Expressive Power of Finite State Models

Nisarg Patel

ABSTRACT

We will look at extensions of finite state automata by augmenting the finite state machine with stacks, queue, counters and read/write tape. We also look at Petri Nets. Later, we compare the expressive power of the same models and form the expressive canvas.

We will start with the definitions of various models and languages associated with them.

1 Definitions

1.1 Finite Automata

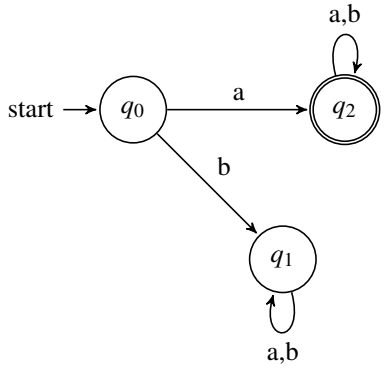
A *non-deterministic finite automata* (NFA) is 5-tuple $M = (Q, \Sigma, \Delta, I, F)$
where

- Q is the set of states;
- Σ is the finite alphabet, the *input alphabet*.
- Δ is the *transition relation*;
  $\Delta \subseteq (Q \times \Sigma) \times Q$;
- $I (\subseteq Q)$ is the set of *start states*;
- $F (\subseteq Q)$ is the set of *final states*;

A *deterministic finite automata* (DFA) is one where $|I| = 1$ and the relation δ is a function $\delta : Q \times \Sigma \to Q$, while the rest is the same as in non-deterministic finite automata.

The NFA are equivalent to DFA, as we will see later (refer to [Kozen, Lecture 6]) . The languages accepted by NFA (or DFA) are called Regular Languages (RL).

Below is an example of a finite automaton which accepts words starting with a, i.e, $\{aw | w \in \Sigma^*\}$.



1.2 Pushdown Automata

We will only look at non-deterministic Pushdown automata, and will refer to them as simply *Pushdown automata* (PDA).

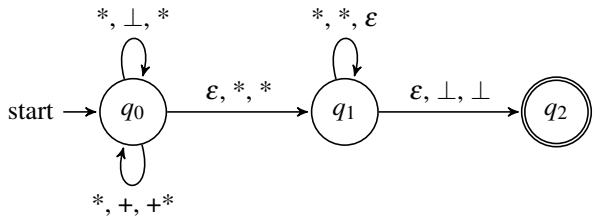
A PDA is 7-tuple $M = (Q, \Sigma, \Gamma, \delta, s, \perp, F)$
where

- Q is a finite set of states;

- Σ is the finite *input alphabet*;
- γ is the finite *stack alphabet*;
- δ is the finite transition relation,
  $\delta \subseteq (Q \times (\Sigma \cup \varepsilon) \times \Gamma) \times (Q \times \Gamma^*)$;
- $s \in Q$, the start state;
- $\perp \in \Gamma$, the initial stack symbol;
- $F \subseteq Q$, the set of final states;

The languages accepted by PDA are called Context Free Languages (CFL), because of the equivalence between PDA and Context Free Grammars.

A pushdown automaton accepting the language $\{w \cdot w^r \mid w \in \Sigma^*\}$, where w^r denotes the reverse of word w. The transition (q,a,b,q,w) is denoted by $q \xrightarrow{a,b,w} q'$. We will assume that if no transition can be applied at some point, then the machine halts and the input word is rejected.



Here, for the transition $(q_0, *, +, q_0, +*)$, means that $\forall * \in \Sigma$ *and* $\forall + \in \Gamma$, the above transition is valid. Similarly for other transitions.

1.3 Pushdown Automata with Regular Queries

Pushdown Automata with Regular Queries (PDARQ) is a model where we have a Pushdown Automata and a finite set of regular languages (on stack alphabet). The additional power is, the transitions can be guarded with conditions on stack word. For ex. let $R = \{L_1, L_2, .., L_n\}$ be the finite set of chosen regular languages.Then,

$(q,a,b) \xrightarrow{wb \in L_i \wedge L_j} (q',x)$ or

$(q,a,b) \xrightarrow{wb \notin L_i} (q',x)$; where wb is the word in the stack.

What this means is that, in state q, on reading a, if top symbol of the stack is b, and the stack word satisfies the guards, then the control moves to state to q', symbol b is popped and word x is pushed on top of the stack, so the new stack word is wx.

