# COMP30026 Models of Computation

## Pushdown Automata





Lecture Week 9



Semester 2, 2021

# Leftmost derivation

Consider the grammar:



$$E \rightarrow T \mid T + E$$



Here is the



$$\begin{aligned} E &\Rightarrow T \Rightarrow F * T \Rightarrow (E) * T \\ &\Rightarrow (F + E) * T \Rightarrow (3 + E) * T \\ &\Rightarrow (3 + F) * T \Rightarrow (3 + 7) * T \Rightarrow (3 + 7) * F \\ &\Rightarrow (3 + 7) * 2 \end{aligned}$$

# Pushdown Automata

The automata we saw so far were limited by their lack of memory.

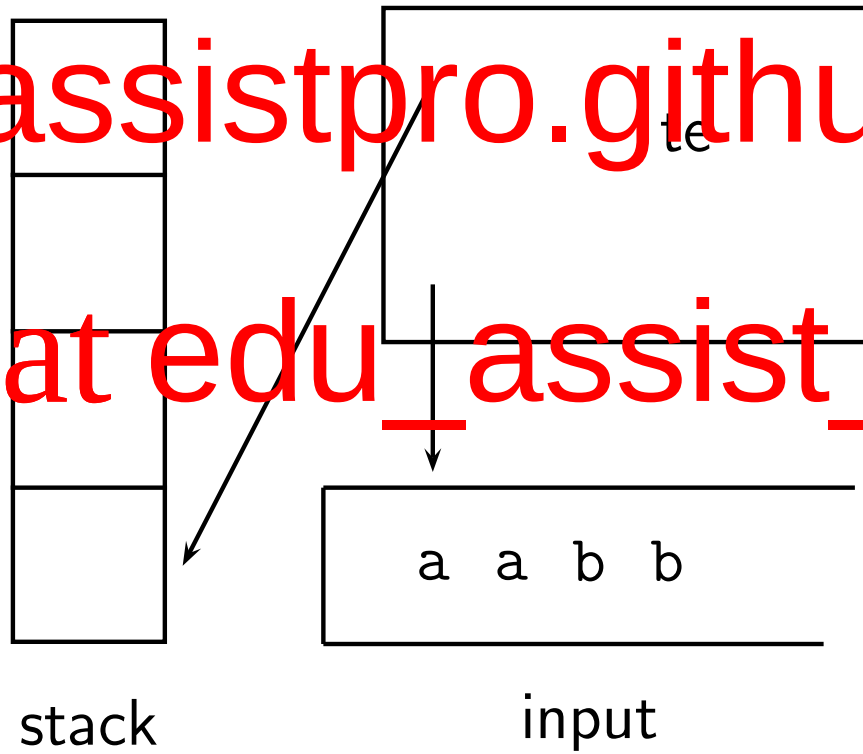


A pushdown automaton (PDA) is a finite-state automaton, equipped with a s



The lang
not reco
since it requires the ability of a recogniser to remember how many consecutive as have been consumed from the input.





# Pushdown Automata

The automata we saw so far were limited by their lack of memory.

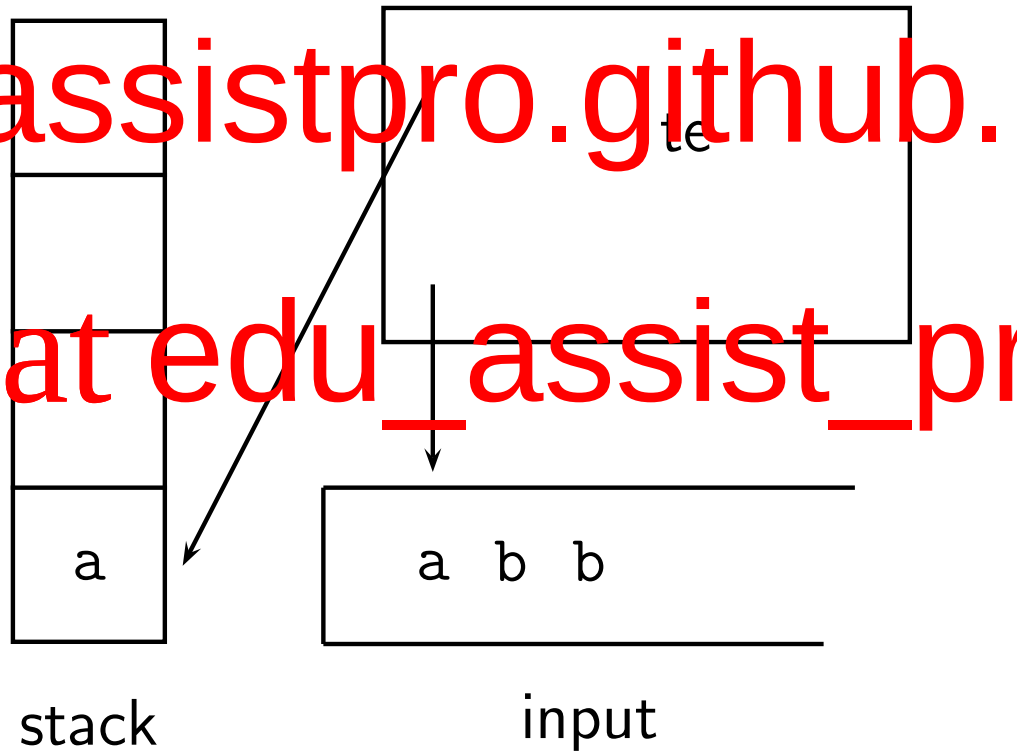


A pushdown automaton (PDA) is a finite-state automaton, equipped with a s



The lang
not reco
since it requires the ability of a recogniser to remember how many consecutive as have been consumed from the input.





# Pushdown Automata

The automata we saw so far were limited by their lack of memory.

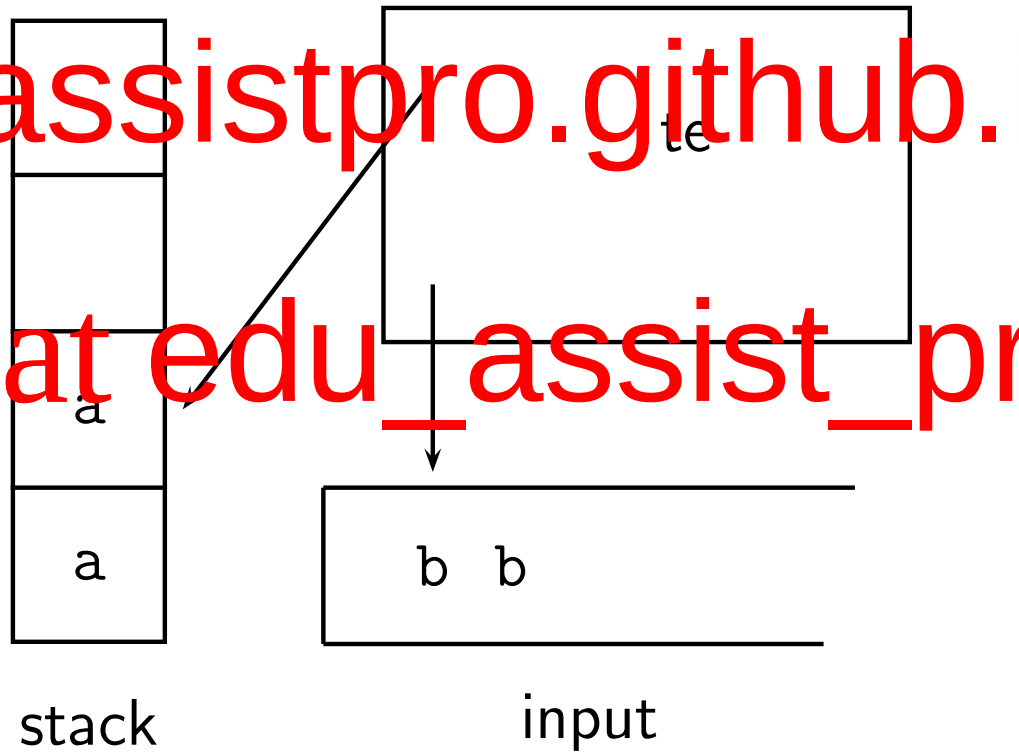


A pushdown automaton (PDA) is a finite-state automaton, equipped with a s



The lang
not reco
since it requires the ability of a recogniser to remember how many consecutive as have been consumed from the input.





# Pushdown Automata

The automata we saw so far were limited by their lack of memory.

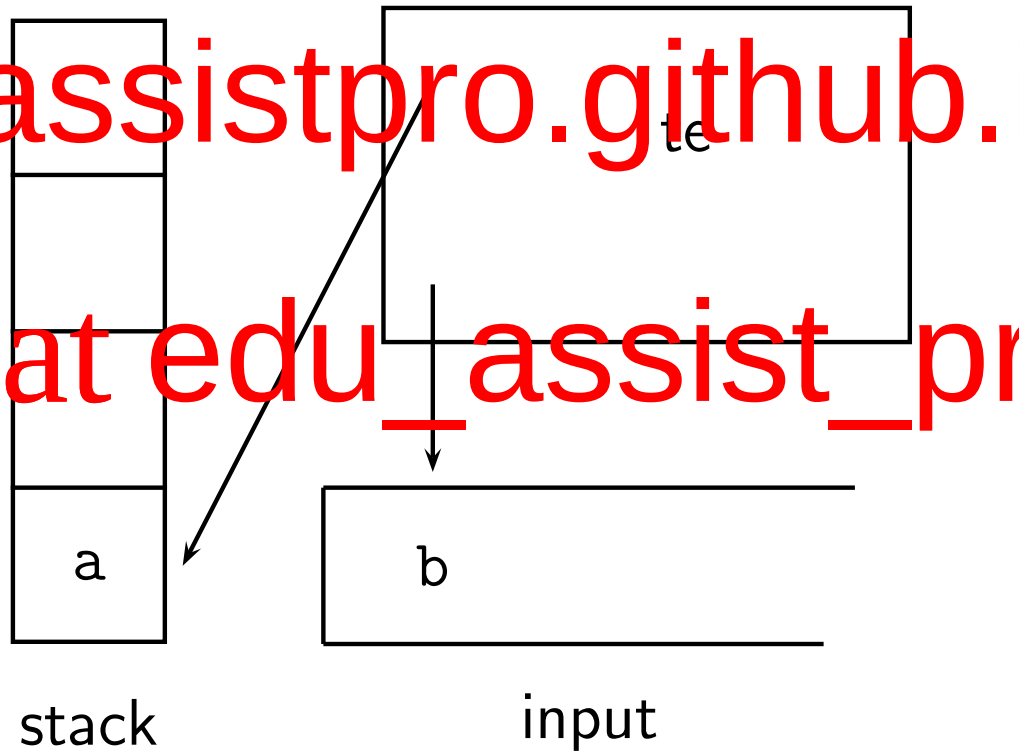


A pushdown automaton (PDA) is a finite-state automaton, equipped with a s



The lang
not reco
since it requires the ability of a recogniser to remember how many consecutive as have been consumed from the input.





