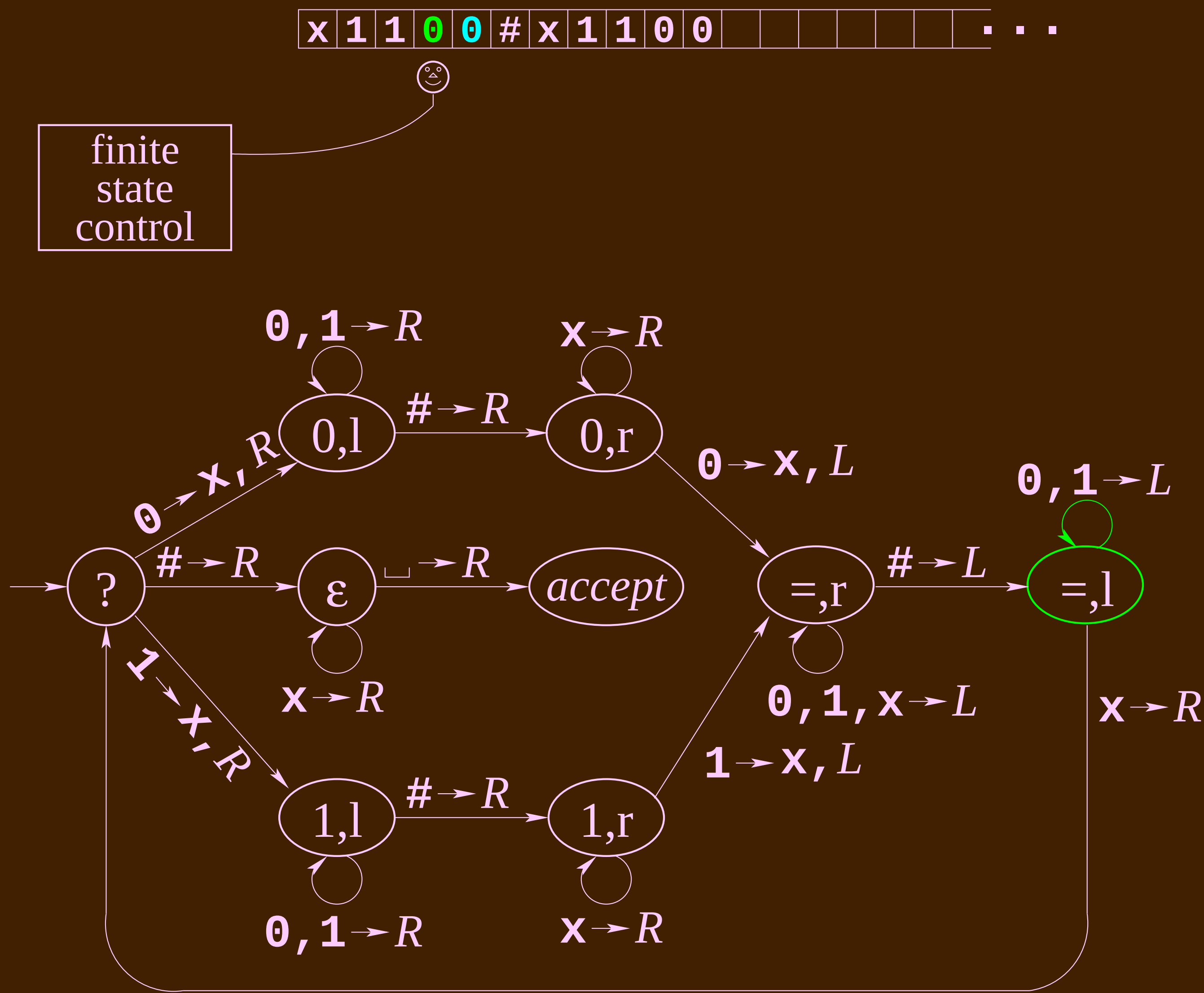




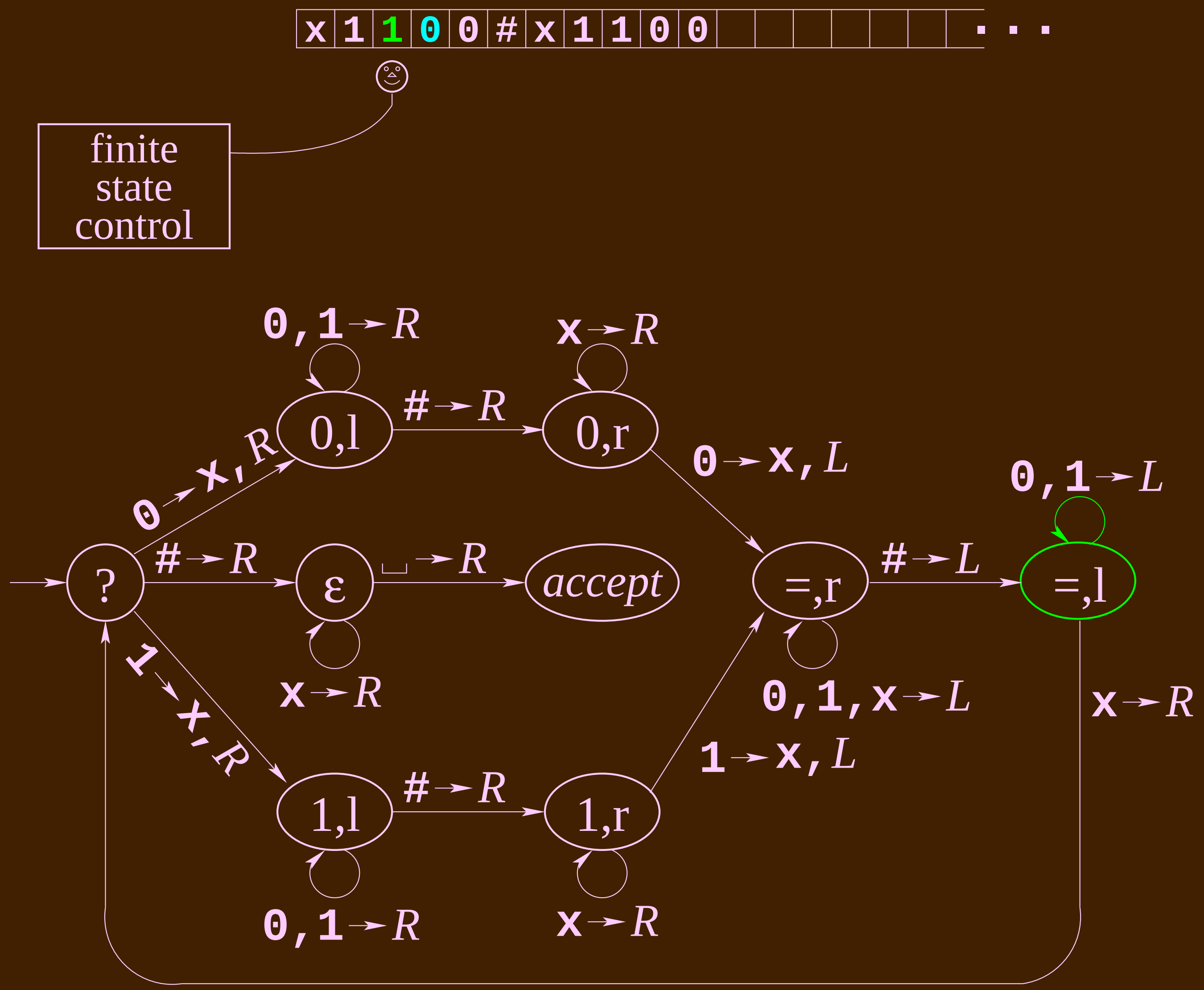
A Machine for $w\#w$



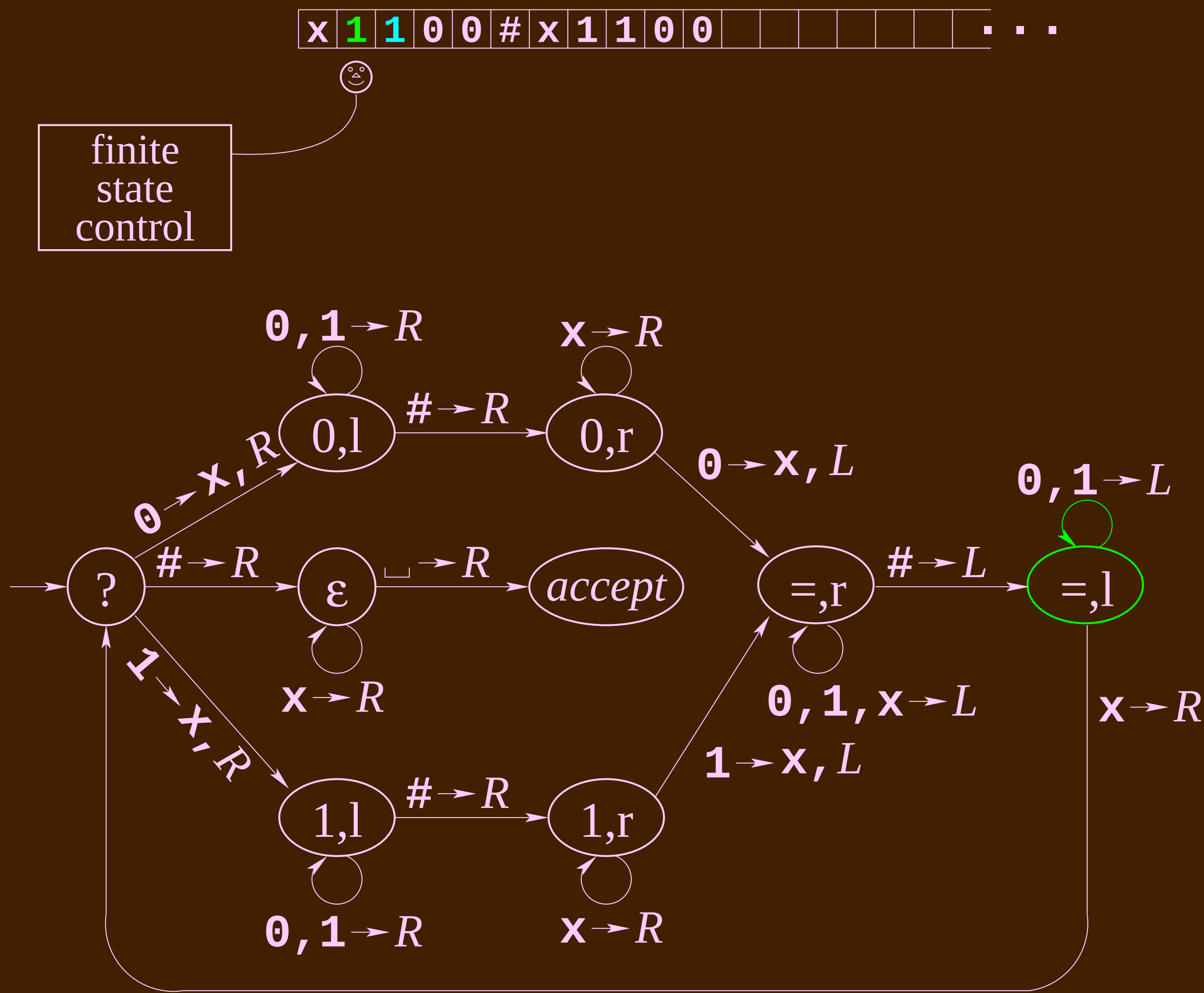
A Machine for $w\#w$



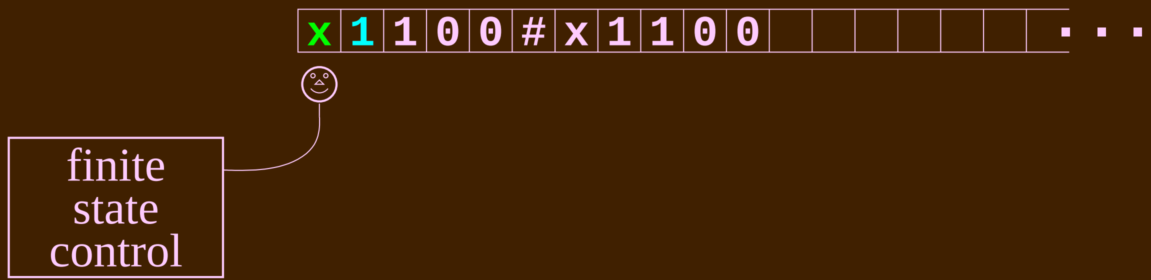
A Machine for $w\#w$

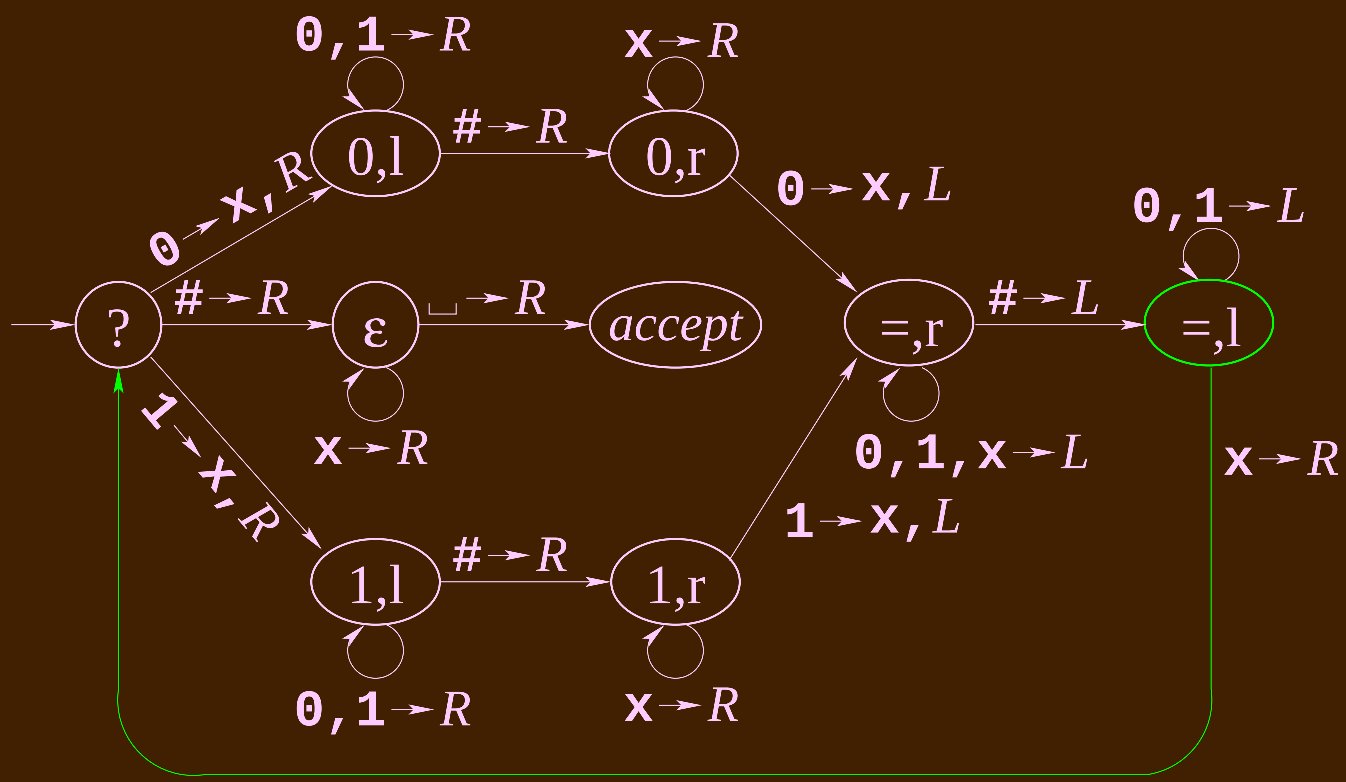


A Machine for $w\#w$

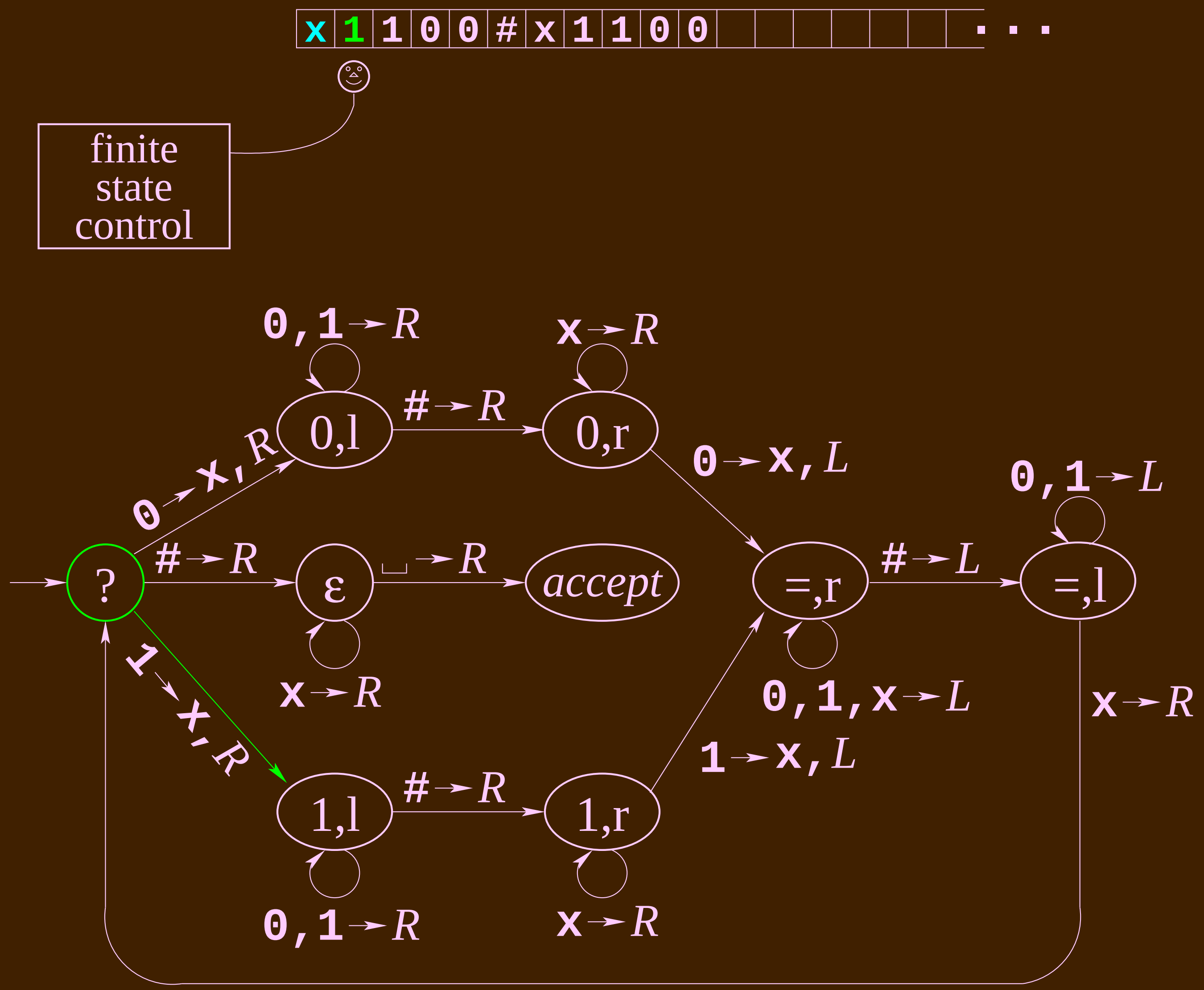


A Machine for $w\#w$

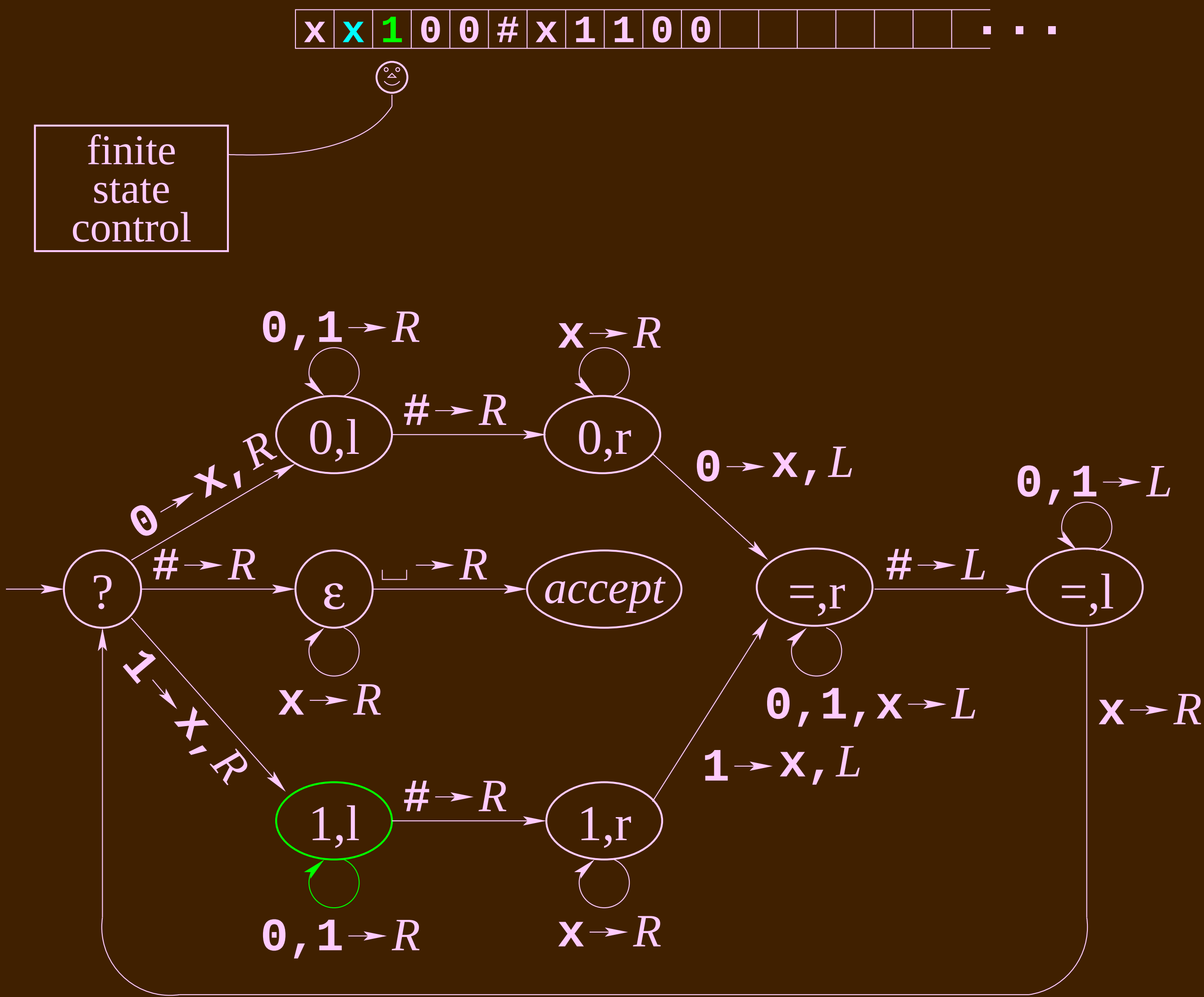




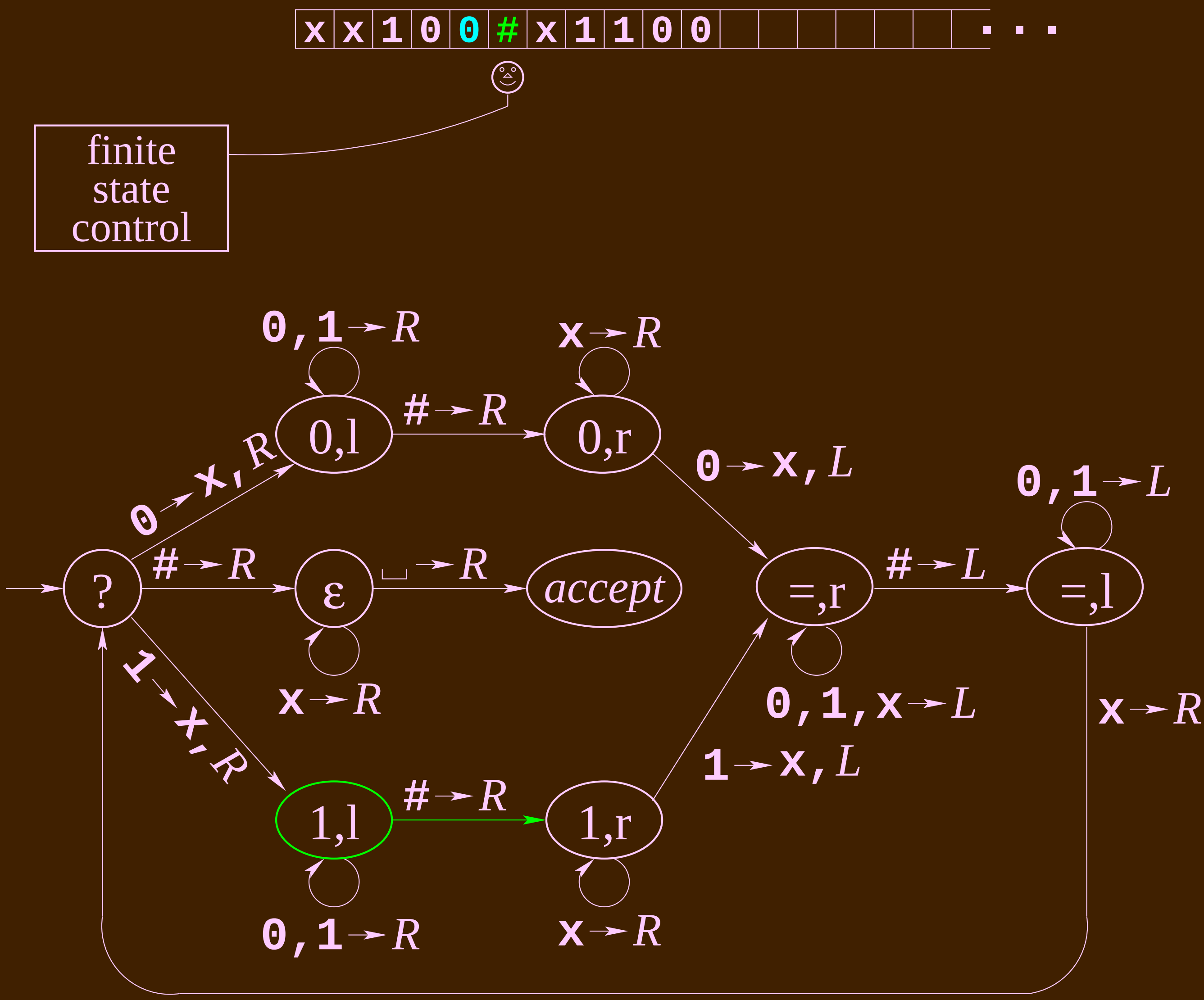
A Machine for $w\#w$



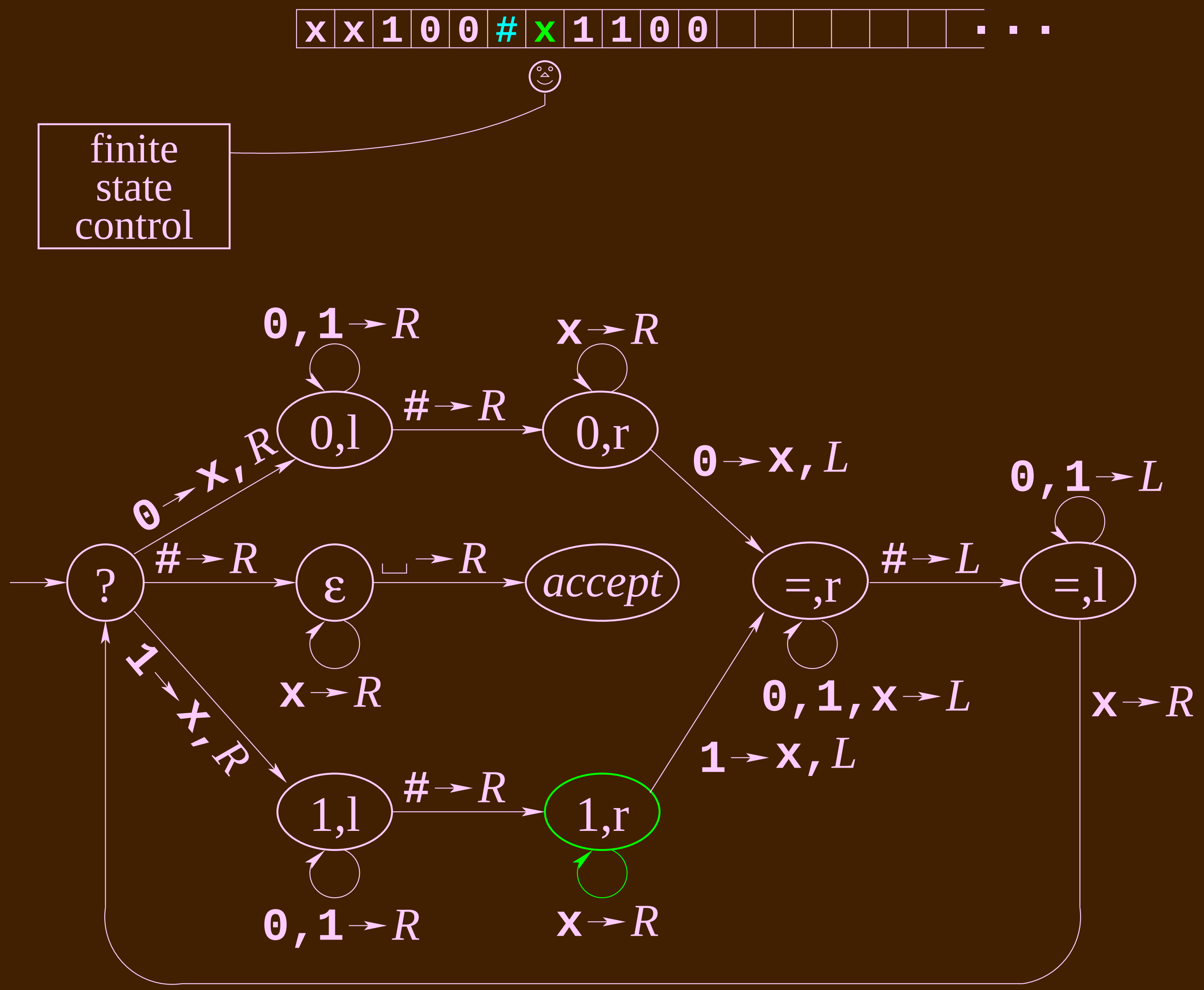
A Machine for $w\#w$



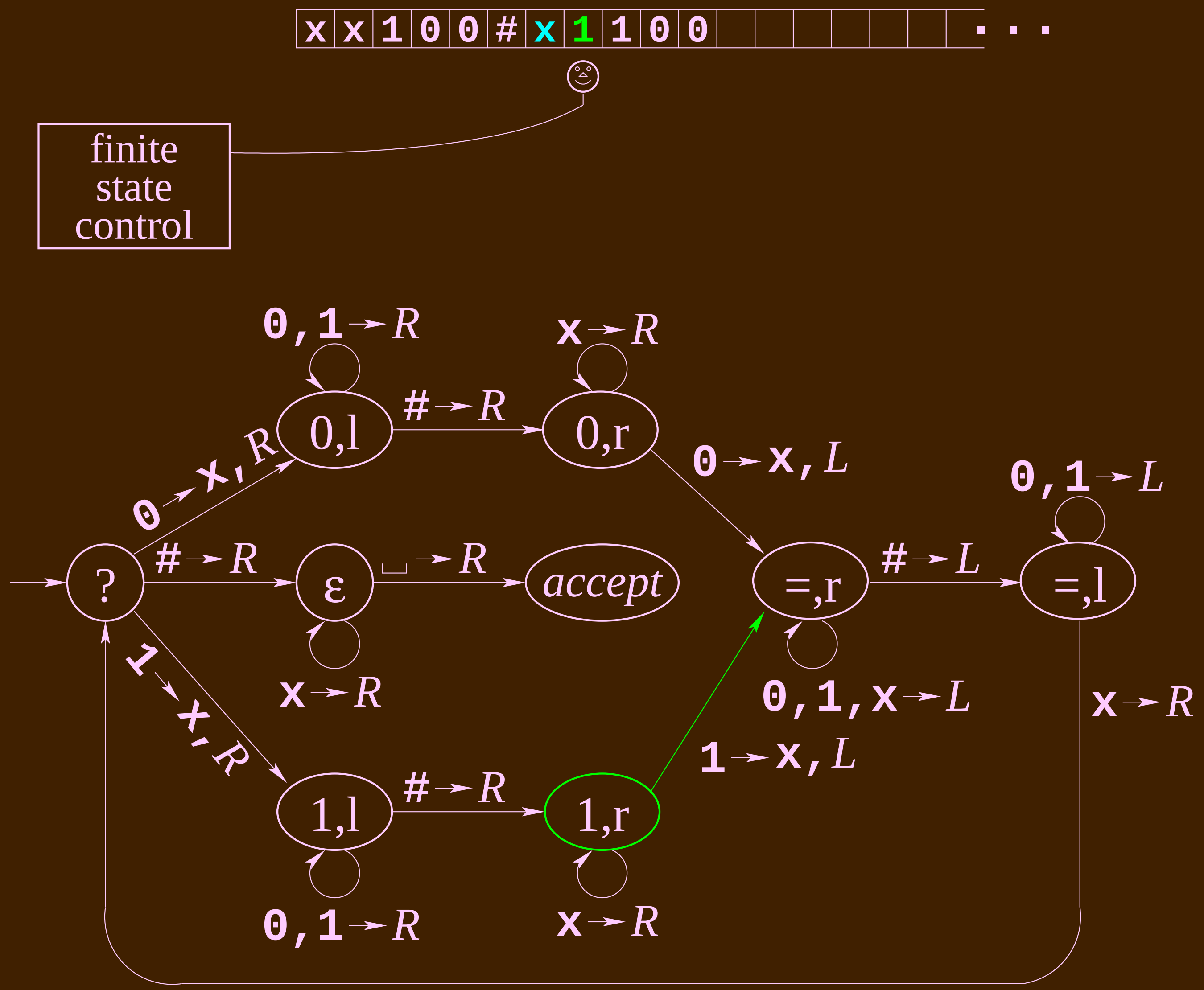