Formally, a PDARQ is 8-tuple $M = (Q, \Sigma, \Gamma, \delta, s, \perp, F, R)$
where

- Q is a finite set of states;
- Σ is the finite *input alphabet*;
- γ is the finite *stack alphabet*;
- δ is the finite transition relation,
  $\delta \subseteq (Q \times (\Sigma \cup \varepsilon) \times \Gamma \times \{0,1\}^n) \times (Q \times \Gamma^*)$;
- $s \in Q$, the start state;
- $\perp \in \Gamma$, the initial stack symbol;
- $F \subseteq Q$, the set of final states;
- R is a finite set of regular languages;

Let $R = \{L_1, L_2, L_3\}$. The transition $(q, a, b, (1,0,1)) \to (q', x)$ is valid if the machine is in state q, reads letter a, top most symbol of the stack is b and the stack word is in L_1 and L_3, but not in L_2. After the transition, the state is q', b is popped and the word x is pushed on top of the stack.

1.4 Queue Automata

A *Queue Automata* (QA) is finite state machine equipped with a two-way reading tape and a queue. Formally, it's an 8-tuple $M = (Q, \Sigma, \Gamma, \star, \delta, s, t, r)$ where

- Q is a finite set of states;
- Σ is the input alphabet;
- Γ is the queue alphabet;
- $\star \in \Gamma$ is the initial queue symbol;
- $\delta : Q \times (\Sigma \cup \{\vdash, \dashv\}) \times \{L, R\} \times \Gamma \longrightarrow Q \times \Gamma^* \times \{L, R\}$ is the transition function.
  The transition $(q, a, L, b) \to (q', x, R)$ is enabled if the machine is in state q and reads letter a and the left end of the queue is b. After taking the transition, the machine is in state q', b is popped from the queue on the left end and x pushed on the same end, and the tape head moves one cell to the right.
  Note that $\vdash$ is the left endmarker and $\dashv$ the right one, so the input word w is given as $\vdash w \dashv$ on the tape with the tape head on $\vdash$.
- s is the start state;
- t is the accept state;
- $r \neq t$ is the reject state;

The transitions are such that once it enters an accept or reject state, it stays there. The word is accepted if the machine enters accept state and rejected if it enters the reject state.

1.5 Queue Automata (with 1 queue symbol)

A Queue Automata with 1 queue symbol (QA (1 qs)) are queue automata, with the restriction that only 1 queue symbol is allowed, i.e, $\Gamma = \{\star\}$.

1.6 Queue Automata (with 1 queue symbol and Zero test)

For this model, we have a queue automata with 1 queue symbol and a zero test (QA (1 qs, ZT)). The zero test works in the following manner, we can check if the queue is in it's initial condition, i.e, the word in the queue is $\star$. Formally, the only change is in the transition function, which is now, $\delta : Q \times (\Sigma \cup \{\vdash, \dashv\}) \times \{L, R\} \times \Gamma \times \{0, 1\} \longrightarrow Q \times \Gamma^* \times \{L, R\}$, where the $\{0, 1\}$ component denotes the zero test, 1 if the word in the queue is $\star$, 0 otherwise.

As an example of the above model, consider $M = (\{q_0, q_1, q_2, q_3, q_4\}, \{a, b\}, \{\star\}, \delta, q_0, q_4, q_3)$, where following transitions are valid :
$(q_0, \vdash, R, \star, *) \mapsto (q_1, \star, L)$,
$(q_1, a, R, \star, *) \mapsto (q_1, \star^2, L)$,
$(q_1, b, R, \star, *) \mapsto (q_2, \varepsilon, L)$,
$(q_2, a, R, \star, *) \mapsto q_3$,
$(q_2, b, R, \star, *) \mapsto (q_2, \varepsilon, L)$,
$(q_2, \dashv, R, \star, 0) \mapsto q_3$,
$(q_2, \dashv, R, \star, 1) \mapsto q_4$.

Assuming that the machine halts and rejects the input, if no transition is applicable at any point, it can easily checked that the above automaton accepts the language $\{a^n b^n \mid n \in \mathbb{N}\}$.

1.7 1-way Queue Automata

A 1-way Queue Automata (QA (1-way)) is finite state equipped with a 1-way reading tape (which moves from left to right) and a queue. Formally, it's a 7-tuple $M = (Q, \Sigma, \Gamma, \star, \delta, s, F)$ where

- Q is a finite set of states;

- Σ is the finite input alphabet;
- Γ is the finite queue alphabet;
- $\star \in \Gamma$ is the initial queue symbol;
- $\delta : Q \times \Sigma \times \{L,R\} \times \Gamma \longrightarrow Q \times \Gamma^*$ is the transition function.
- $s \in Q$ is the start state;
- $F \subseteq Q$ is the set of final states

We can similarly define 1-way Queue Automata with 1 queue symbol (QA (1-way, 1 qs) and 1-way Queue Automata with 1 queue symbol and zero test (QA (1-way, 1 qs, ZT)).