# Pushdown Automata

The automata we saw so far were limited by their lack of memory.

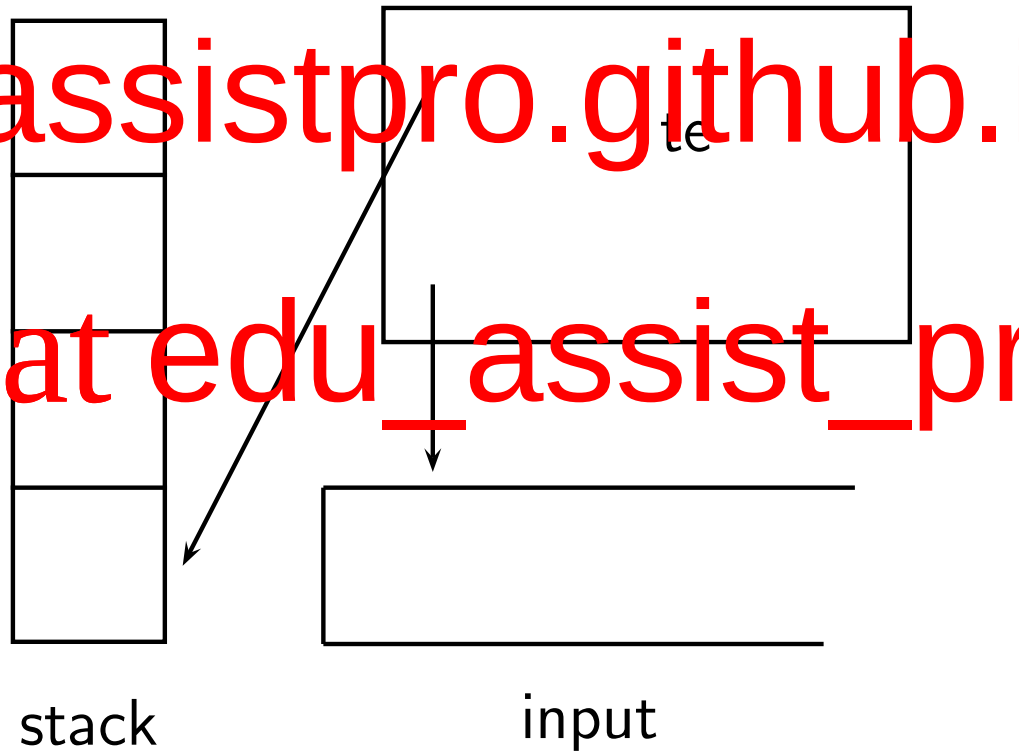


A pushdown automaton (PDA) is a finite-state automaton, equipped with a s



The lang
not reco
since it requires the ability of a recogniser to remember how many consecutive as have been consumed from the input.





# Fine but Important Points



Based on (1) input symbol, (2) top stack symbol and (3) the current state, PDA will decide which state to go to next, as well as, what operatio



In one tran the top stack s e top stack symbol).



We shall consider the non-deterministic

It may also ignore the input.

# Pushdown Automata Formally



A pushdown automaton is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where

- $Q$
- $\Sigma$ is t
- $\Gamma$ is th
- $\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to \mathcal{P}(Q \times \Gamma_\epsilon)$ is the
- $q_0 \in Q$ is the start state, and
- $F \subseteq Q$ are the accept states.

# Example Transitions



$\delta(q_5, a, b) = \{(q_7, \epsilon)\}$ means:

If in state $q_5$ and if the top of the stack holds 'a', consume that a and b by a and go to state $q_7$, and leave the stack holding $\epsilon$.



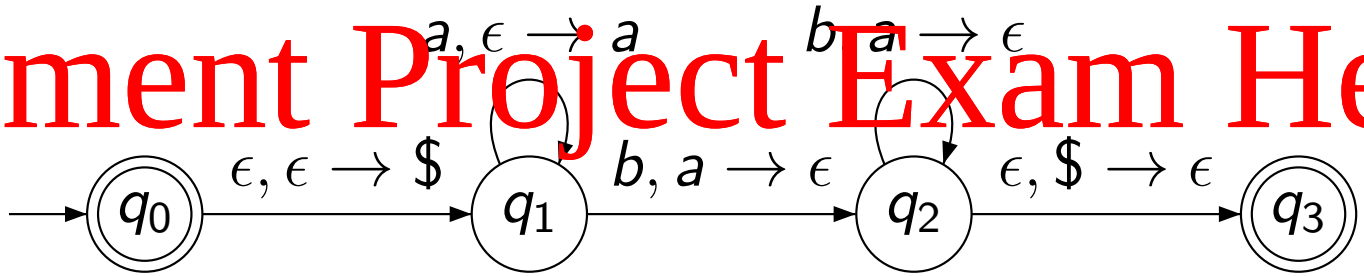
$\delta(q_5, \epsilon, \epsilon) = \{(q_6, \epsilon)\}$ means:

If in state $q_5$, and if the top of the stack holds 'ε', consume that ε by a and go to state $q_6$, and leave the stack holding $\epsilon$.
In either case do not consume an input symbol.

# PDA Example 1

This PDA recognises $\{a^n b^n \mid n \geq 0\}$:





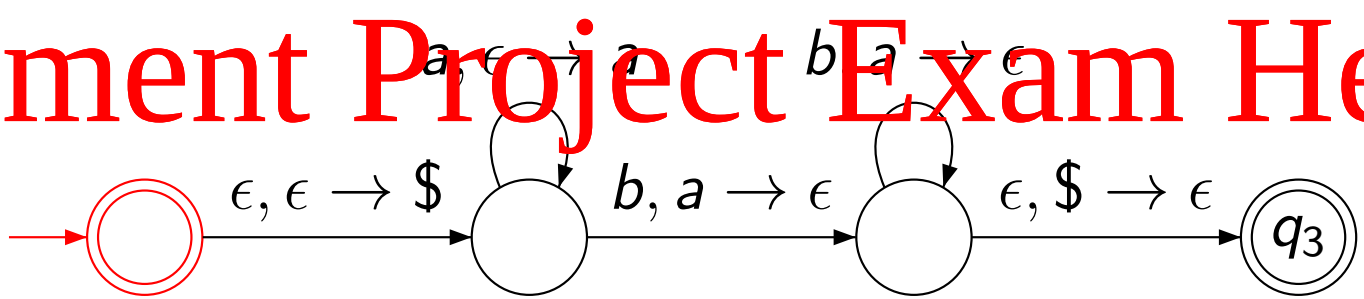



- $Q$
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, \$\}$;
- $\delta(q_0, \epsilon, \epsilon) = \{(q_1, \$)\}$,$\delta(q_1, a, \epsilon) =$
  $\delta(q_1, b, a) = \{(q_2, \epsilon)\}$,$\delta(q_2, b, a) = \{(q_2, \epsilon)\}$,
  $\delta(q_2, \epsilon, \$) = \{(q_3, \epsilon)\}$, for other inputs $\delta$ returns $\emptyset$;
- $q_0 = q_0$;
- $F = \{q_0, q_3\}$.

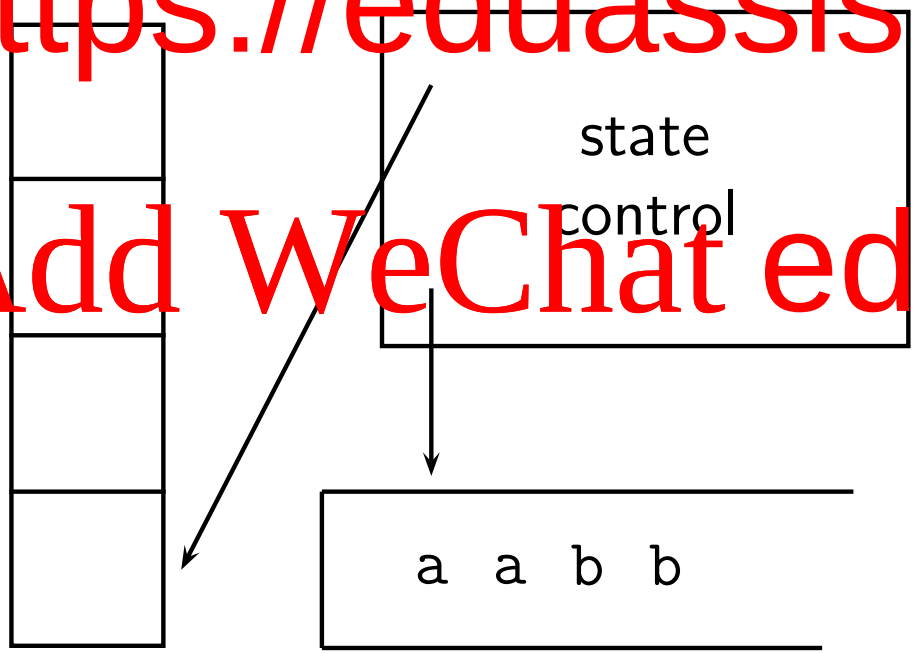
# PDA Example 1

This PDA recognises $\{a^n b^n \mid n \geq 0\}$:





