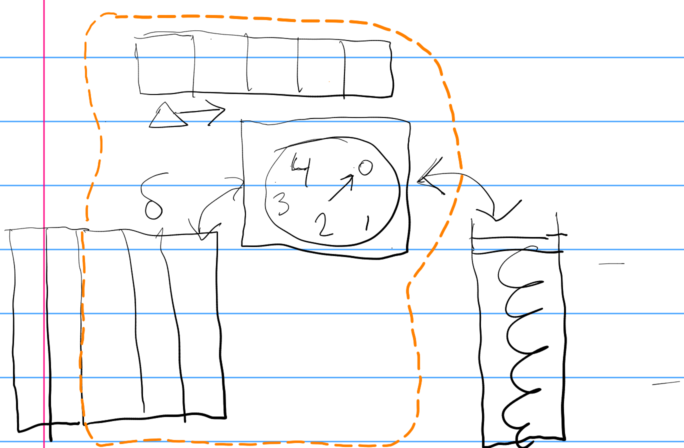
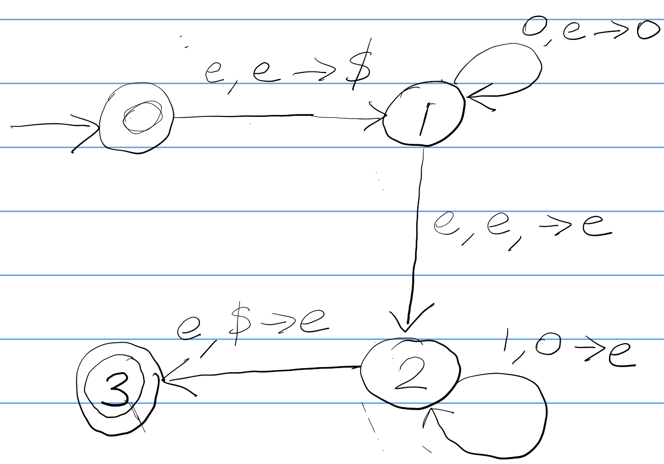
CS605
7 February 2025

2.2 PUSHDOWN AUTOMATA

In this section we introduce a new type of computational model called ***pushdown automata***. These automata are like nondeterministic finite automata but have an extra component called a ***stack***. The stack provides additional memory beyond the finite amount available in the control. The stack allows pushdown automata to recognize some nonregular languages.



$\{0^n 1^n : n \geq 0\}$



Q_1 Σ_1 Γ_1 Q_2 Γ_2

0000111
0001111
010
111000

~~000111~~

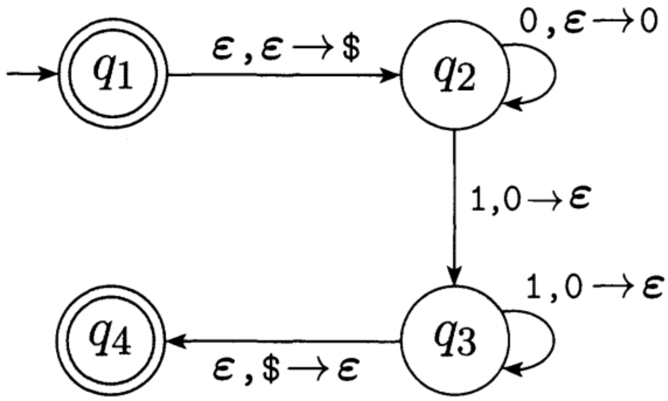
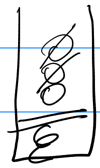
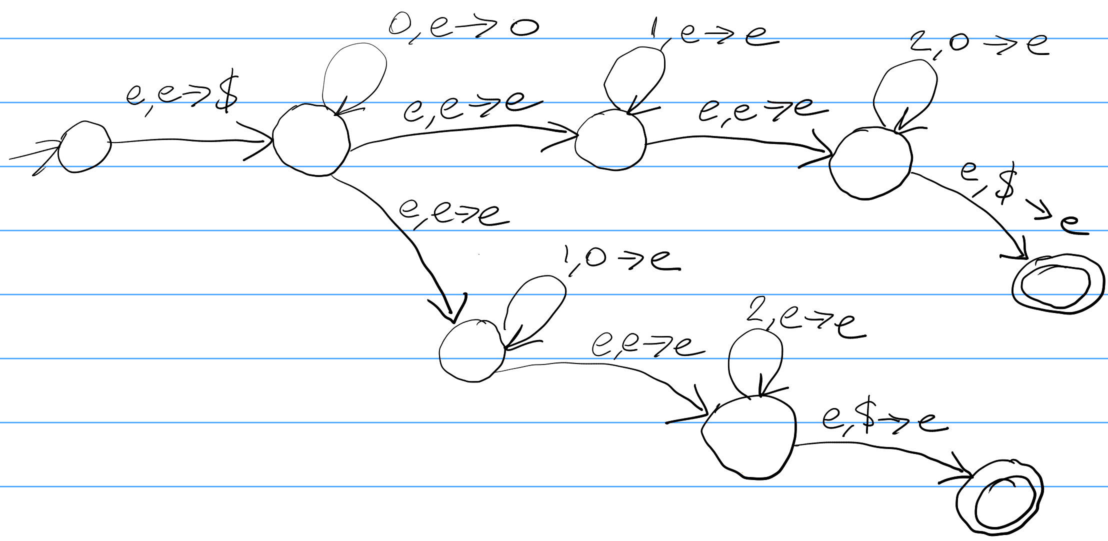