1.8 k-Counter Automata

A k-counter automaton(k-CA) is a machine equipped with a two-way read-only input head and k integer counters. Each counter can store an arbitrary non- negative integer. In each step, the automaton can independently increment the counters by a constant, or decrement its counters by 1, and test them for 0 and can move its input head one cell in either direction. It cannot write on the tape.

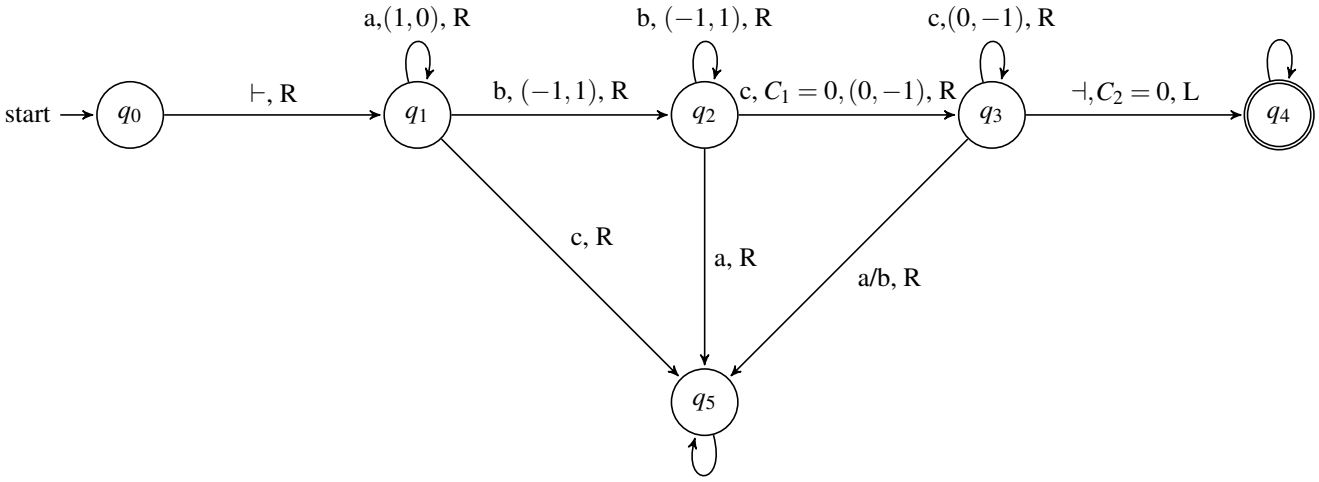
Formally, k-counter automata is 7-tuple $M = (Q, \Sigma, C, \delta, s, t, r)$ where

- Q is the finite set of states;
- Σ is the finite input alphabet;
- $C = \{C_1, C_2, .., C_n\}$ is the set of counters;
- $\delta : Q \times (\Sigma \cup \{\vdash, \dashv\}) \times \mathscr{P}(C) \longrightarrow Q \times S^{|C|} \times \{L,R\})$ is the transition function, where $S = \{-1, 0, 1, 2, ...\}$.
  The transitions are of the form $(q, a, T) \to (q', (a_1, a_2, .., a_n), L)$. This means that if the automaton is in state q and reads letter a, then it can take this transition if T is precisely the set of counters with value zero, and after the transition, the machine moves to state q', the reading head moves one cell to the left and the value of the counter C_i *changes* by a_i, i.e, if the previous value of C_i was l, then the value after the transition is $l + a_i$. Here $\mathscr{P}(C)$ denotes the superset of C.
- s is the start state;
- t is the accept state;
- $r \neq t$ is the reject state;

Similar to Queue Automaton, the input word w is given as $\vdash w \dashv$ and the transitions are such that once it reaches the final state or the reject state, it stays there. The word is accepted if the machine reaches an accept state and rejected if it reaches reject state.

Following is 2-Counter Automata accepting the language $\{a^n b^n c^n \mid n \in \mathbb{N}\}$. We will denote the transition $(q, a, \{C_1, c_2\}) \mapsto (q', (a_1, a_2, a_3), R)$ as $q \xrightarrow{a, C_1=0, C_2=0, (a_1,a_2,a_3), R} q'$. We may omit the tuple if all it's entries are zero, i.e, no change is made to any counters. We will assume that, if the value of a counter is 0, and a transition decreases it's value by 1, then the machine halts after the transition and rejects the word.

