# Automata Theorems

Issei Sakashita

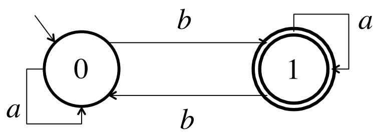
February 14, 2014

## 1 General Lemma

**Lemma 1** $\forall \delta \in D,\ \forall q \in Q,\ \forall u, v \in \Sigma^*, \delta^*(q, uv) = \delta^*(\delta^*(q, u), v).$

```
Lemma dstarLemma :
forall d : State -> Symbol -> State,forall q :State,
forall u v : string,
(dstar d q (u ++ v)) = (dstar d (dstar d q u) v).
Proof.
move => d q u v;move : q;elim : u.
by [].
move => a s H q;simpl.
by rewrite  H.
Qed.
```

## 2 Automaton $M_1$




**Theorem 1**

$$L(M_1) = \{w \in \Sigma^* \mid |w|_b \textit{ is odd number}\}.$$

We prove

1. $w \in \{w \in \Sigma^* \mid |w|_b \text{ is odd number}\} \to w \in L(M_1)$ and (**Lemma 2**)

2. $w \in L(M_1) \rightarrow w \in \{w \in \Sigma^* \mid |w|_b \text{ is odd number}\}$. (**Lemma 3**)

**Lemma 2**

$$\forall w \in \Sigma^* \ s.t.\ |w|_b \ is\ odd\ number\ \rightarrow w \in L(M_1).$$

```
Lemma m1_A_odd1 :
forall w : string, abword w -> odd (numb w) -> accept m1 w.
Proof.
simpl.
move => w H H'.
rewrite inE.
apply /eqP.
move : w H H'.
elim.
by [].
move => x s H0 H1 H2.
move : ((head_ab x s) H1).
move => Hab.
move : H1 H2.
case : Hab => [Ha | Hb].
rewrite Ha.
by simpl.

rewrite Hb.
simpl.
move => H Heven.
move : ((m1_A_odd1_lemma s) H).
case => H1 H2.
move : (H0 H) => H3.
clear H0 H2.
by move : (H1 Heven H3).
Qed.
```

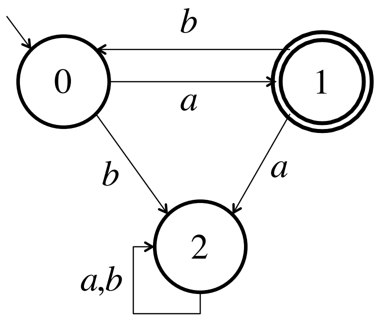
**Lemma 3**

$$\forall w \in L(M_1) \rightarrow |w|_b \ is\ odd\ number.$$

```
Lemma m1_A_odd2 :
forall w : string, abword w  -> accept m1 w -> odd (numb w).
Proof.
simpl.
move => Hx Hy Hz.
rewrite inE in Hz.
elim : Hx Hy Hz.
by simpl.
```

```
move => x s H1 H2.
move : ((head_ab x s) H2) => Hab.
move : H2.
case : Hab => H;rewrite H //=.
move => Hs H2.
move : ((m1_A_odd2_lemma s) Hs).
case => H3 H4.
apply H3.
by [].
by apply H1.
Qed.
```

## 3 Automaton $M_2$




**Theorem 2**

$$L(M_2) = \{(ab)^n a | n \in \mathbb{N}\}.$$

We prove

1. $w \in \{(ab)^n a | n \in \mathbb{N}\} \rightarrow w \in L(M_2)$ and $\cdots$ (**Lemma 4**)
2. $w \in L(M_2) \rightarrow w \in \{(ab)^n a | n \in \mathbb{N}\}$. (**Lemma 5**)

**Lemma 4**

$$\forall n \in \mathbb{N},\ a(ba)^n \in L(M_2).$$

```
Lemma abnaAm2b : forall n : nat, accept m2 (aban n).
Proof.
by elim;simpl.
Qed.
```

**Lemma 5**

$$\forall w \in L(M_2) \to \exists n \in \mathbb{N} \ s.t. \ w = a(ba)^n.$$

```
Lemma abnaAm2' :
forall w : string, abword w ->
accept m2 w -> exists n, w = aban n.
Proof.
elim.
by simpl.
rewrite /accept.
rewrite /m2.
move => x s H1 H2.
move : ((head_ab x s) H2) => Hab.
move : ((sub_abword_lemma x s) H2) => Hs.
move : (H1 Hs) => H.
move : H1 H2 => _ _.
move : ((m2lemma s) Hs);case=> K1;case => K2 K3.
case : Hab => Haob;rewrite Haob;simpl;move => H1;rewrite //.
move : (K2 H1) => H2.
destruct H2 as [i H3].
exists i.
by rewrite H3.
Qed.
```