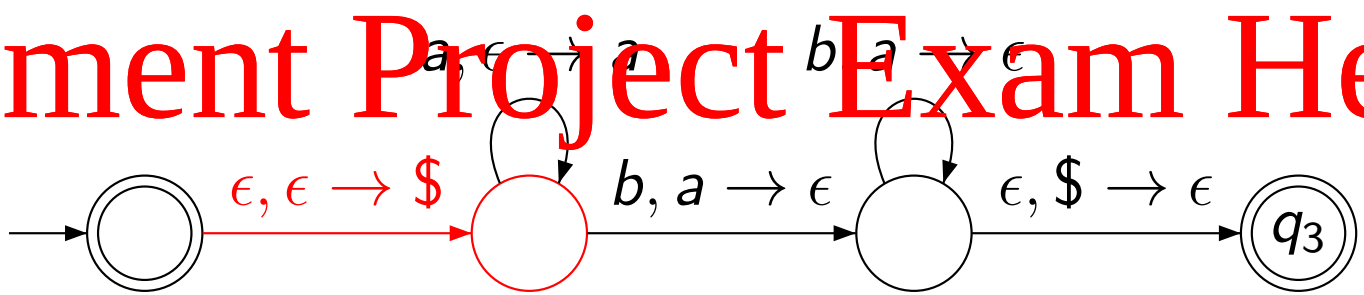


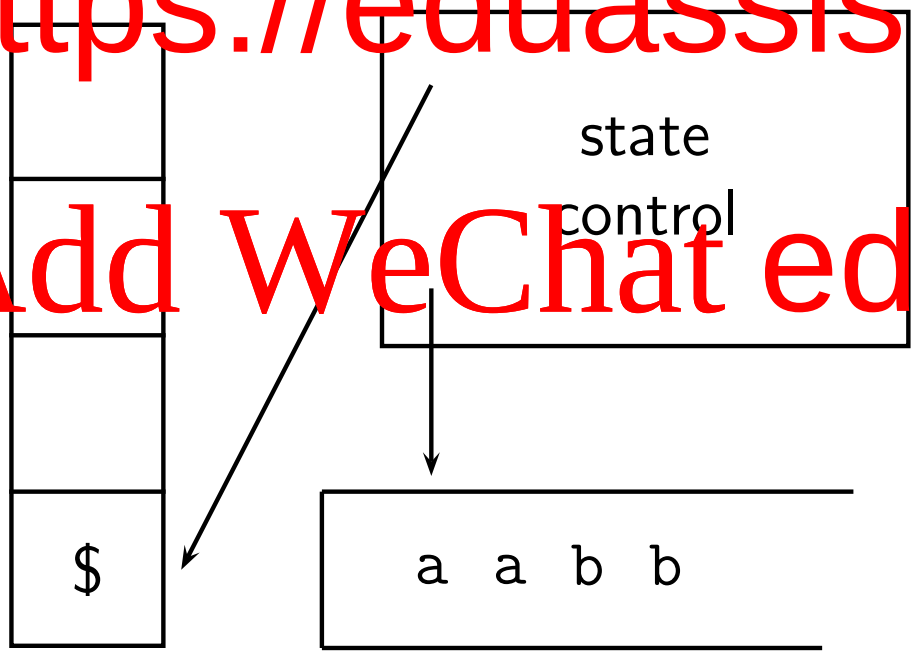
# PDA Example 1

This PDA recognises $\{a^n b^n \mid n \geq 0\}$:





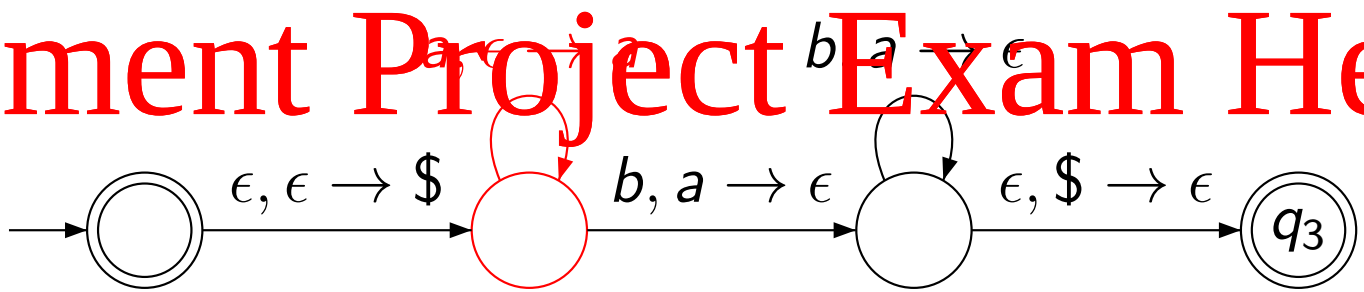


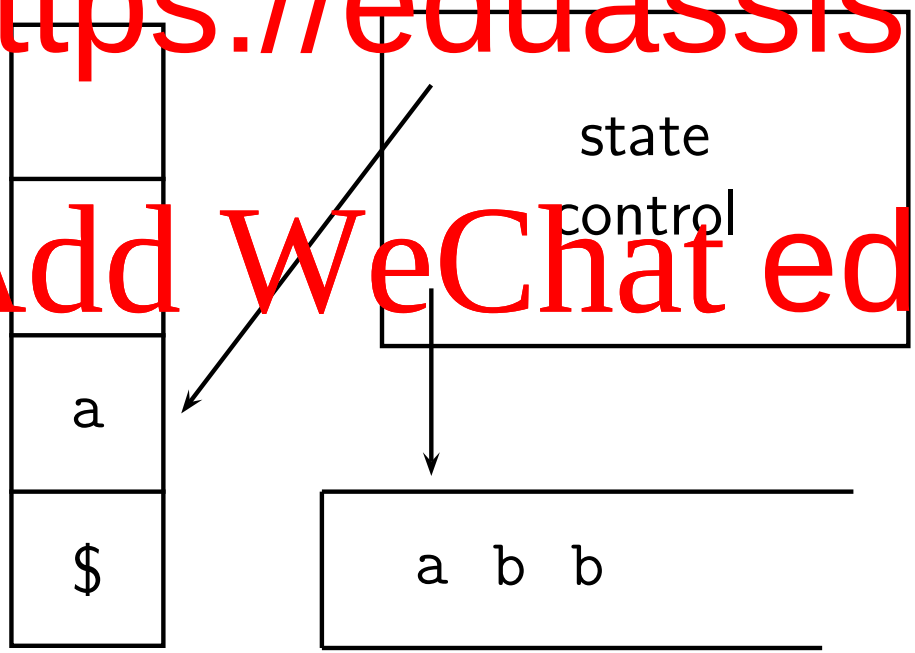
# PDA Example 1

This PDA recognises $\{a^n b^n \mid n \geq 0\}$:





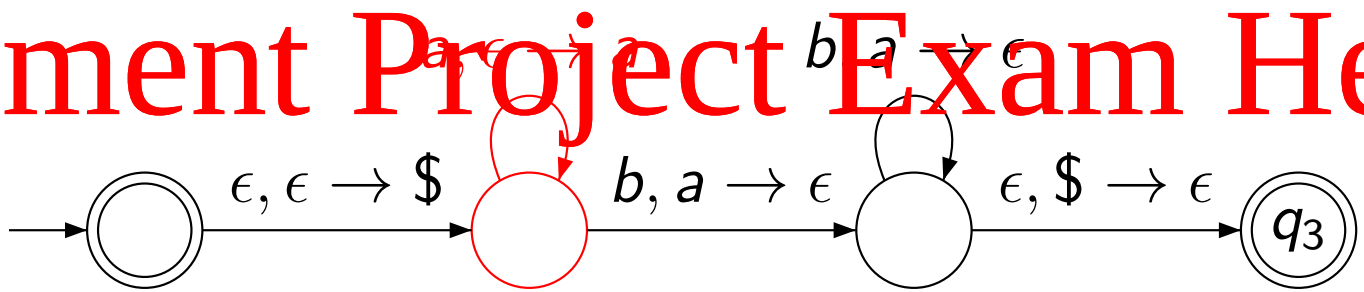


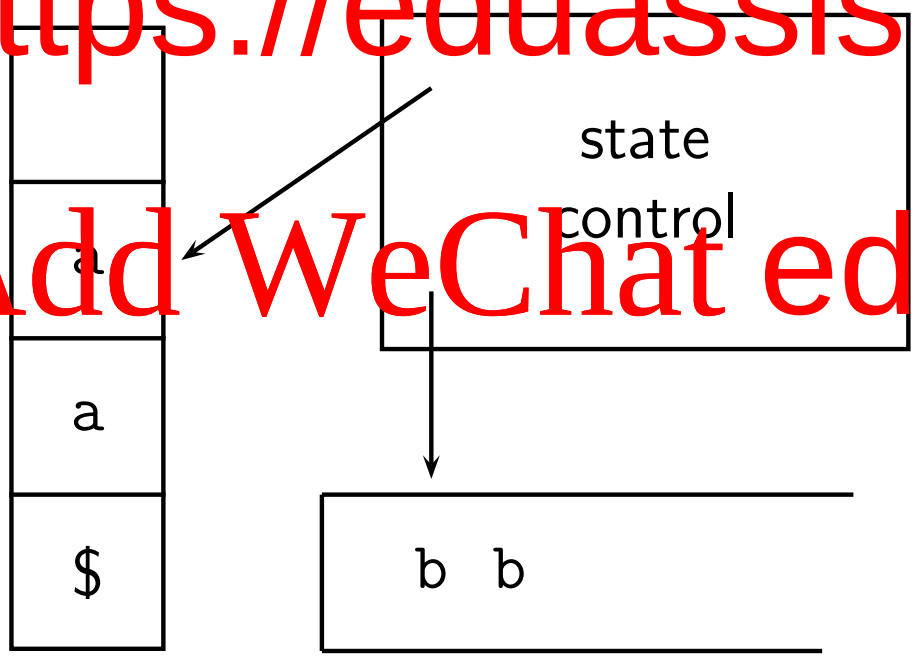
# PDA Example 1

This PDA recognises $\{a^n b^n \mid n \geq 0\}$:





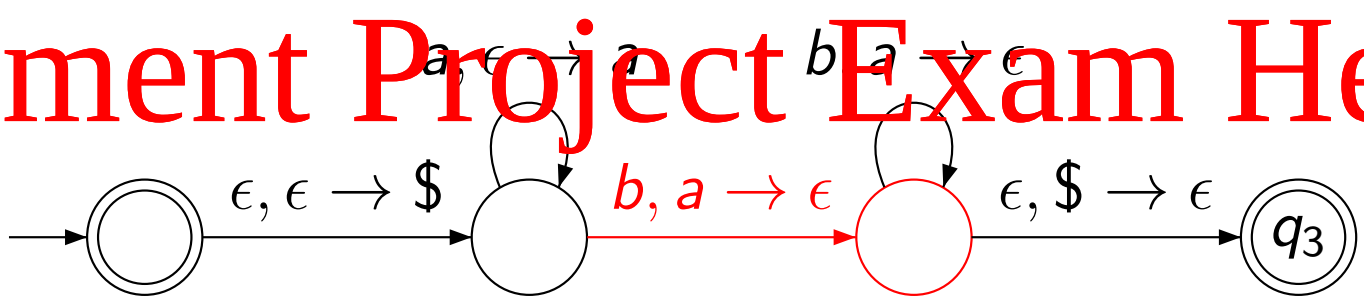


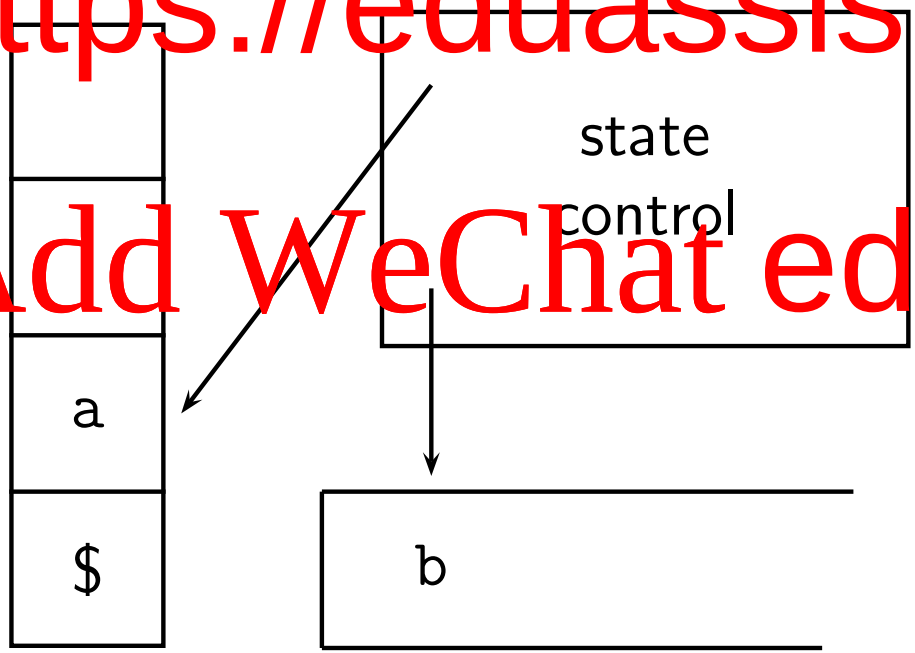
# PDA Example 1

This PDA recognises $\{a^n b^n \mid n \geq 0\}$:





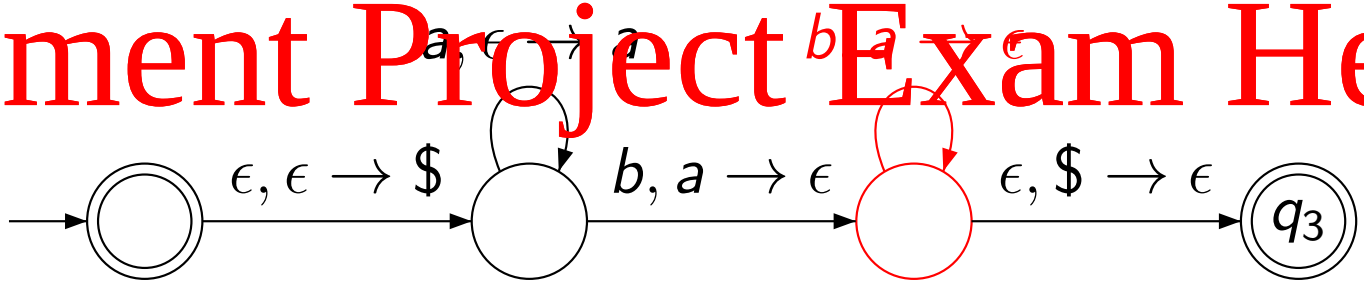


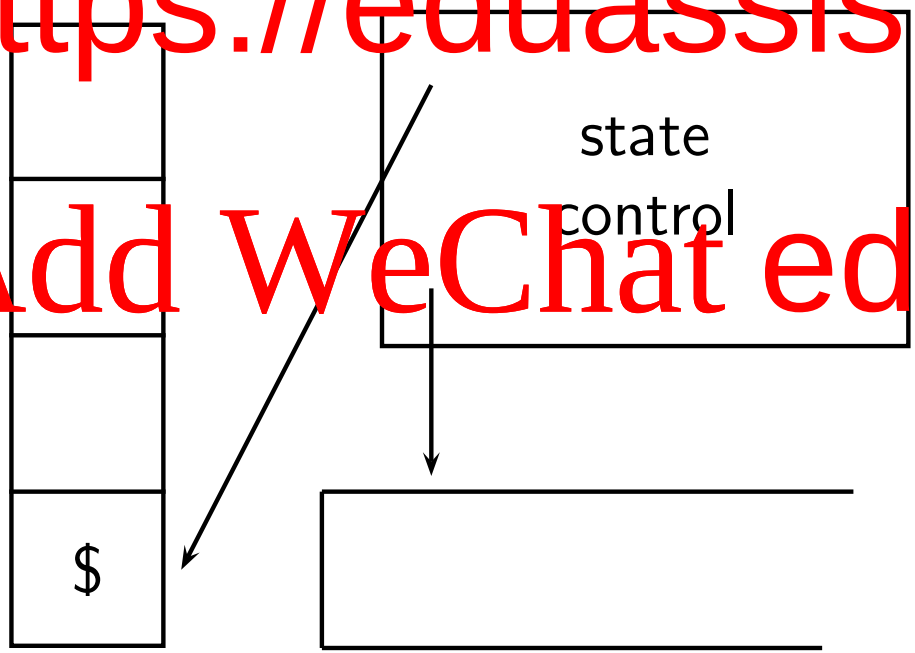
# PDA Example 1

This PDA recognises $\{a^n b^n \mid n \geq 0\}$:





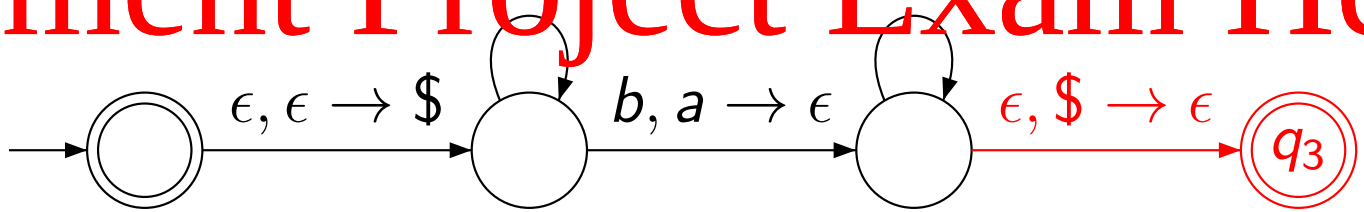


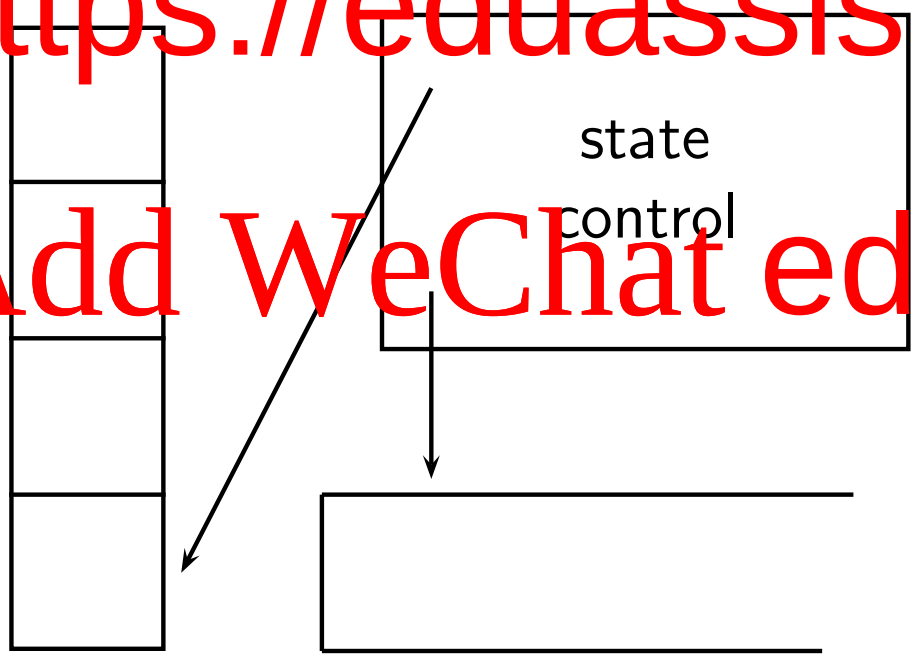
# PDA Example 1

This PDA recognises $\{a^n b^n \mid n \geq 0\}$:







# Acceptance Precisely

The PDA $(Q, \Sigma, \Gamma, \delta, q_0, F)$ accepts input $w$ iff $w = v_1 v_2 \cdots v_n$ with each $v_i \in \Sigma_\epsilon$, and there are states $r_0, r_1, \ldots, r_n \in Q$ and strings $s_0, s_1, \ldots, s_n \in \Gamma^*$ such that

1. $r_0$
2. $(r_i, t) \in \delta(r_{i-1}, v_i, s_{i-1})$ where $t \in \Gamma_\epsilon$ and $s_{i-1} = t s_i$
3. $r_n \in F$.