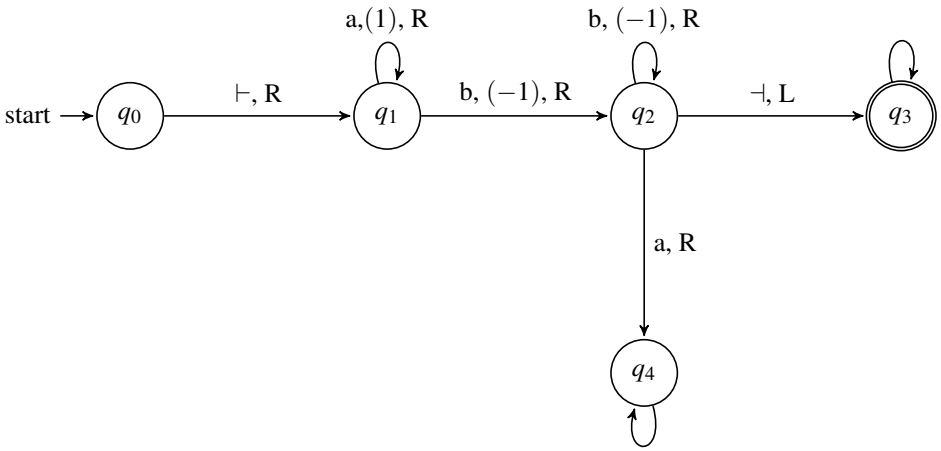
Formally, it's $M = (\{q_0, q_1, q_2, q_3, q_4, q_5\}, \{a, b, c\}, \{C_1, C_2\}, \delta, q_0, q_4, q_5)$, where the transition function δ is as shown below.



1.9 1-Counter Automata (without Zero Test)

A 1-Count er Automata without Zero Test (1-CA(w/o ZT) is, as the name suggests, 1-counter Automata without the power of the zero test. The only change this makes in the formal definition, is that, the transition function is $\delta : Q \times (\Sigma \cup \{\vdash, \dashv\}) \longrightarrow Q \times S \times \{L, R\}$, where S is the set defined earlier, in the above section.

Below is an example of 1-CA(w/o ZT) which accepts the language $\{a^m b^n \mid m \geq n\}$.
Formally, $M = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \{C_1\}, \delta, q_0, q_3, q_4)$, where δ is as shown below.



1.10 1-way k-Counter Automata

Similar to 1-way Queue Automata, 1-way k-counter automata (k-CA(1 way)) are identical to k-counter automata, the only difference being that k-CA(1-way) is equipped with 1-way input tape instead of a 2-way tape.

Formally, k-CA(1-way) is a 6-tuple $M = (Q, \sigma, C, \delta, s, F)$ where

- Q is the finite set of states;
- Σ is the finite input alphabet;
- $C = \{C_1, C_2, .., C_n\}$ is the set of counters;
- $\delta : Q \times \Sigma \times \mathscr{P}(C) \longrightarrow Q \times S^{|C|}$ is the transition function.
  The semantics of the transition function is the same as before.
- s is the start state;
- $F \subseteq Q$ is set of accept states;

1-way 1-Counter Automata (1-CA(1-way)) and 1-way 1-Counter Automata without zero test (1-CA(1-way, w/o ZT)) are defined in similar fashion.

1.11 Turing Machine

Formally, (deterministic one tape) Turing Machine (TM) is a 9-tuple $M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t, r)$ where

- Q is the finite set of states;
- Σ is the finite input alphabet;
- Γ is the finite tape alphabet containing Σ, i.e $\Sigma \subseteq \Gamma$;
- $\sqcup \in \Gamma - \Sigma$, the blank symbol;
- $\vdash \in \Gamma - \Sigma$, the left endmarker;
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$, the transition function.
  Here, $\delta(p, a) = (q, b, d)$ means, "When in state p scanning symbol a, write b on that tape cell, move the head in direction d, and enter state g." The symbols L and R stand for left and right, respectively.
- $s \in Q$, the start state;
- $t \in Q$, the accept state;
- $r \in Q$, the reject state, $r \neq t$;

Notice that for Queue Automata, Counter Automata and Turing machines, not every word is either accepted or rejected. It is possible, to have a Turing machine, which loops indefinitely on some word, without accepting or rejecting it. Same can be said for Queue Automata and Counter Automata. We will say that the language L is accepted if every word in L is accepted, and every word not in L is either rejected or the machine loops on it indefinitely. The languages accepted by Turing Machines are called Recursive Languages. For an example of a TM accepting $\{a^n b^n c^n \mid n \in \mathbb{N}\}$, refer to [Kozen, Lecture 28, Example 28.1].