FIGURE 2.15
State diagram for the PDA M_1 that recognizes $\{0^n1^n \mid n \geq 0\}$

```
# This is a PDA to recognise
# {0ˣ1ʸ2ᶻ : x,y,z ≥ 0, x=y or x=z}.
```

00011122 ~~000111222~~





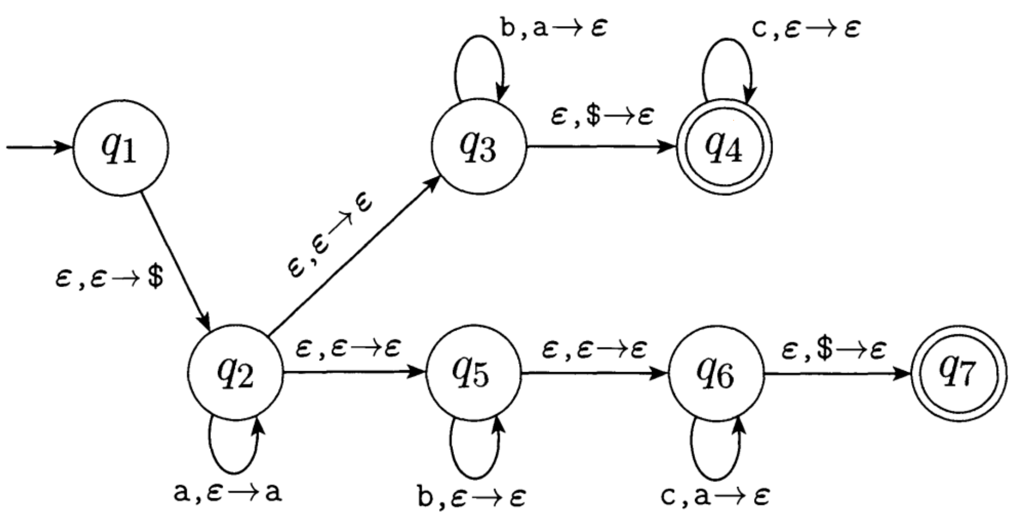


FIGURE 2.17
State diagram for PDA M_2 that recognizes
$\{a^i b^j c^k \mid i, j, k \geq 0 \text{ and } i = j \text{ or } i = k\}$

PDAs to create during Thursday afternoon (6th February 2025)

Th1 $\{0^n1^n, n\geq 0\}$

Th2 $\{0^n1^x2^n, x,n\geq 0\}$

Th3 $\{a^xb^yc^z : x,y,z \geq 0, x=y \text{ or } x=z\}$

Th4 $\{wxw^R, w \in \{0, 1\}^*\}$

PDAs to create during Friday morning (7th February 2025)

L4 $\{xwyw^Rz : x,y,z,w \in \{a,b\}^*, |w|>0\}$

L5 $\{w : w \in \{0,1\}^*$, w has twice as many 0s as 1s$\}$

PDAs to create during Friday afternoon (7th February 2025)