**Note 1:** There is no requirement that $s_n$ be non-empty when the machine stops (even when it accepts).

**Note 2:** Trying to pop an empty stack leads to rejection of input, rather than "runtime error".

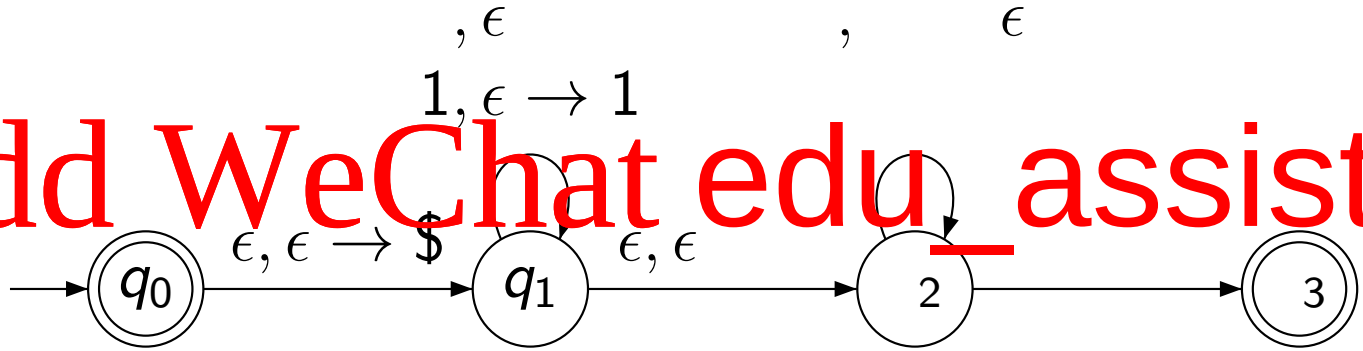
# PDA Example 2



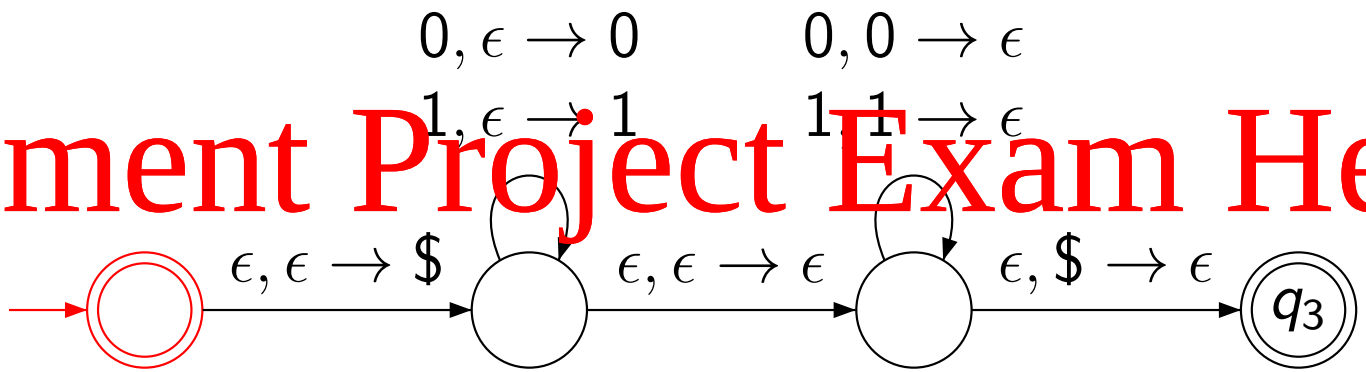
Let $w^{\mathcal{R}}$ denote the string $w$ reversed.

Let us design a PDA to recognise $\{ww^{\mathcal{R}} \mid w \in \{0,1\}^*\}$, the set of even-len