A Machine for $w\#w$



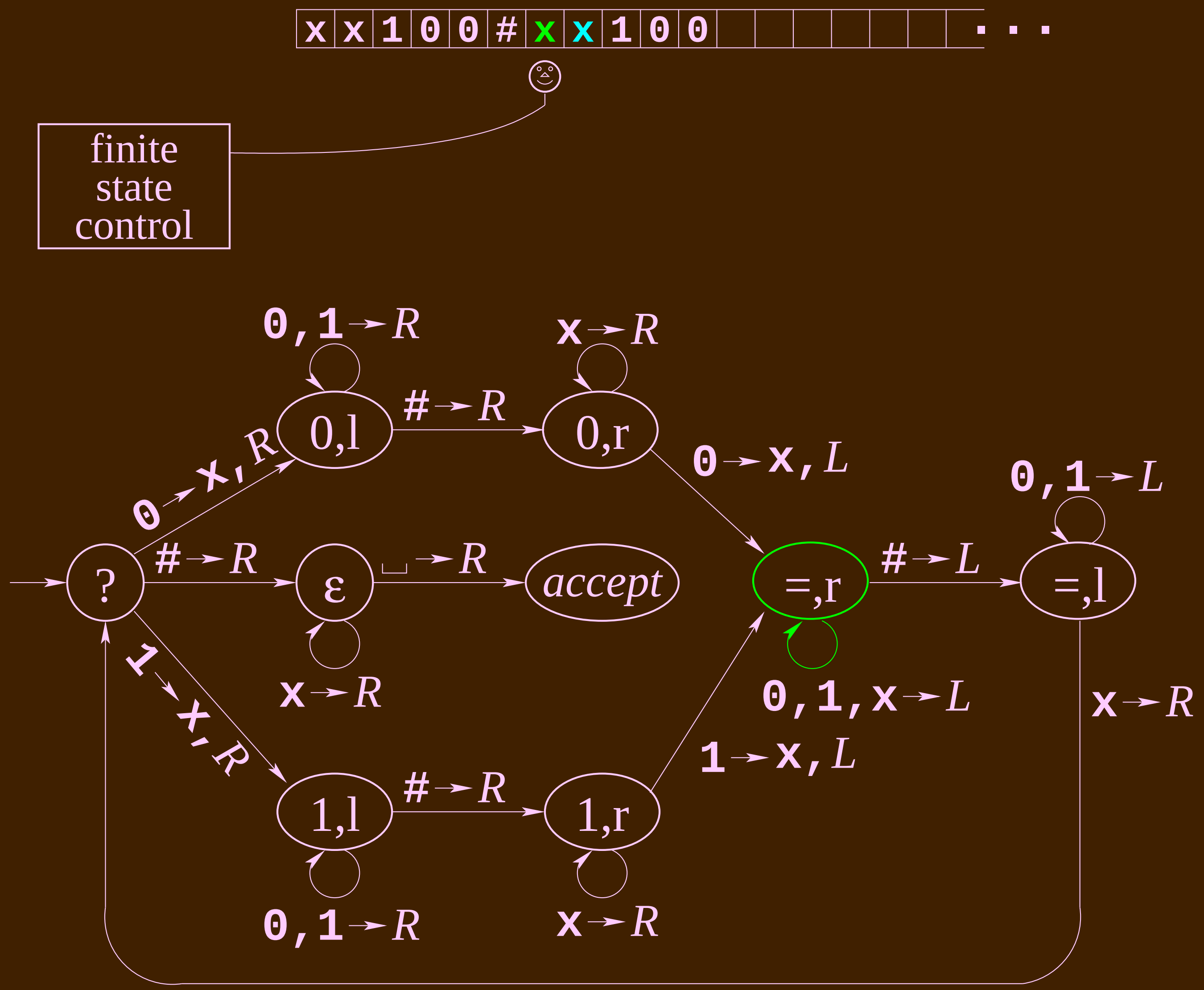
A Machine for $w\#w$



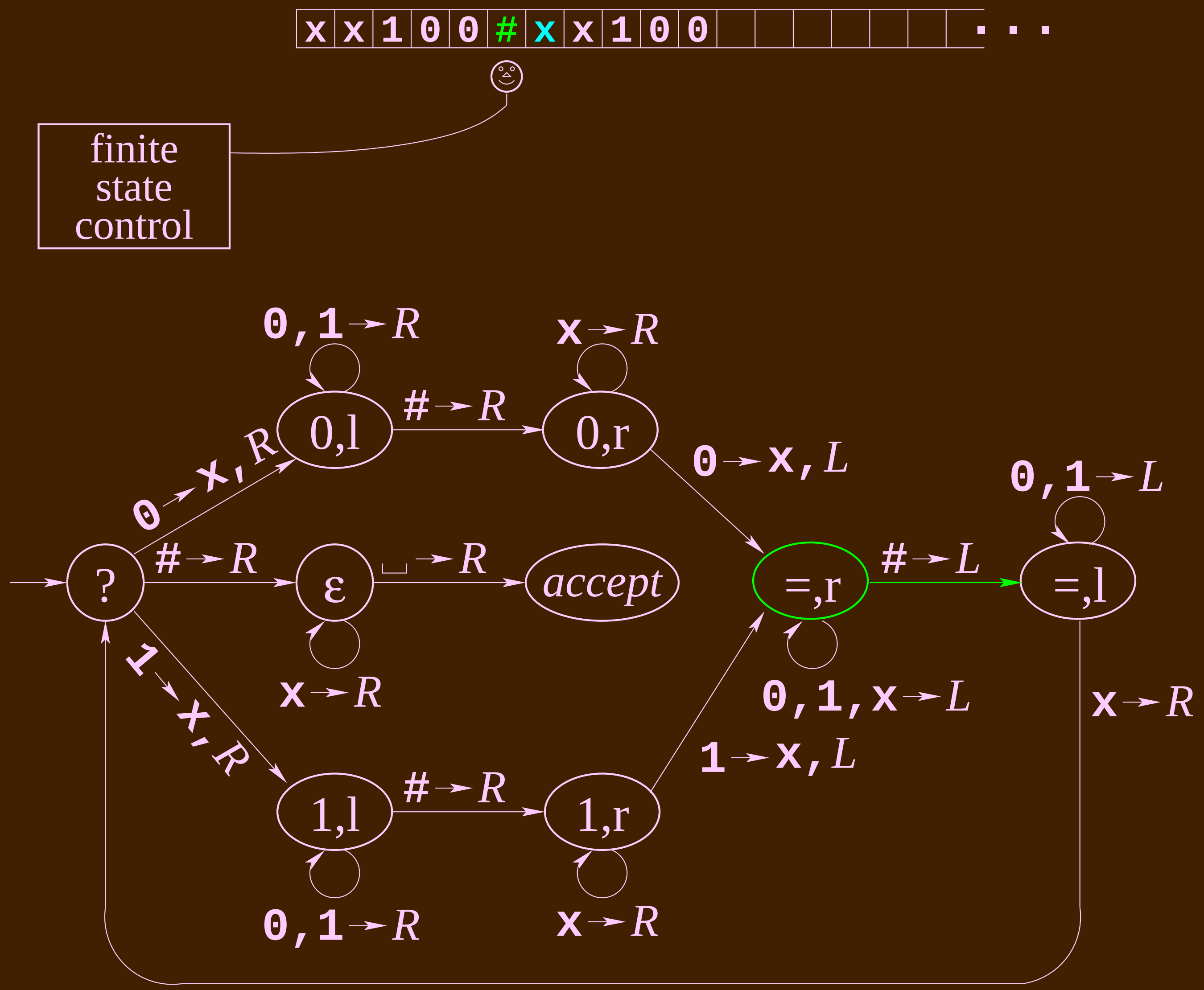
A Machine for $w\#w$



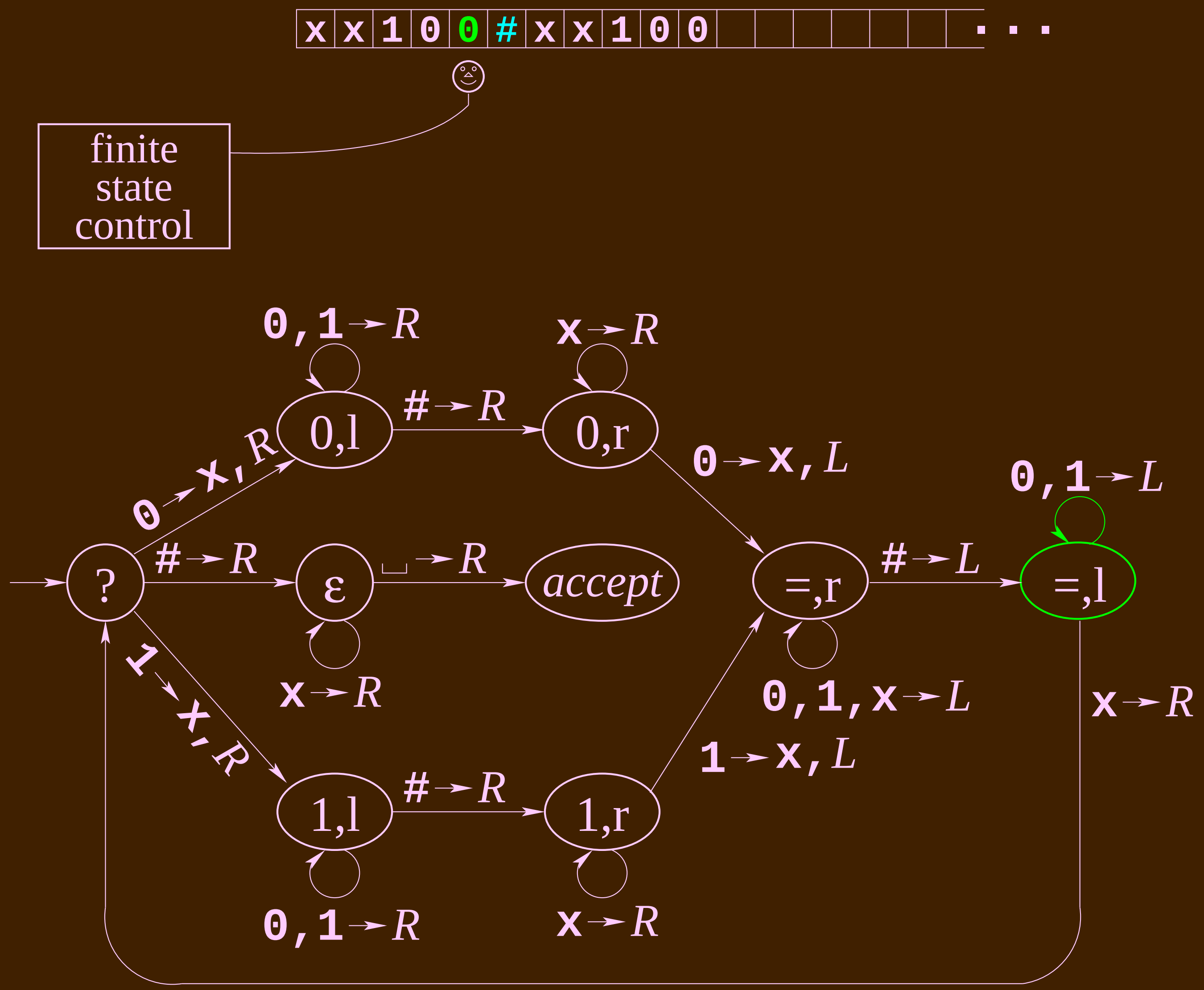
A Machine for $w\#w$



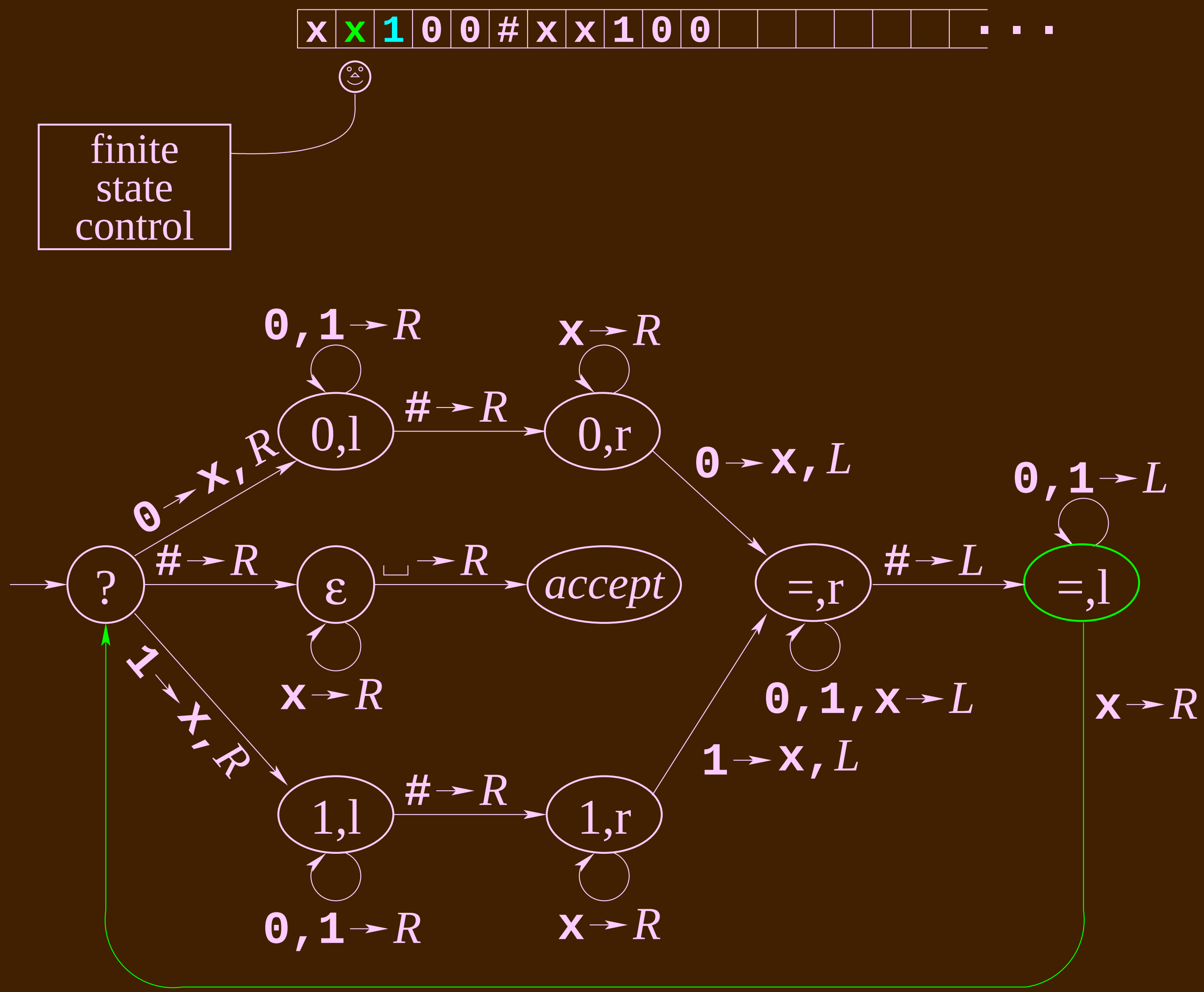
A Machine for $w\#w$



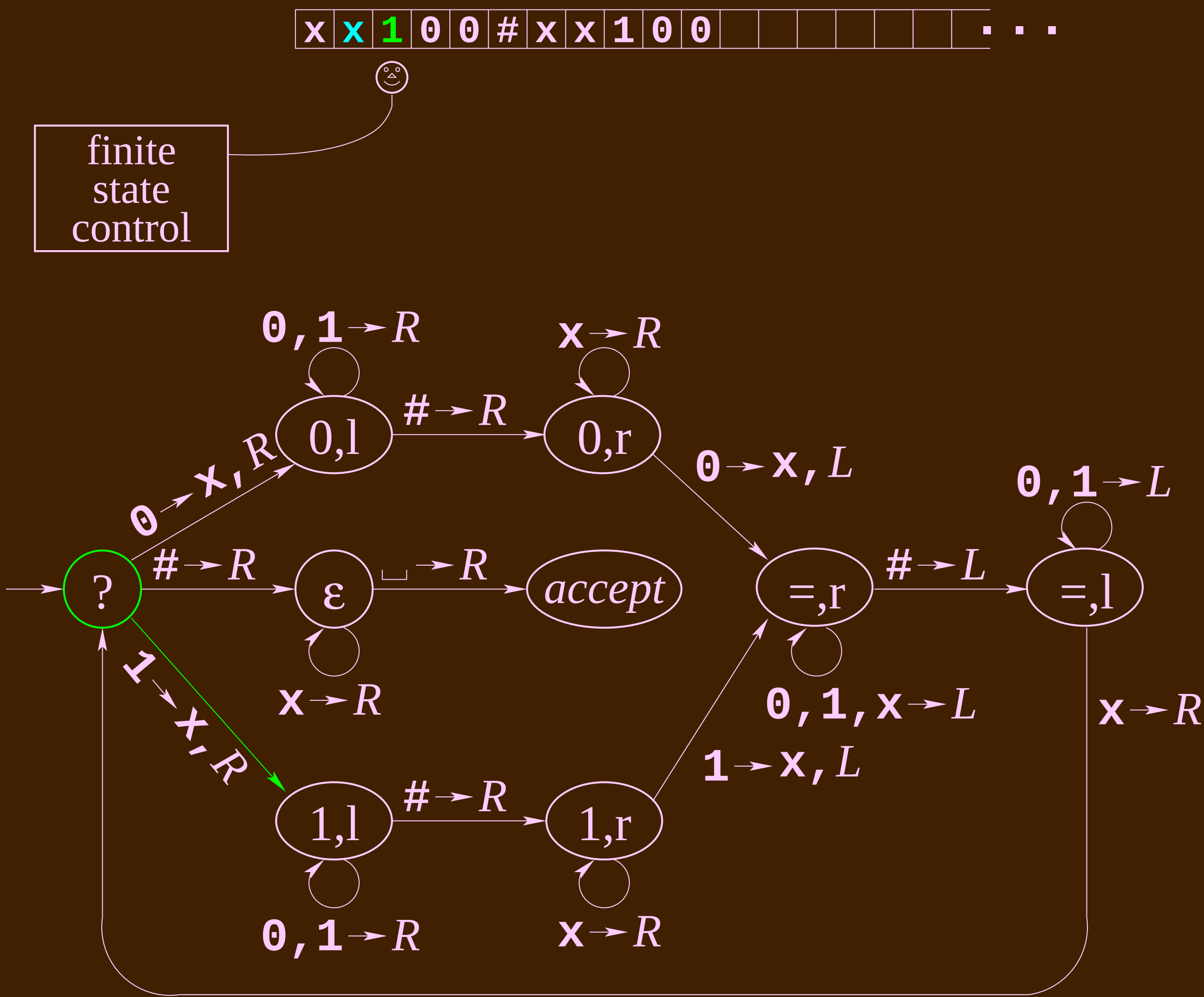
A Machine for $w\#w$



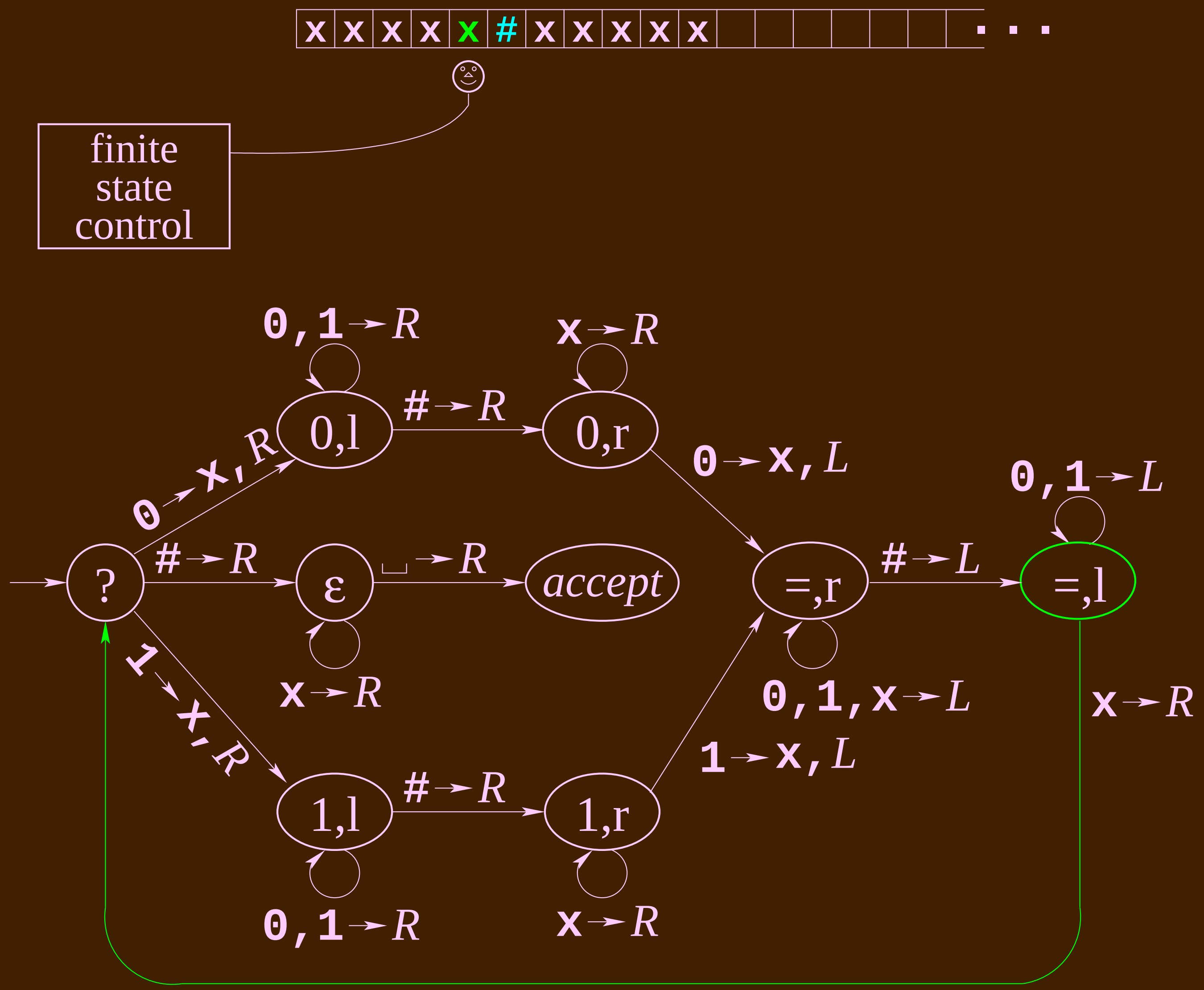
A Machine for $w\#w$



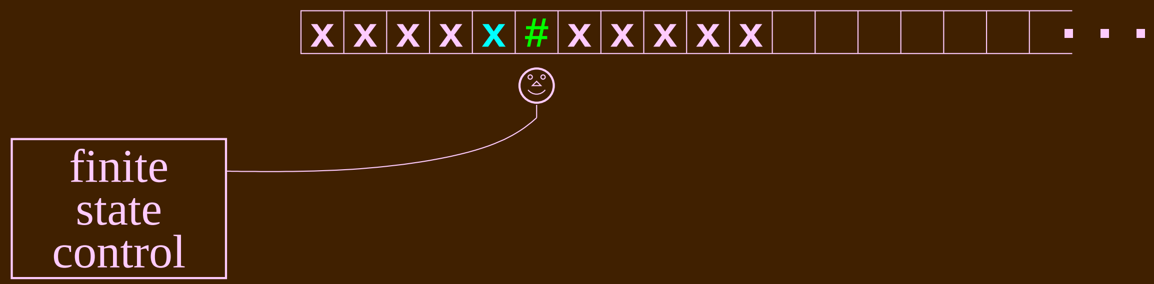
A Machine for $w\#w$

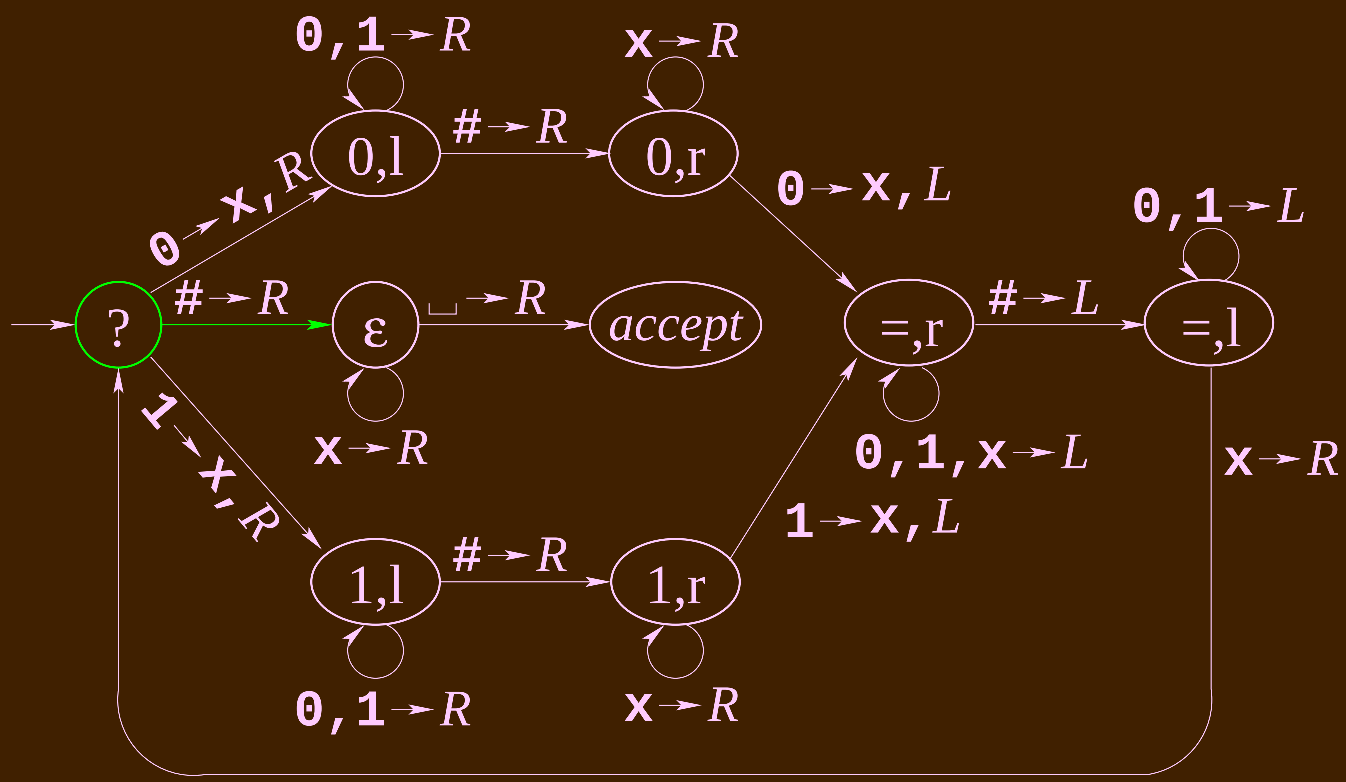


A Machine for $w\#w$

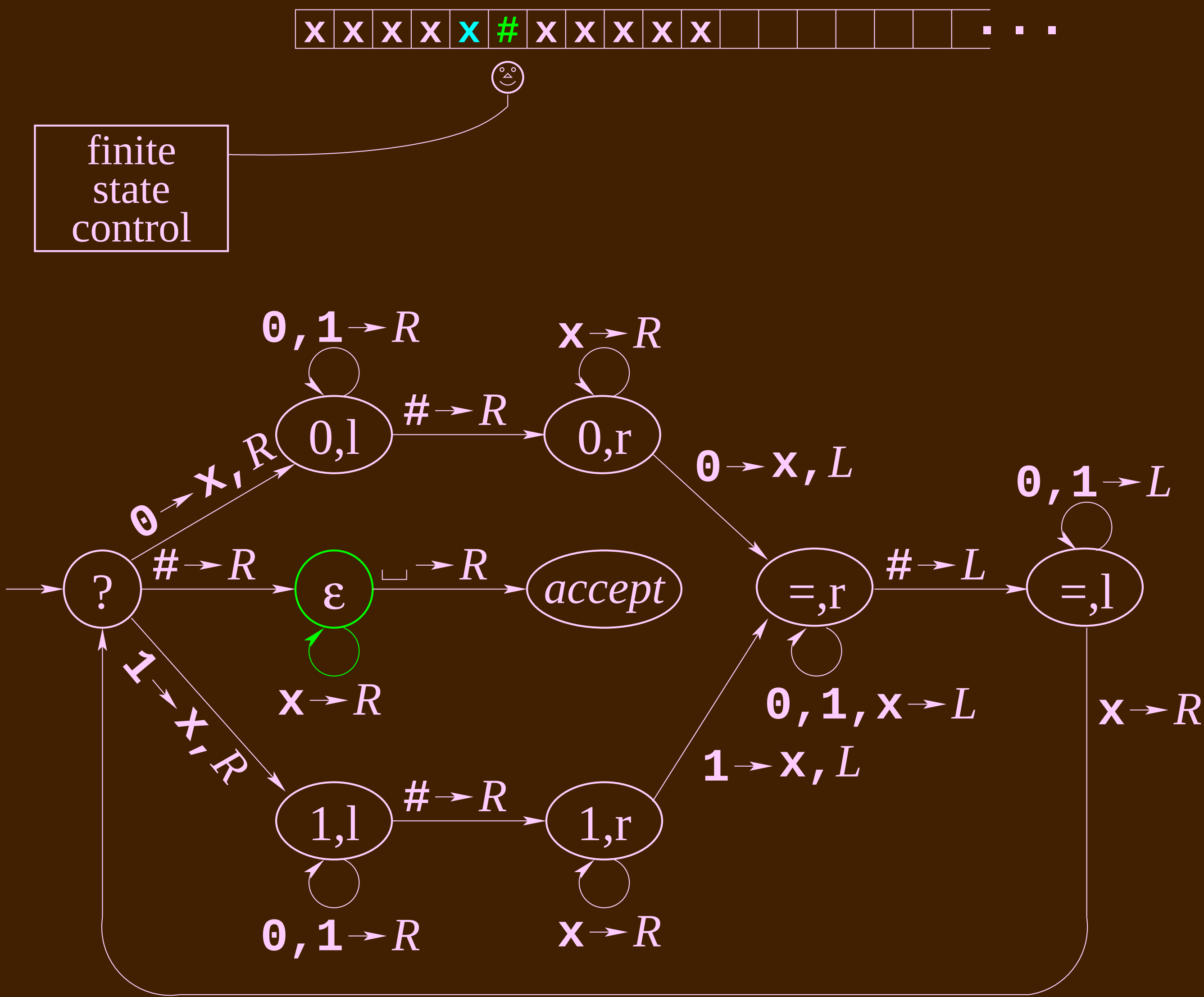