L4b $\{0^{2n}1^n : n \geq 0\}$

L4c $\{w : w \in \{0,1\}^*$, w has an equal number of 0s as 1s$\}$

L6 $\{u111v : u,v \in \{0, 1\}^*, |u| = |v|\}$

EXAMPLE 2.18

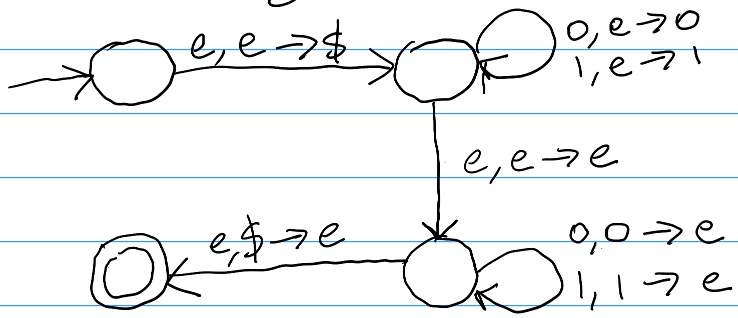
In this example we give a PDA M_3 recognizing the language $\{ww^{\mathcal{R}} | w \in \{0,1\}^*\}$. Recall that $w^{\mathcal{R}}$ means w written backwards. The informal description of the PDA follows.

> Begin by pushing the symbols that are read onto the stack. At each point nondeterministically guess that the middle of the string has been reached and then change into popping off the stack for each symbol read, checking to see that they are the same. If they were always the same symbol and the stack empties at the same time as the input is finished, accept; otherwise reject.

$L = \{ww^R : w \in \{0,1\}^*\}$. Prove L is c.f.

Proof

We will prove L is c.f. by constructing a PDA M that recognises L. Let M be defined as

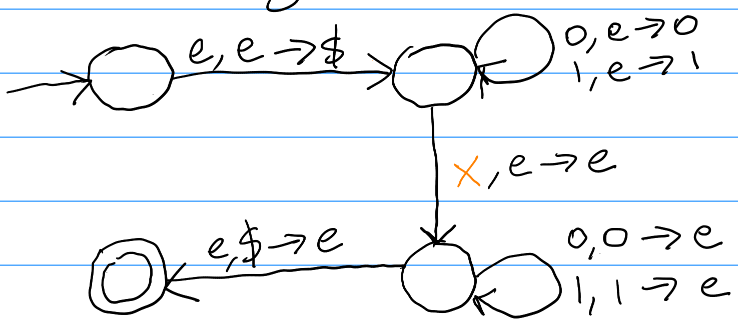


PDA M recognises L, therefore L is c.f.

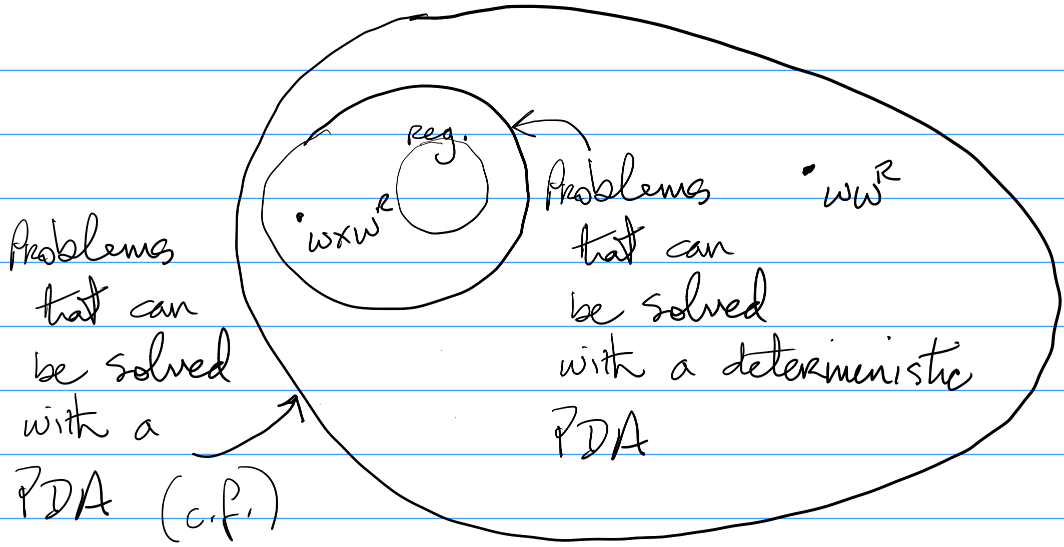
$L = \{wxw^R : w \in \{0,1\}^*\}$. Prove L is c.f.