1.12 Petri Nets

A net N is constituted by

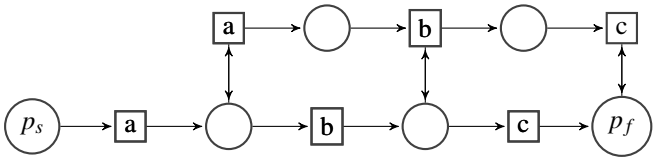
- a set of places, P;
- a set of transitions, T;
- a set of directed arcs, S
  $S \subseteq (P \times T) \cup (T \times P), satisfying$
  $S \cap (P \times P) = S \cap (T \times T) = \phi$

A marking of a net N is a mapping $m : P_N \to \mathbb{N}$.

A marked net is a net equipped with a marking, initial marking.

Let $\delta(\mu, \beta)$ denote the marking obtained by executing transition sequence $\beta \in T^*$ from marking μ, assuming β is enabled at μ. A language $L(\subseteq \Sigma^*)$ is an (L-type) Petri net language if there exists a Petri net structure (P, T, S, μ, F), a labelling of the transitions $\sigma : T \to \Sigma$, an initial marking μ, and a finite set of final markings F such that
$L = \{\sigma(\beta) \in \Sigma^* \mid \beta \in T^* and \delta(\mu, \beta) \in F\}$.

Below is the Petri Net which accepts the language $\{a^n b^n c^n \mid n \in \mathbb{N}\}$. The circles denote the places and the squares denote the transitions. The labels of the transitions are written inside the square. The initial marking is $\mu(p_s) = 1$ *and* $\mu(p) = 0 \; \forall p \neq p_s$. Similarly, only final marking is $\mu'(p_f) = 1$ *and* $\mu'(p) = 0 \; \forall p \neq p_f$.



We now define following sets of classes :

- Class 1 (C1) : NFA, DFA;
- Class 2 (C2) : QA(1 qs), 1-CA(w/o ZT);
- Class 3 (C3) : QA(1 qs,w/ ZT), 1-CA;
- Class 4 (C4) : PDA, PDARQ;
- Class 5 (C5) : Petri Nets
- Class 6 (C6) : PDAMS, 2-CA, k-CA(k $\geq$ 2), QA(2 qs), TM;
- Class 7 (C7) : QA(1-way, 1 qs), 1-CA(1-way, w/o ZT);
- Class 8 (C8) : QA(1-way, 1 qs,w/ ZT), 1-CA(1-way);

We will prove that the models in the same class are language-equivalent.

We will say $Ci \subseteq Cj$, if for every language accepted by an instance of a model in Ci, there is an instance of a model in Cj which accepts the same language.

We will say $Ci \subset Cj$, if $Ci \subseteq Cj$ and there is a language accepted by an instance of a model in Cj which cannot be accepted by any instance of any model in Ci.

We write $Ci \nsubseteq Cj$ as negation of $Ci \subseteq Cj$, i.e there is a language which can be accepted by an instance of a model in Ci, but not by any instance of any model in Cj.

2 Results

We will prove following two theorems :

- Theorem 1 : Models in the same class are equivalent.
- Theorem 2 :
  (a) $C1 \subset C7 \subset C8 \subset C4 \subset C6$
  (b) $C1 \subset C7 \subset C8 \subset C3 \subset C6$
  (c) $C1 \subset C7 \subset C2 \subset C3 \subset C6$
  (d) $C1 \subset C5 \subset C6$
  (e) $C4 \nsubseteq C5, C5 \nsubseteq C4$
  (f) $C2 \nsubseteq C4, C4 \nsubseteq C3$
  (g) $C8 \nsubseteq C2, C2 \nsubseteq C8$

2.1 Proof of Theorem 1:

2.1.1 C1 - NFA, DFA

It's clear that every DFA is an NFA. An NFA can be converted to a DFA by the famous subset construction (refer [Kozen, Lecture 6]).

2.1.2 C2 - QA(1 ss), 1-CA(w/o ZT)

We will first show that QA(1 ss) can simulate 1-CA(w/o ZT). Since we do not have the power of zero test, we push a $\star$ when there's an increment and pop one when decrementing to simulate a counter by queue. Formally, we have the same set of states for the two, and for every transition of the form $(q,a) \to (q',d,C_1,\{\})$(increment), we have the transition $(q,a,R,\star) \to (q',\star^2,d)$, where $a \in \Sigma \cup \{\vdash,\dashv\}$ and $d \in \{L,R\}$. For the transition of the form $(q,a) \to (q',d,\{\},C_1)$(decrement), we have the transition $(q,a,R,\star) \to (q',\varepsilon,d)$. This way we can simulate the counter by a queue.