A Machine for $w\#w$

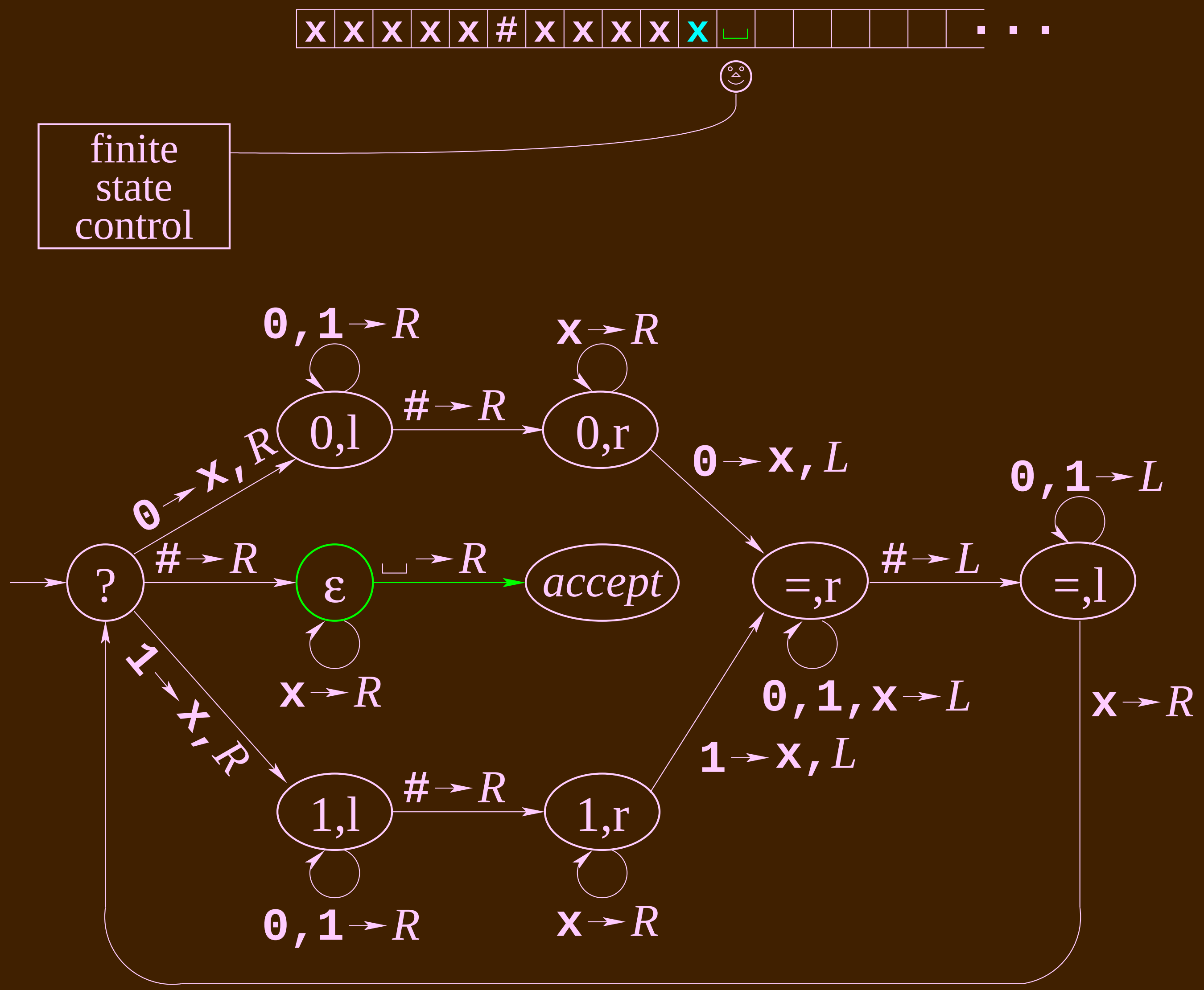




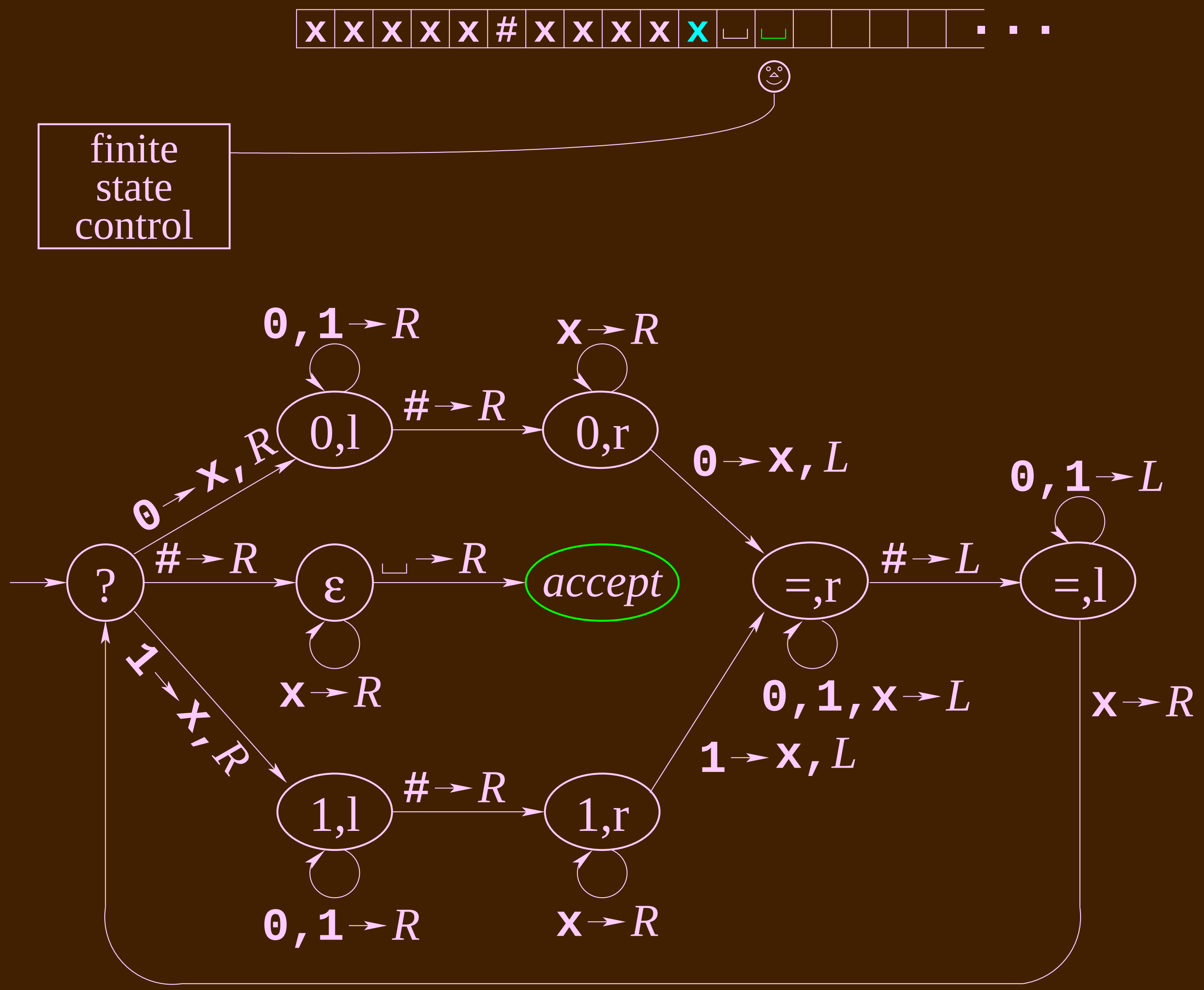
A Machine for $w\#w$



A Machine for $w\#w$



A Machine for $w\#w$



Formalizing Turing Machines

- A Turing Machine is a 7-tuple: $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where:
  - Q is the finite set of states;
  - Σ is the finite input alphabet. The blank symbol, $\Box$ is not in Σ;
  - Γ is the tape alphabet: $\Sigma \cup \{\Box\} \subseteq \Gamma$;
  - $\delta : Q \times \Gamma \ \rightarrow \ Q \times \Gamma \times \{L, R\}$ is the transition function;