Proof

We will prove L is c.f. by constructing a PDA M that recognises L. Let M be defined as



PDA M recognises L, therefore L is c.f.



FORMAL DEFINITION OF A PUSHDOWN AUTOMATON

The formal definition of a pushdown automaton is similar to that of a finite automaton, except for the stack. The stack is a device containing symbols drawn from some alphabet. The machine may use different alphabets for its input and its stack, so now we specify both an input alphabet Σ and a stack alphabet Γ.

At the heart of any formal definition of an automaton is the transition function, which describes its behavior. Recall that $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ and $\Gamma_\varepsilon = \Gamma \cup \{\varepsilon\}$. The domain of the transition function is $Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon$. Thus the current state, next input symbol read, and top symbol of the stack determine the next move of a pushdown automaton. Either symbol may be ε, causing the machine to move without reading a symbol from the input or without reading a symbol from the stack.

$P(Q \times \Gamma_\varepsilon) = 2^{Q \times \Gamma_\varepsilon}$

For the range of the transition function we need to consider what to allow the automaton to do when it is in a particular situation. It may enter some new state and possibly write a symbol on the top of the stack. The function δ can indicate this action by returning a member of Q together with a member of Γ_ε, that is, a member of $Q \times \Gamma_\varepsilon$. Because we allow nondeterminism in this model, a situation may have several legal next moves. The transition function incorporates nondeterminism in the usual way, by returning a set of members of $Q \times \Gamma_\varepsilon$, that is, a member of $\mathcal{P}(Q \times \Gamma_\varepsilon)$. Putting it all together, our transition function δ takes the form $\delta\colon Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \longrightarrow \mathcal{P}(Q \times \Gamma_\varepsilon)$.

DEFINITION 2.13

A ***pushdown automaton*** is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$, where Q, Σ, Γ, and F are all finite sets, and

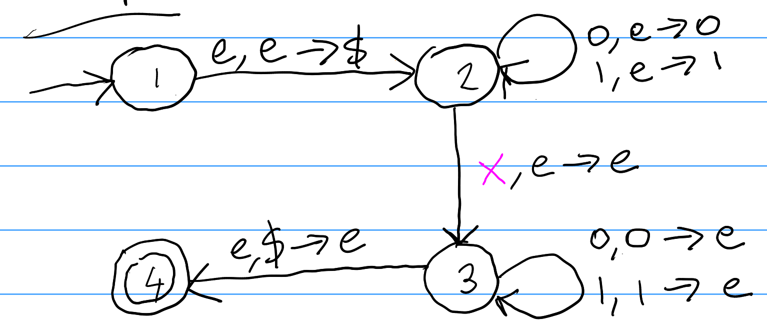
1. Q is the set of states,
2. Σ is the input alphabet,
3. Γ is the stack alphabet,
4. $\delta \colon Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \longrightarrow \mathcal{P}(Q \times \Gamma_\varepsilon)$ is the transition function,
5. $q_0 \in Q$ is the start state, and
6. $F \subseteq Q$ is the set of accept states.

Formal definition of a PDA computation (what is the mathematical definition of whether a PDA accepts a word or not).

A pushdown automaton $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ computes as follows. It accepts input w if w can be written as $w = w_1 w_2 \cdots w_m$, where each $w_i \in \Sigma_\varepsilon$ and sequences of states $r_0, r_1, \ldots, r_m \in Q$ and strings $s_0, s_1, \ldots, s_m \in \Gamma^*$ exist that satisfy the following three conditions. The strings s_i represent the sequence of stack contents that M has on the accepting branch of the computation.

1. $r_0 = q_0$ and $s_0 = \boldsymbol{\varepsilon}$. This condition signifies that M starts out properly, in the start state and with an empty stack.
2. For $i = 0, \ldots, m-1$, we have $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$, where $s_i = at$ and $s_{i+1} = bt$ for some $a, b \in \Gamma_\varepsilon$ and $t \in \Gamma^*$. This condition states that M moves properly according to the state, stack, and next input symbol.
3. $r_m \in F$. This condition states that an accept state occurs at the input end.

Example



w = 0110

w = e01e10e

1 2223334

w = 0101

w = no sequence of states possible