Now, we want to show the other side, i.e, a counter can simulate a queue with only one symbol. The idea is the same as the previous case, that of simulating a queue by a counter. For every transition of the form $(q,a,L,\star) \to (q',\star^m,d)$ (where $d \in \{L,R\}, m \in \mathbb{N}$), we add the transition $(q,a) \to (q',(m-1),d)$. Note that, when $m = 0, \star^m = \varepsilon$, in which case we have to decrement the counter by 1, which is reflected above.

Thus, we have shown that 1-CA(w/0 ZT) can simulate QA(1 qs). Thus, both have the same expressive power.

We can also apply similar technique and prove the equivalence between models in C3, C7 and C8.

2.1.3 C4 - PDA, PDARQ

It is clear a PDARQ can model a PDA by taking $R = \emptyset$ and making no queries at all, while having the rest of the structure identical to the required PDA.

Interesting result to prove is that a PDA can simulate a PDARQ. We will give the proof idea for the case when $R = \{L\}$. The proof for more general case will follow in manner similar to the proof we give below.

Let the PDARQ be $M = (Q,\Sigma,\Gamma,\delta,s,\perp,F,R)$. Let L be recognized by the deterministic finite state automaton $\mathscr{A} = (Q^L,\Gamma,\Delta^L,p_0,F^L)$.

We will refer to states of $\mathscr{A}$ by $p_0,p_1,...$ etc. Also, for $p_i \in Q^L$, let $[p_i] = 0 \; if \; p_i \notin F \; and \; [p_i] = 1 \; if \; p_i \in F$.

We will simulate M by PDA M' defined as below:
$M' = (Q,\Sigma,\Gamma',s,(\perp,p_0,[\Delta^L(p_0,\perp)]),F)$ where

- $\Gamma' = \Gamma \times Q^L \times \{0,1\}$;
- $(\perp,p_0,[\Delta^L(p_0,\perp)])$ is the new initial stack symbol;
- for every transition of the form
  $\delta(q,a,\perp,0) = (q',\perp \cdot x_1 \cdot x_2 \cdots x_n)$, we have the transition (assuming $\perp \notin L$)
  $\delta'(q,a,(\perp,p_0,[p_1])) = (q',(\perp,p_0,[p_1]) \cdot (x_1,p_1,[p_2]) \cdot (x_2,p_2,[p_3]) \cdots (x_n,p_n,[p_{n+1}])$
  where $p_0 \xrightarrow{\perp} p_1 \xrightarrow{x_1} p_2 \xrightarrow{x_2} ....p_n \xrightarrow{x_n} p_{n+1}$ is the unique run of $\mathscr{A}$ on $\perp x_1x_2..x_n$
- for transitions of the form $\delta(q,a,b,i) = (q',x_1 \cdot x_2)$, where $b \in \Gamma \; and \; i \in \{0,1\}$, add the transitions
  $\delta'(q,a,(b,p,[\Delta^L(p,b)]) = (q',(x_1,p,[\Delta^L(p,x_1)]) \cdot (x_2,\Delta^L(p,x_1),[\Delta^L(\Delta^L(p,x_1),x_2)]))$
  Here, word of length 2, i.e, $x_1 \cdot x_2$, is chosen only for simplicity. For the transitions which have words of length more than 2 inserted, it can be done in similar fashion.

Intuitively, if the word in the stack of M is $\perp w_1w_2..w_n$, and the unique run of $\mathscr{A}$ on $w_1w_2..w_n$ is $p_0 \xrightarrow{w_1} p_1 \xrightarrow{w_2} ... \xrightarrow{w_n} p_n$, then we store the word $\perp' \; (w_1,p_0,[p_1]) \cdot (w_2,p_1,[p_2]) \cdot \cdot (w_n,p_{n-1},[p_n])$ in the stack of M'.

The last bit keeps the information if the word is in the language L or not. Also, note that the set of final states remain the same.

2.1.4 C6 - PDAMS, 2-CA, k-CA(k $\geq$ 2), QA(2 ss), TM

Refering to [Kozen, Lecture 30], we get the following results:

- A machine with multiple read/write tapes is equivalent to TM.
- A machine with two-way infinite read/write tape is equivalent to TM.
- A machine with a two-way, read-only input head and two stacks is equivalent TM.
- 2 counters can simulate a stack.

- More than 2 counters can be simulated by 2 counters.