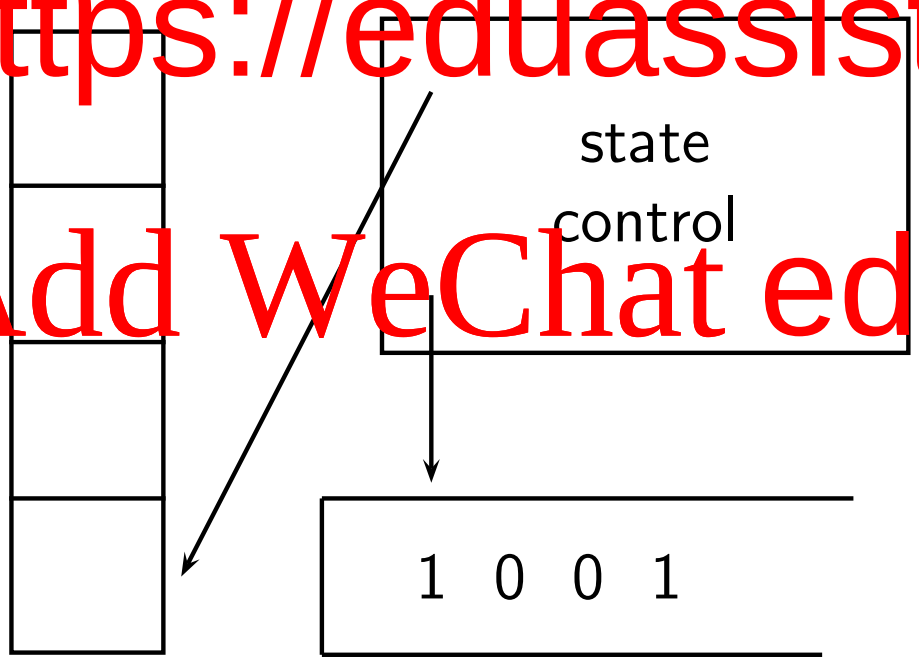




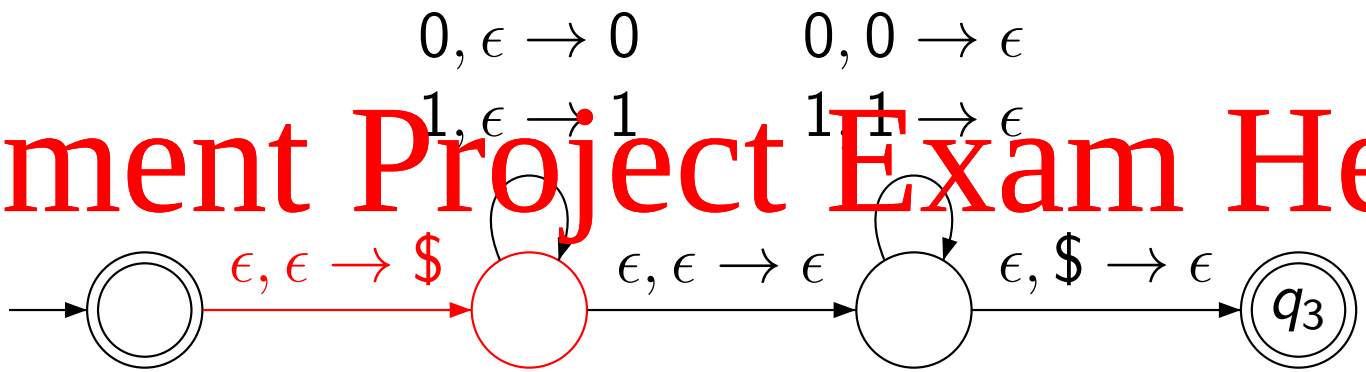
# PDA Example 2



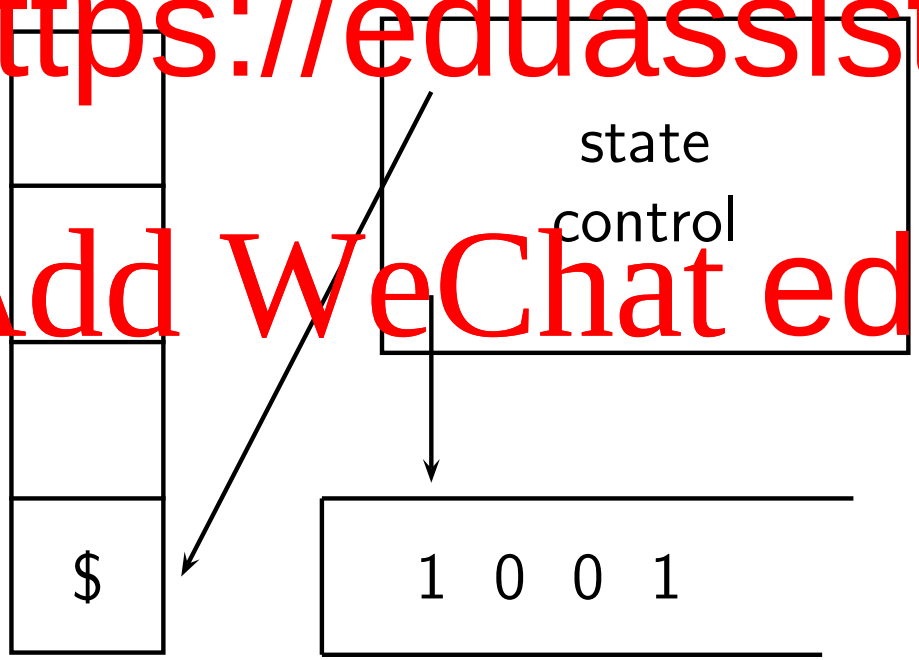


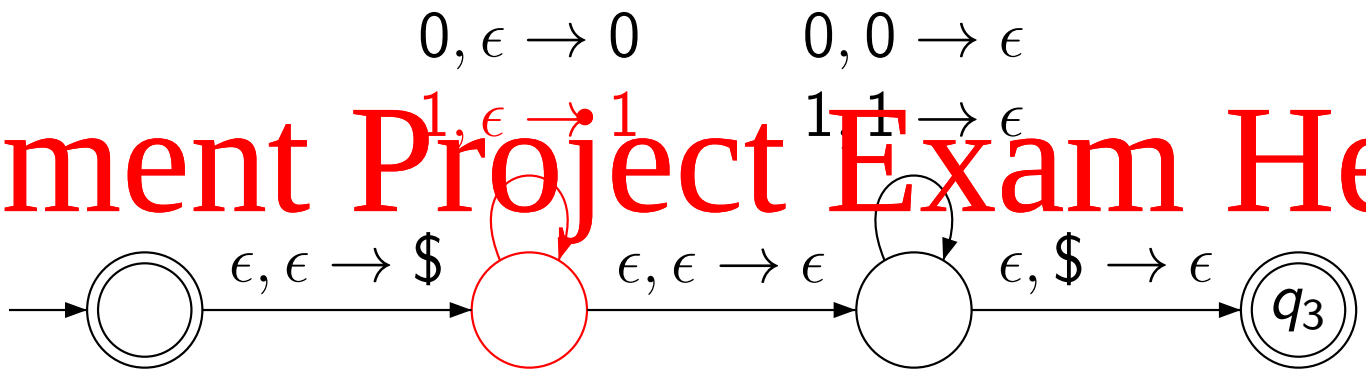
# PDA Example 2



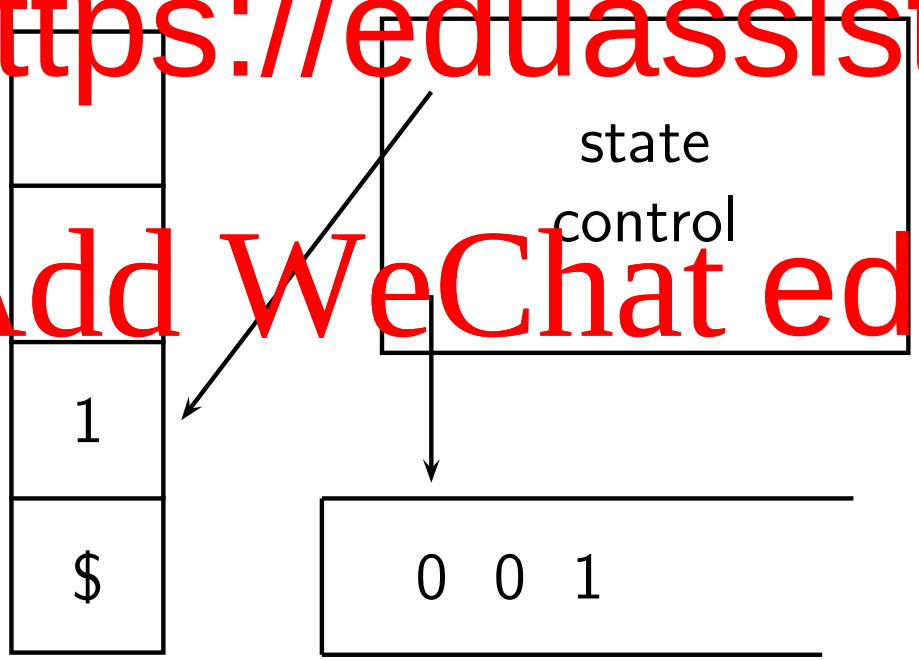


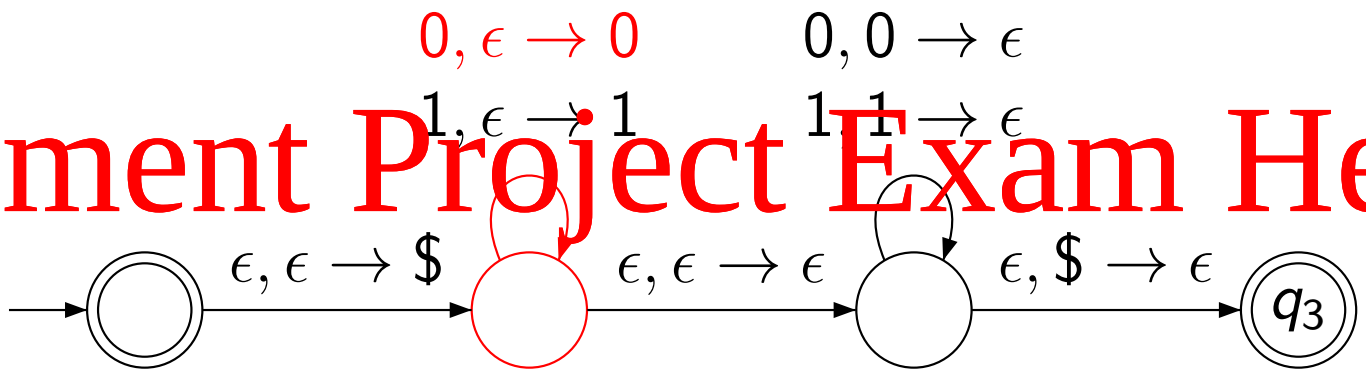
# PDA Example 2



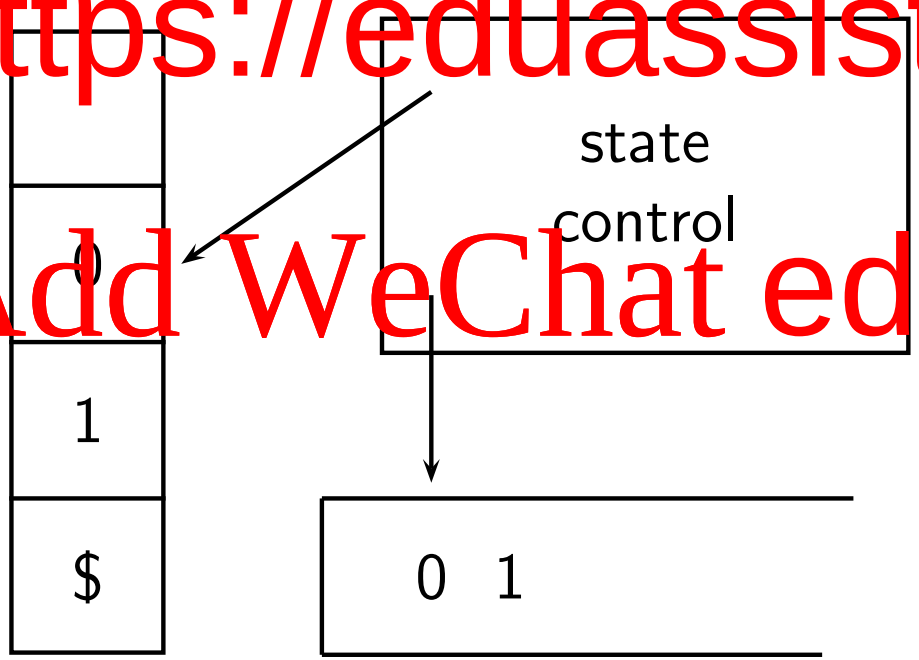


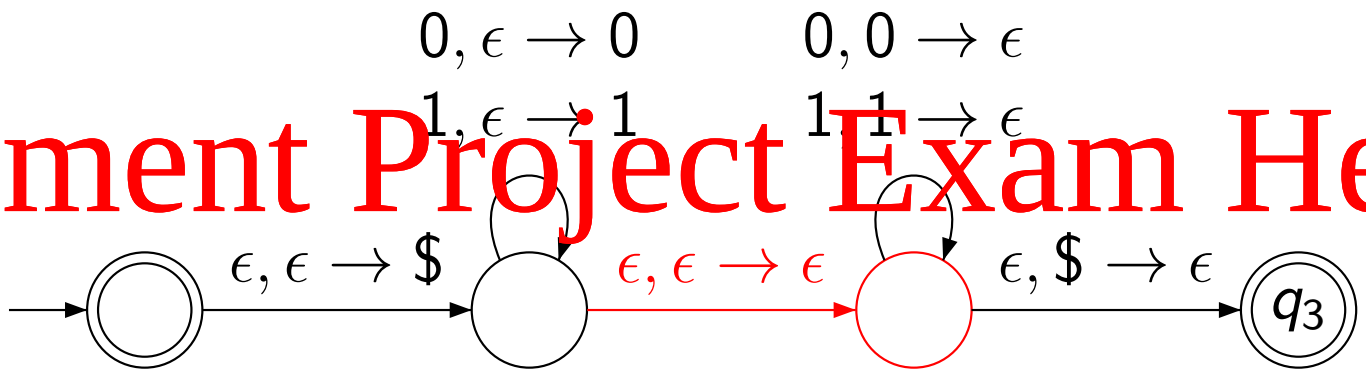
# PDA Example 2



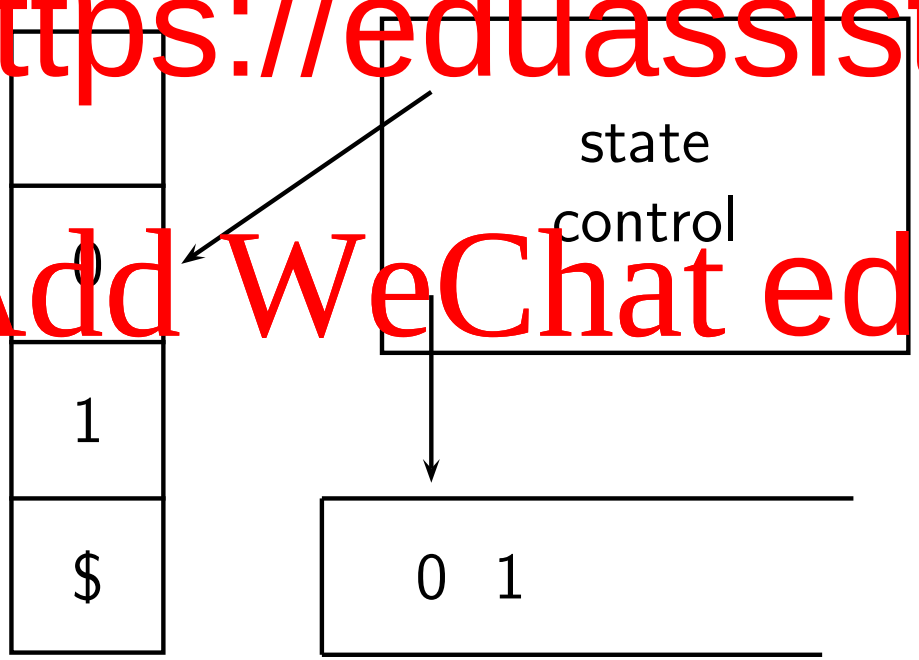


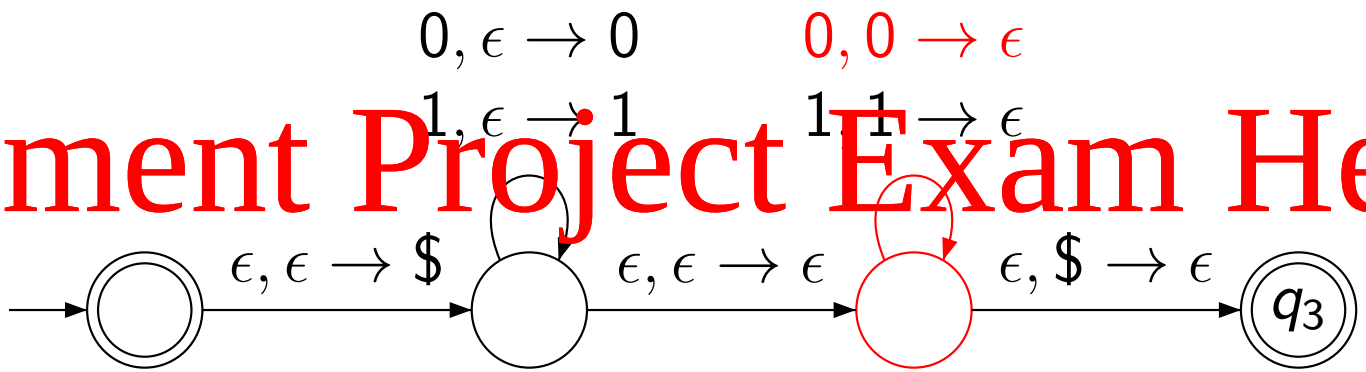
# PDA Example 2



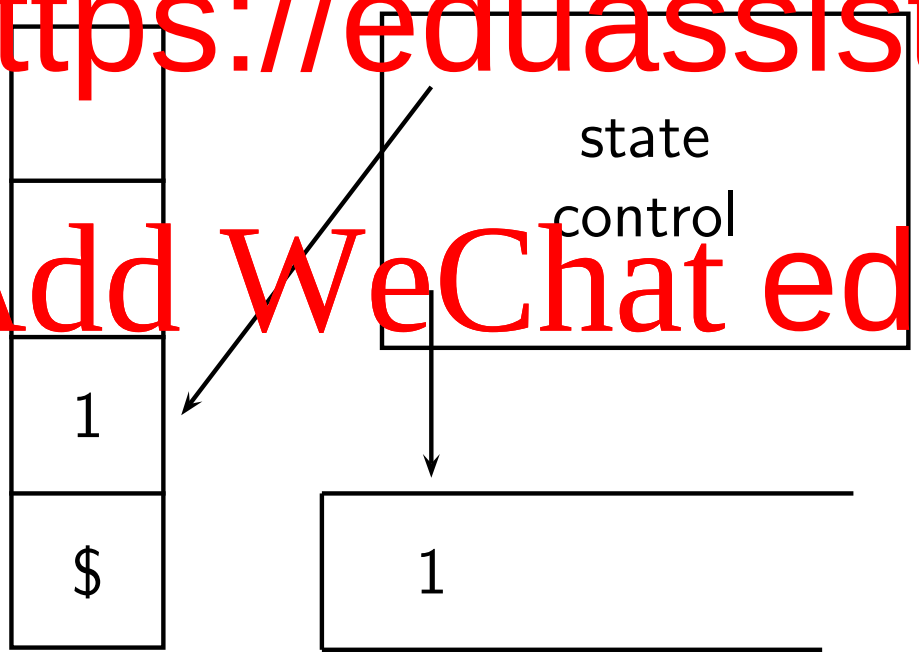


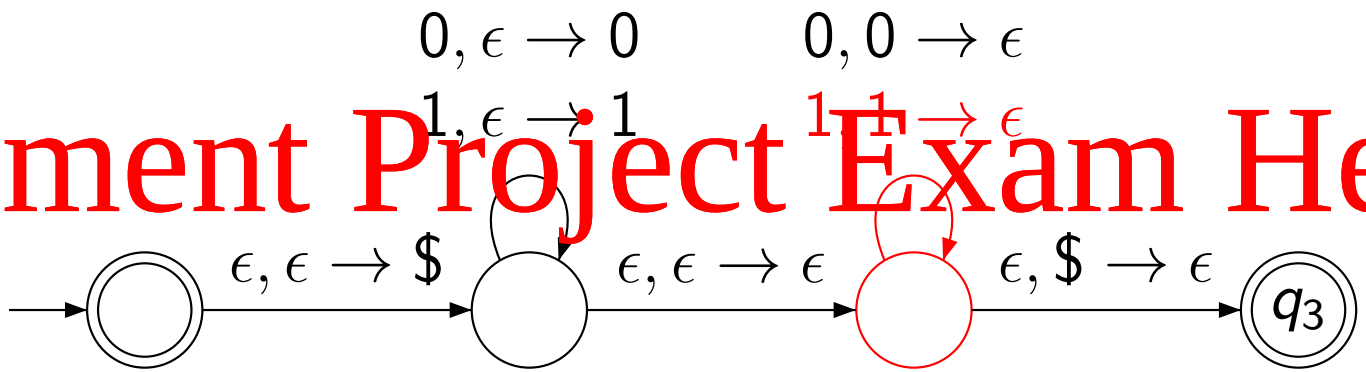
# PDA Example 2



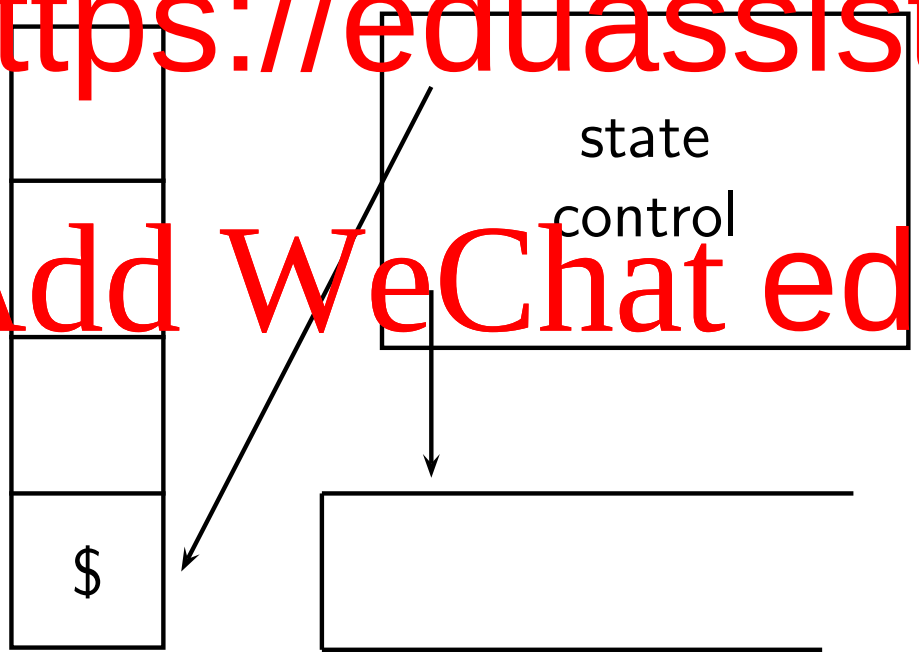


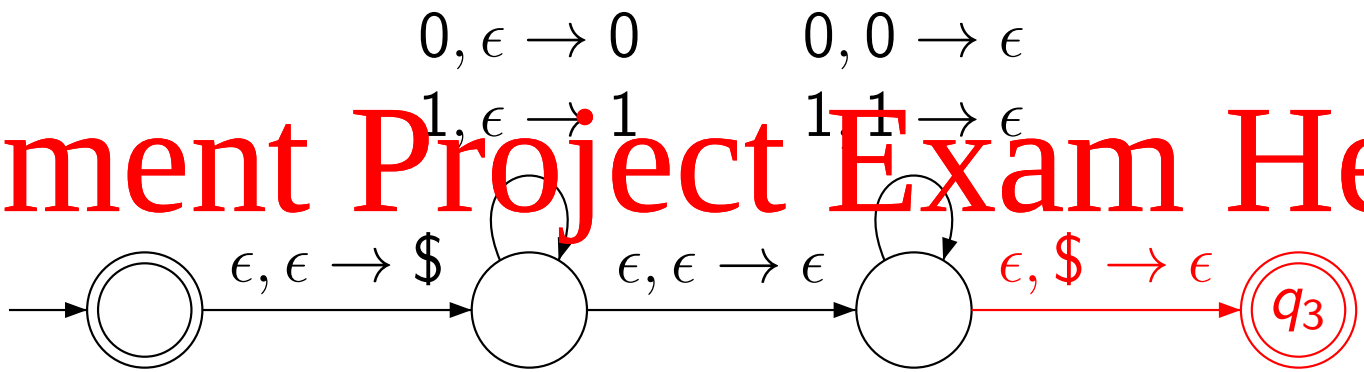
# PDA Example 2



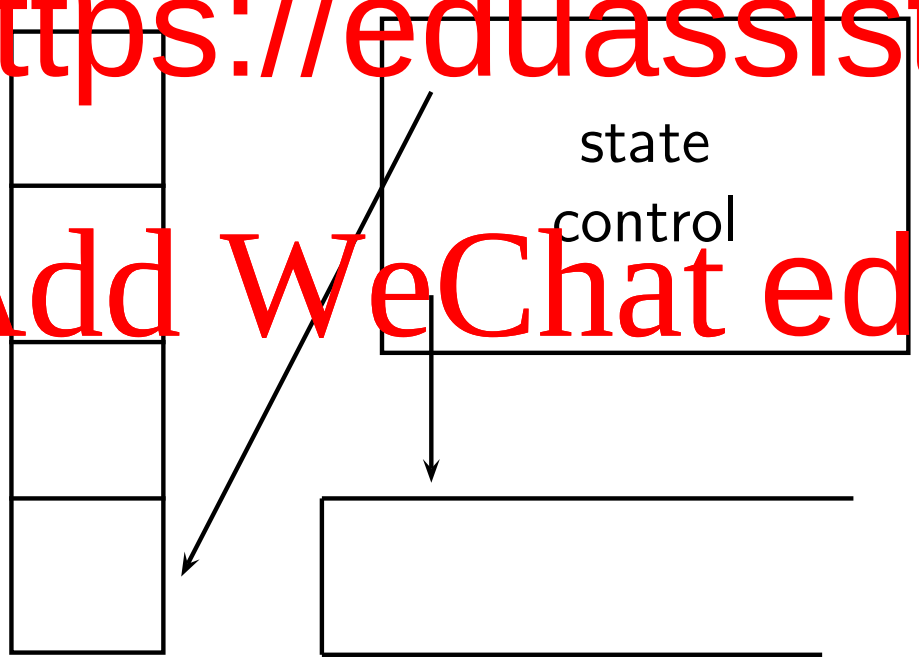


# PDA Example 2





# Progressive PDAs



A pushdo

$\forall q \in Q$



A pushdo

step consumes exactly one input symbol.

# Deterministic PDAs

Is a deterministic PDA (a DPDA) as powerful as a PDA?