# Appndix

**Lemma 6** `head_ab`

$$\forall x \in \Sigma, w \in \Sigma^*, xw \ \textit{is abword} \to x = \text{“}a\text{”} \ \textit{or} \ x = \text{“}b\text{”}.$$

```
Lemma head_ab :
forall (x : ascii)(w :string),
abword (String x w) -> x = "a"%char \/ x = "b"%char.
Proof.
move => x w H.
destruct x.
destruct b ;destruct b0;
destruct b1;destruct b2;
destruct b3;destruct b4;
destruct b5;destruct b6;
by [left | right].
Qed.
```

**Lemma 7** `Lemma m1_A_odd1_lemma`
$\forall w \in \Sigma^*$,

$$[|w|_b \textit{ is even } \rightarrow (|w|_b \textit{ is odd } \rightarrow \delta^*(0,w) = 1) \rightarrow \delta^*_{M_1}(1,w) = 1] \textit{ and}$$

$$[|w|_b \textit{ is odd } \rightarrow (|w|_b \textit{ is even } \rightarrow \delta^*(1,w) = 1) \rightarrow \delta^*_{M_1}(0,w) = 1].$$

```
Lemma m1_A_odd1_lemma :
forall w : string,  abword w ->
(~~ odd (numb w) ->
(odd (numb w) -> dstar m1_d 0 w = 1) -> dstar m1_d 1 w = 1) /\
(odd (numb w) ->
(~~ odd (numb w) -> dstar m1_d 1 w = 1) -> dstar m1_d 0 w = 1).
Proof.
elim.
by [].

move => x w H1 H2.
move : ((head_ab x w) H2) => Hab.
move : H2.
case : Hab => [Ha | Hb].
rewrite Ha //=.

rewrite Hb /=.
move => H2.
move : (H1 H2) => H.
clear H1.
case : H => Hx Hy.
by split;rewrite Bool.negb_involutive.
Qed.
```

**Lemma 8** `Lemma m1_A_odd2_lemma`
$\forall w \in \Sigma^*$,

$$[\delta^*_{M_1}(1,w) = 1 \rightarrow (\delta^*_{M_1}(0,w) = 1 \rightarrow |w|_b \textit{ is odd}) \rightarrow |w|_b \textit{ is even}] \textit{ and}$$

$$[\delta^*_{M_1}(0,w) = 1 \rightarrow (\delta^*_{M_1}(1,w) = 1 \rightarrow |w|_b \textit{ is even}) \rightarrow |w|_b \textit{ is odd}].$$

```
Lemma m1_A_odd2_lemma :
forall w : string,
abword w ->
(dstar m1_d 1 w == 1 -> (dstar m1_d 0 w == 1 -> odd (numb w)) -> ~~ odd (numb w)) /\
(dstar m1_d 0 w == 1 -> (dstar m1_d 1 w == 1 -> ~~ odd (numb w) ) -> odd (numb w) ).
Proof.
```

```
elim.
by simpl.
move => x s H Hs.
move : ((head_ab x s) Hs) => Hab.
move : Hs.
case : Hab => Hab; rewrite Hab //=.
move => Hs.
move : (H Hs).
case => H1 H2.
clear H Hs.
by split;rewrite Bool.negb_involutive.
Qed.
```

**Lemma 9** `sub_abword_lemma`

$$\forall x \in \Sigma, \forall w \in \Sigma^*, xw \textit{ is abword} \rightarrow w \textit{ is abword.}$$

```
Lemma sub_abword_lemma :
forall (x : ascii)(w : string), abword (String x w) -> abword w.
Proof.
move => x w H.
move : (head_ab x w H) => H'.
by case : H' => H'; rewrite H' in H; move : H;simpl.
Qed.
```

**Lemma 10** `m2lemma`
$\forall w \in \Sigma^*,$

$$[\delta^*_{M_2}(0, w) = \{1\} \rightarrow \exists n \in \mathbb{N}, w = a(ba)^n] \textit{ and}$$

$$[\delta^*_{M_2}(1, w) = \{1\} \rightarrow \exists n \in \mathbb{N}, w = (ba)^n] \textit{ and}$$

$$[\delta^*_{M_2}(2, w) = \{1\} \rightarrow \textit{ False}].$$

```
Lemma m2lemma : forall w : string, abword w ->
(dstar m2_d 0 w \in [:: 1] -> exists n : nat, w = aban n)
/\
(dstar m2_d 1 w \in [:: 1] -> exists n : nat, w = ban n)
/\
(dstar m2_d 2 w \in [:: 1] -> False).
Proof.
elim.
simpl.
move => _.
```

```
split.
by [].
split;move => H.
by exists 0.
by [].
move => a s H1 H2.
move : ((head_ab a s) H2) => Hab.
move : ((sub_abword_lemma a s) H2) => Hs.
move : (H1 Hs).
case => K1.
case.
move => K2 K3.
move : H1 H2 Hs => _ _ _.
case : Hab => H1;rewrite H1;simpl;split.
move => H2.
move : (K2 H2) => H.
destruct H as [i K].
exists i.
by rewrite K.
by split => H2;move : (K3 H2).
by move => H2; move : (K3 H2).
split;move => H2;rewrite //.
move : (K1 H2) => H.
destruct H as [i K].
exists (i.+1).
rewrite K.
by rewrite /aban.
Qed.
```