Further, it's clear that a two-way read/write tape can simulate a stack.

Thus, from above results it's evident that 2-CA and k-CA(k ≥ 2) are equivalent. Also, PDA with 2 stacks can simulate a TM as well as 2-CA (and k-CA(k ≥ 2). Only thing that remains to show, is that TM can simulate a PDA with multiple stacks. But this can be done by having a machine with multiple read/write tapes (as many as stacks), and that is equivalent to the TM.

A queue automaton can simulate a pushdown automaton with 2 stacks. If the words in the two stacks are $\perp w_1$ and $\perp w_2$, then we store the word $w_2^r \perp w_1$. We simulate the first stack by operating on the left side of the queue, and the second one on the right. But since we are allowed to operate on only one end of the queue for a single transition, while both the stacks can be operated simultaneously, we make some minor modifications. Applying the same trick as earlier, for a transition of the form $(q,a,b_1,b_2) \rightarrow (q',x_1,x_2,d)$ we have the states $q \times \{0,1\} \times \{0,1\}$ and the transitions
$((q,0,0),a,R,b_1) \rightarrow ((q,1,1),x_1,R)$
$((q,1,1),*,R,c) \rightarrow ((q,1,0),c,L)$
$((q,1,0),*,L,b_2) \rightarrow ((q',0,0),x_2^r,d)$

This way, the queue automaton can simulate the pushdown automaton with 2 stacks. Note that it's necessary to have more than 1 symbol for the queue. On the other side, the two-way read/write tape can easily simulate a queue. Thus, the expressive power of QA(2 qs) is equal to that a TM.

Proof of Theorem 2:

We will use the following lemmas, which will help us prove Theorem 2 :

Lemma 1 : $\{a^n b^n \mid n \in \mathbb{N}\}$ ***and*** $\{a^m b^n \mid m \geq n\}$ ***are not Regular.***
Lemma 2 : 1-CA(w/o ZT) cannot recognize $\{a^n b^n \mid n \in \mathbb{N}\}$***.***
Lemma 3 : 1-CA cannot recognize $\{w \cdot w^r \mid w \in \Sigma^*\}$***.***
Lemma 4 : PDA cannot recognize $\{a^n b^n c^n \mid n \in \mathbb{N}\}$ ***and*** $\{a^m b^n c^l \mid m \geq n \geq l\}$***.***

It is also very easy to see that a finite machine with an auxillary structure can clearly simulate Finite Automata, except for the Petri Nets. The proof of Lemma 1 and 4 can be obtained by pumping argument for DFA and PDA (refer to [Kozen, Lecture 11] and [Kozen, Lecture 22] respectively).

(a) C1 $\subset$ C7 $\subset$ C8 $\subset$ C4 $\subset$ C6

It is clear that $C1 \subseteq C7$. 1-CA(1-way, w/o ZT) is a special case of 1-CA(1-way), thus $C7 \subseteq C8$. 1-CA(1-way) can be simulated by a PDA with unary stack alphabet and a fixed initial symbol, which is how we have defined the PDA. A two -way read/write tape can clearly simulate a stack, which means $C8 \subseteq C4 \subseteq C6$. Thus, we can say $C1 \subseteq C7 \subseteq C8 \subseteq C4 \subseteq C6$. To show $C1 \subset C7$, note that a 1-CA (1-way, w/o ZT) can accept the language $\{a^m b^n \mid m \geq n\}$. By Lemma 1, the above language is not regular. Thus, we have established that $C1 \subset C7$.

From Lemma 2, it follows that 1-CA(1-way, w/o ZT) cannot recognize $\{a^n b^n \mid n \in \mathbb{N}\}$, but 1-CA(1-way) can, which gives us $C7 \subset C8$. Also, it follows from Lemma 3, that 1-CA(1-way) cannot recognize $\{w \cdot w^r \mid w \in \Sigma^*\}$, but a PDA can, as shown in the example in Section 1.2. Thus $C8 \subset C4$. By Lemma 4 and the fact that a TM can recognize $\{a^n b^n c^n \mid n \in \mathbb{N}\}$, clearly $C4 \subset C6$. Thus, we have proved that $C1 \subset C7 \subset C8 \subset C4 \subset C6$.