No. A DPDA can recognise the context-free



but not th

Intuitively a deterministic machine cannot kno dle of the input has been reached. Suppose it gets input



000011000000000110000

A deterministic machine won't know when to start popping the stack.

# Deterministic PDAs



A pushdown automaton $(Q, \Sigma, \Gamma, \delta, q_0, F)$ is *deterministic* iff $\forall q \in$

$|\delta(q, v,$



For any co itions.

A *deterministic* pushdown automaton (DP position where it can make two different transition s

# CFLs Have PDAs as Recognisers

Given a context-free language $L$ (in the form of a grammar), we can find a PDA which recognises $L$.



And, every PDA recognises a context-free language.

We won't be seen to hol



Namely, given a CFG $G$, we show how to const uch that $L(P) = L(G)$.



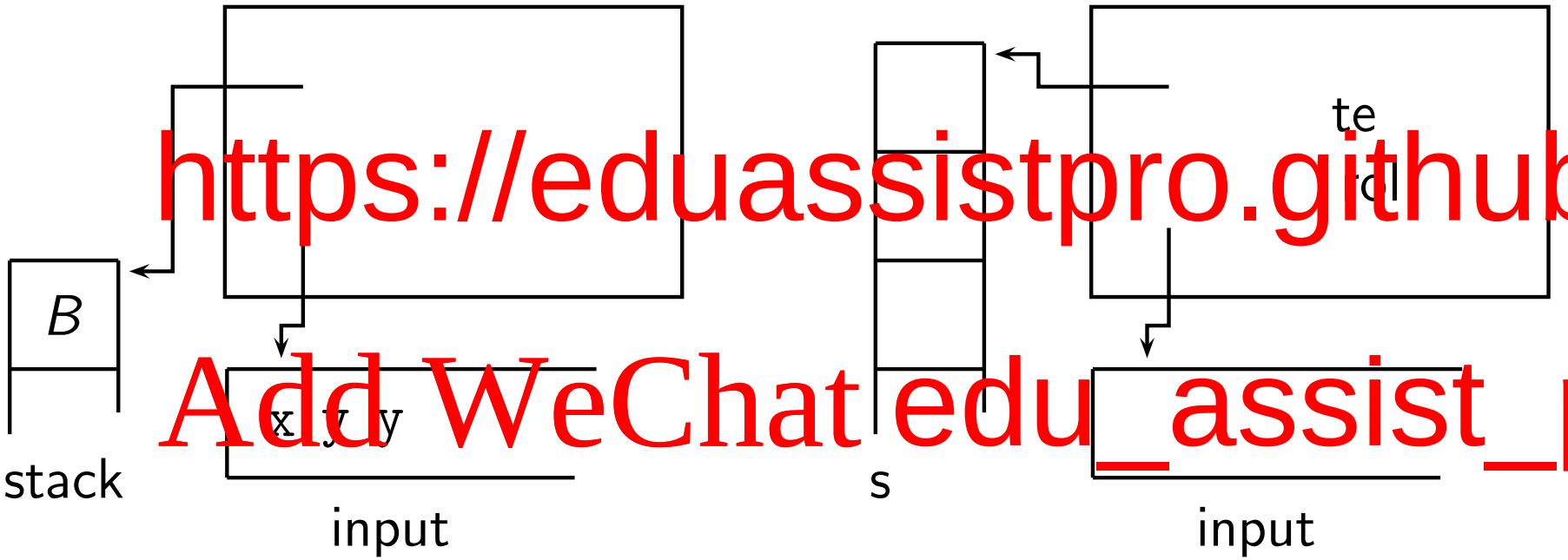
The idea is to let the PDA use its stack to store a list of "pend ng" recogniser tasks.