  - $q_0 \in Q$ is the start state;
  - $q_{accept} \in Q$ is the accept state; and
  - $q_{reject} \in Q$ is the reject state.

Configurations

- A configuration is a 3-tuple (u, q, v) where
  - $u \in \Gamma^*$ is the tape content to the left of the head.
  - $v \in Q$ is the current state of the Turing machine.
  - $v \in \Gamma^*$ is the tape content starting at the head and to the right. The first symbol of v is the current symbol under the tape head. There are an infinite number of blanks to the right of v.
- We will often write $u\,q\,v$ to denote a configuration.
- The initial configuration is with input w is $q_0\,w$.

Moves

- A Turing Machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ can move from configuration C_i to configuration C_j iff

$$\begin{array}{lllllll}
C_i = u\, a\, q_i\, b\, v & \wedge & C_j = u\, q_j\, a\, c\, v & \wedge & \delta(q_i, a) = (q_j, c, L), & \text{move to lef} \\
C_i = u\, q_i\, b\, v & \wedge & C_j = u\, d\, q_j\, v & \wedge & \delta(q_i, b) = (q_j, d, R) & \text{move to rig} \\
C_i = \epsilon q_i\, a\, v & \wedge & C_j = q_j\, c\, v & \wedge & \delta(q_i, a) = (q_j, c, L), & \text{stuck at lef} \\