(b) C1 $\subset$ C7 $\subset$ C8 $\subset$ C3 $\subset$ C6

We have already established that $C1 \subset C7 \subset C8$. Since 1-CA(1-way) is a special case of 1-CA, thus $C8 \subseteq C3$. Also, a two-way read write tape can simulate a counter, thus $C3 \subseteq C6$. To show $C8 \subset C3$, we argue that $\{a^n b^n c^n \mid n \in \mathbb{N}\}$ is the language that can be recognized by 1-CA, but not by 1-CA(1-way). We give a description of 1-counter automaton that accepts the above language. It is easy to check with finite state machine that the input is of the form $a^* b^* c^*$. We first check if the number of a's are the same as b's, using the Zero Test. If it's not, then we reject the word. Else, we start reading the word from the left end again, ignore the a's, and check if there are as many b's as c's. We accept or reject the word accordingly. The reason 1-CA(1-way) cannot accept the above language is because it has to empty the counter to check if there are as many a's as b's. Once this is done, there is no memory of how many a's (or b's) there were to check against c's. Also, since the reading head can only move left, it can't read the input more than once.

For $C3 \subset C6$, note that by Lemma 3, $\{w \cdot w^r \mid w \in \Sigma^*\}$ is the language which can be recognized by TM, but not by 1-CA. Thus, we have proven that $C1 \subset C7 \subset C8 \subset C3 \subset C6$.

(c) C1 $\subset$ C7 $\subset$ C2 $\subset$ C3 $\subset$ C6

Using the information from (a) and (b) above, it only remains to show that $C7 \subset C2 \subset C3$. It is obvious that $C7 \subseteq C2 \subseteq C3$, as 1-CA(1-way, w/o ZT) is a special case of 1-CA(w/o ZT), which in turn is a special case of 1-CA. For $C7 \subset C2$, we can argue in the same manner as above, that the language $\{a^m b^n c^l \mid m \geq n \geq l\}$ can be accepted by 1-CA(w/o ZT), but not by 1-CA(1-way, w/o ZT). This proves that $C7 \subset C2$. By Lemma 2 and the fact that 1-CA can accept $\{a^n b^n \mid n \in \mathbb{N}\}$, it is clear that $C2 \subset C3$. With this, we have proved that $C1 \subset C7 \subset C2 \subset C3 \subset C6$.

(d) C1 $\subset$ C5 $\subset$ C6 and (e) C4 $\not\subseteq$ C5, C5 $\not\subseteq$ C4

Refering to [Peterson, Section 6.6], Theorem 6-8 states that every regular language is a Petri Net Language. Theorem 6-11 states that every Petri Net Language is a context-sensitive Language. A contex-sensitive language is a language which can be accepted by a linear bounded automaton. Since every context-sensitive language can also be accepted by a Turing machine, we have $C1 \subseteq C5 \subseteq C6$. The example of a Petri Net given above accepts the language $\{a^n b^n c^n \mid n \in \mathbb{N}\}$, which can neither be accepted by Finite Automata nor PDA. This proves that $C1 \subset C5$ and $C5 \not\subseteq C4$. Theorem 6-9 states that no Petri Net can accept the language $\{w \cdot w^r \mid w \in \Sigma^*\}$. But this language can be accepted by PDA, as given in the example of a PDA above. This gives us $C4 \not\subseteq C5$ and $C5 \subset C6$. Thus we have proved the required results.

(f) C2 $\not\subseteq$ C4, C4 $\not\subseteq$ C3

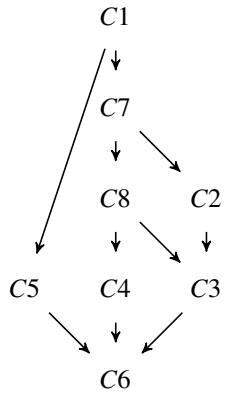
A 1-CA(w/o ZT) can recognize $\{a^m b^n c^l \mid m \geq n \geq l\}$ in the same manner as 1-CA recognizes $\{a^n b^n c^n \mid n \in \mathbb{N}\}$. By Lemma 4, PDA cannot recognize the above language. This gives us $C2 \not\subseteq C4$. $C4 \not\subseteq C2$ directly follows from Lemma 3.

(f) C2 $\not\subseteq$ C8, C8 $\not\subseteq$ C2

1-CA(1-way) can recognize $\{a^n b^n \mid n \in \mathbb{N}\}$, but 1-CA(w/o ZT) cannot, by Lemma 2. On the other side, 1-CA(w/o ZT) can recognize $\{a^m b^n c^l \mid m \geq n \geq l\}$, but 1-CA(1-way) cannot.

Conclusion

Denoting the relation $Ci \subset Cj$ as $Ci \rightarrow Cj$, we have proved the following picture.



References

1. [Kozen] Automata and Computability, Dexter C Kozen.
2. [Peterson] Petri Net Theory and Modeling of Systems, James L. Peterson.