The construction does not give the cleverest PDA, but it always works.

# From Context-Free Grammars to PDAs

Say $B \to xAy$ is a rule in $G$, and the PDA finds the symbol $B$ on top of its stack, it may pop $B$ and push $y$, $A$, and $x$, in that order.



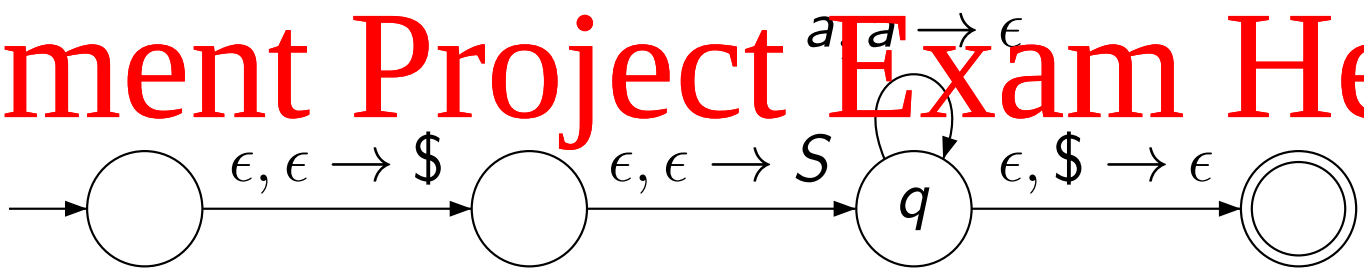



If it finds the terminal $x$ on top of the stack, and $x$ is the next input symbol, it may consume the input and pop $x$.

# From Context-Free Grammars to PDAs

Construct the PDA like this:



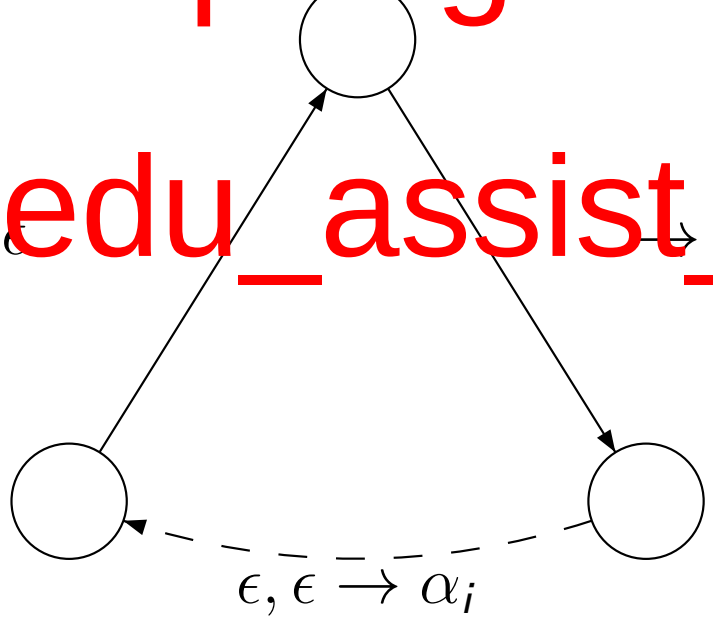



with a self-
($S$ is the g



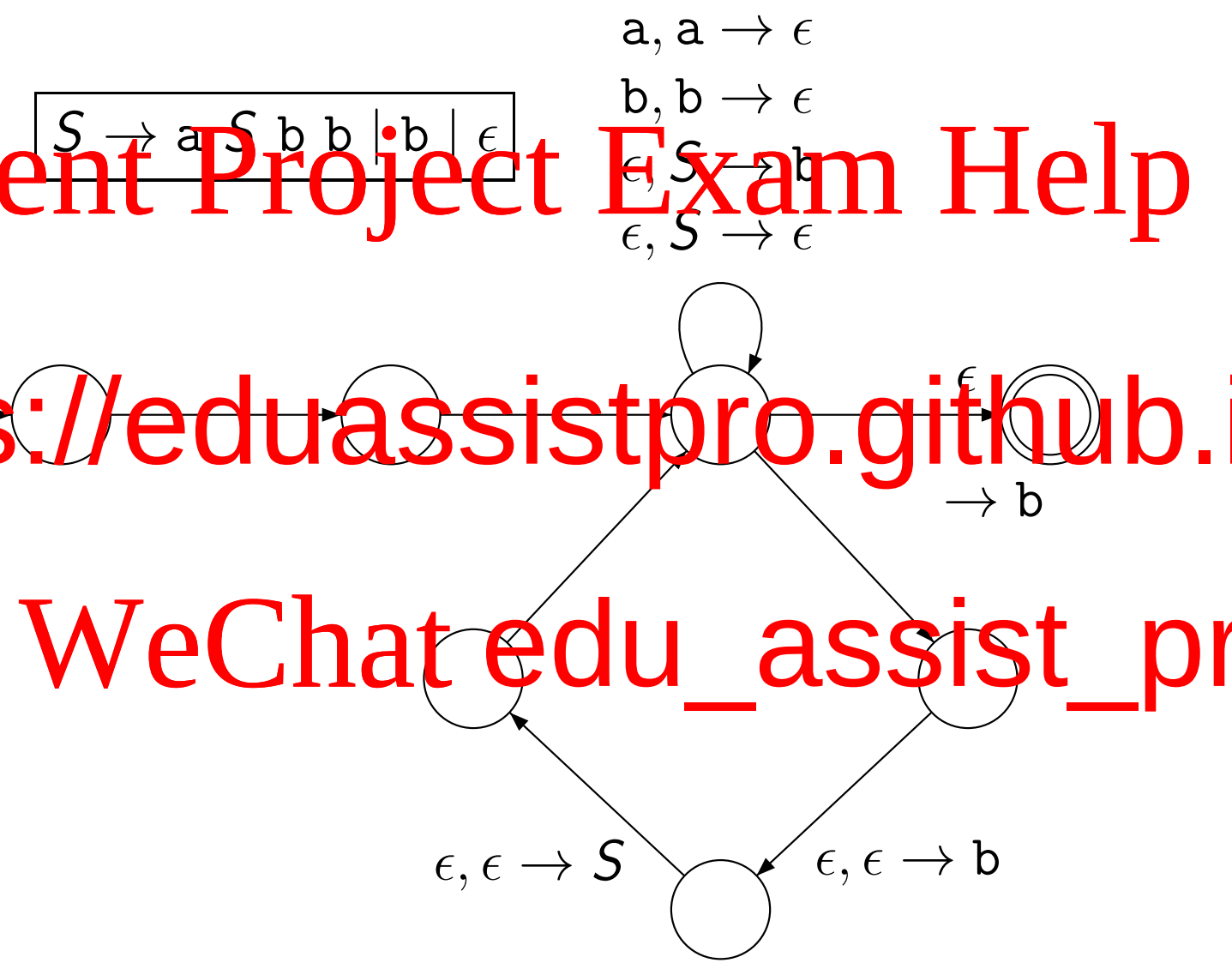


For each rule $A \to \alpha_1 \ldots \alpha_n$,
add this loop from $q$ to $q$:



# Example Recogniser

For the grammar $S \to \mathtt{a}\ S\ \mathtt{b}\ \mathtt{b} \mid \mathtt{b} \mid \epsilon$



# Pumping Lemma for CFLs

There are languages that are not context-free, and again there is a pumping lemma that can be used to show (some) languages non-context-free:

If $A$ is co ring $s \in A$ w &

1. $uv^i xy^i z \in A$ for all $i \geq 0$
2. $|vy| \geq 0$
3. $|vxy| \leq p$

We won't prove this lemma, but we give two examples of its use.

# Pumping Example 1

$A = \{ww \mid w \in \{0,1\}^*\}$ is not context-free.

Assume it is, let $p$ be the pumping length, take $0^p1^p0^p1^p$.



By the pumping lemma, $0^p1^p0^p1^p = uvxyz$, with $uv^ixy^iz$ in $A$ for all $i \geq 0$, an



There are

$$00\ldots0011\ldots1100\ldots$$

If it straddles the midpoint, it has form $1$



are left with $0^p1^i0^j1^p$, with $i < p$, or $j$

If it is in the first half, $uv^2xy^2z$ will have pushed a 1 into the first position of the second half.

Similarly if $vxy$ is in the second half.

# Pumping Example 2

$B = \{a^n b^n c^n \mid n \in \mathbb{N}\}$ is not context-free.



Assume it is, let $p$ be the pumping length, and take $a^p b^p c^p \in B$.

By the pu $B$ for all $i$.



Either $v$

If one of them contains two different symbols from en $uv^2xy^2z$ has symbols in the wrong order, and so ca $B$.



So both $v$ and $y$ must contain only one kind of s
$uv^2xy^2z$ can't have the same number of as, bs, and cs.

In all cases we have a contradiction.

# Closure Properties for CFLs



The class of context-free languages is closed under

- uni
- co
- Kleene star,
- reversal

# Closure Properties for CFLs

The class of context-free languages is not closed under intersection!

Hence it is not closed under complement either (why?)



Conside



**Exercise:** Prove that they are context-free!



But $C \cap D$ is the language $B = \{a^n b^n c$ showed is not context-free.

However, we do have: If $A$ is context-free and $R$ is regular then $A \cap R$ is context-free.