C_i = u\, q_i\, b\, \epsilon & \wedge & C_j = u\, q_j\, d\, \square & \wedge & & \text{tape exten}
\end{array}$$

- We write $C_i \xrightarrow[M]{1} C_j$ to denote that M can move from configuration C_i to configuration C_j in one step.
- If $C_i \xrightarrow[M]{1} C_j$, we say: "$C_i$ yields C_j."

Accepting

- The initial configuration for a TM, M, with start state q_0 reading string w is $q_0 w$.
- M accepts w if there is a sequence of configurations, $C_0, C_1, \ldots C_k$ such that
  - $C_0 = q_0\, w$;
  - For all $0 \leq i < k$, $C_i \xrightarrow[M]{1} C_{i+1}$; and
  - For all $0 \leq i < k$, the state for C_i is not the the reject state.
  - The state for C_k is the accept state.
- If M does not accept w then it may either reject w or M may never terminate. In the latter case, we say that M loops on input w.
- $L(M)$ is the set of all strings that M accepts.

Rejecting

- M rejects w if there is a sequence of configurations, $C_0, C_1, \ldots C_k$ such that
  - $C_0 = q_0\, w$;
  - For all $0 \leq i < k$, $C_i \xrightarrow[M]{1} C_{i+1}$; and
  - For all $0 \leq i < k$, the state for C_i is not the the accept state.
  - The state for C_k is the reject state.
- If M does not reject w then it may either accept w or M may never terminate. In the latter case, we say that M loops on input w.

Recognizable/Decidable Languages

- Language L is Turing recognizable iff there exists a Turing machine M such that:
  - M accepts every string in L.
  - M rejects or loops for every string that is not in w.
- Language L is Turing decidable iff there exists a Turing machine M such that:
  - M accepts every string in L.
  - M rejects every string that is not in w.