http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 4 (3): 274-277, 2005
ISSN 1812-5638

UM and Three Stack PDA

Umer Waqar Anis and Malik Sikander Hayat Khiyal
Department of Computer Science, Faculty of Applied Sciences,
International Islamic University, H-10, Islamabad, Pakistan

Abstract: This study gives an overview of Minsky theorem and then defines two machines. One with one stack and a moving head tape (UM). The second is a pushdown automaton with three stacks (3SPDA). Then we compare these new machines with a Turing Machine (TM).

Key words: Automata, pushdown automata, turing machine, UM

INTRODUCTION

Minsky theorem claims that a two stack pushdown automata (2SPDA) is equivalent to a Turing Machine (TM) that is, 2SPDA can perform any operation that can be performed by a TM. However, it is observed that in some cases this may not be applicable. In this study we discuss a similar scenario where 2SPDA cannot simulate a TM. Then we define two new machines. One with one stack and a moving head tap named as UM and the other machine with three stacks called three stack pushdown automata 3SPDA. Both machines simulate TM in any situation.

MINSKY THEOREM

Claim I: Any language accepted by a two stack pushdown automata (2SPDA) is accepted by some Turing Machine (TM) and any language accepted by a TM is accepted by some 2SPDA[1].

Claim II: Any language accepted by a PDA with a stack ($n \geq 2$) is also accepted by some TM[1].

The 2SPDA has three locations where it stores the information: Input tape, $Stack_1$, $Stack_2$. The input tape is read only in case of 2SPDA and can be read only once from left to right.

Construction: Let us define a machine M, a 6-tuple $M= (K,\Sigma,\Gamma,\delta,s,F)$, where:
K is set of finite states,
Σ is an alphabet of the input tape,
Γ is an alphabet of the stack symbol (assume same alphabet for both)
s is the starting state ($s \in K$)
F is set of accepting states ($F \subseteq K$)
δ is the transition function, a finite subset of $(K\times\Sigma\times\Gamma^*\times\Gamma^*)\times(K\times\Gamma^*\times\Gamma^*)$[2].

Comparison of a 2SPDA with turing machine: A TM is a 5-tuple $T= (K,\Sigma,\Gamma,q_0,\delta)$, where:
K is set of finite states (assumed not to contain h),
Σ is an alphabet of the input tape,
Γ is a finite tape alphabet with $\Sigma \subseteq \Gamma$
q_0 is the initial state ($q_0 \in K$)
δ is the transition function that maps $(K\times(\Gamma\cup\{\#\})$ to $(K\cup\{h\})\times(\Gamma\cup\{\#\})\times\{R,L,S\}$[3].

Initially

#	a	b	c	#

↑

Right move

#	a	b	c	#

↑

Left move

#	a	b	c	#

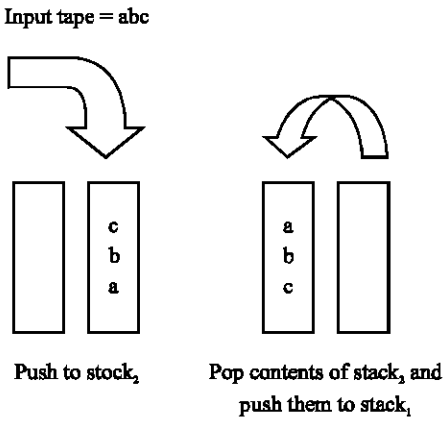
↑

To simulate a TM from 2SPDA

- The two stacks represent the TM ability to move left and right along a tape.
- The first stack represents the portion to the right side of the head of TM.
- The second stack represents the portion to the left side of the head of TM.
- To mark the boundaries of the stacks we push Δ symbol to the stacks.
- Now stacks are ready to simulate TM.
- The input tape is pushed to the second stack.

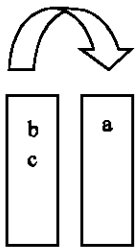
Corresponding Author: Dr. Malik Sikander Hayat Khiyal, Department of Computer Science, Faculty of Applied Sciences, International Islamic University, H-10, Islamabad, Pakistan
Tel: +92 51 9257951

- The input data is in reverse order.
- To correct this reversal problem, push each symbol from $stack_2$ to $stack_1$.
- Now top of $stack_1$ contains the first/start symbol of the input tape.



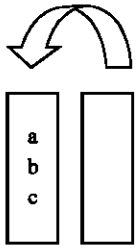
Now top of $stack_1$ contains 'a' on the top which is the start symbol.

Right move: To simulate moving right on the TM tape, the 2SPDA pops the top of $stack_1$ and pushes it to $stack_2$. Whenever a write is required on the right hand of the tape push a character on the right stack that is $stack_2$.



Top of $stack_1$ is pushed to $stack_2$, thus a right move has occurred. Therefore the next symbol on the top of $stack_1$ is b, which is also the next symbol on the input tape.

Left move: To simulate moving left on the TM tape, the 2SPDA pops the top of $stack_2$ and pushes it to $stack_1$.



Top of $stack_2$ is pushed to $stack_1$, thus a left move has occurred. Therefore the next symbol on the top of the $stack_1$ is 'a', which is also the previous symbol on the input tape.

Thus from the above discussion it can be deduced that a two 2SPDA is equivalent to a TM.

Limitation: A problem arises when we use a 2SPDA to simulate a TM that performs copy and paste operation. The language $\{ww: w \in \{a,b\}^*\}$, that is to append the same string at the end of the string.

A TM can perform the operation in the following way for a given tape containing ab.

- Read the symbol from the tape.
- Replace the symbol with a Δ.
- Move to the right end.
- Place the symbol.
- Move back (left) to the first occurrence of Δ.
- Replace the symbol.
- Move right.
- And repeat the above steps until the entire input string is copied.

However, 2SPDA cannot perform the same task because the input is exhausted when tape is read once. We will not use the stacks for sorting the symbols (for copying purpose) because if the symbol is stored on the stack then the left and right movement will be effected. Therefore we cannot generate another string in the same order as that of input string (in this particular case). Therefore Claim I does not hold true for this particular case. Furthermore Claim II is also not satisfied as the number of stacks is two and the 2SPDA cannot generate a ww from a given string w.

SOLUTION

The above stated problem can be solved in various ways. However we define two new machines to solve the problem. They are:

- Single Stack Machine with an input tape having bi-directional head movement (UM).
- Three Stack Pushdown Automata (3SPDA).

UM

We construct a machine with single stack and assume that the tape head can move in both directions (bi-directional). This machine UM is a 7-tuple $M=(K,\Sigma,\Gamma,\delta,s,F,H)$, where:

K is set of finite states,
Σ is an alphabet of the input tape,
Γ is an alphabet of the stack symbol (assume same alphabet for both)
s is the starting state ($s \in K$)
F is set of accepting states ($F \subseteq K$)
H defines the head movement that is {R,L,S}, where, R is right move, L is left move and S denotes a stationary (no movement of head).
δ is the transition function, a finite subset of $(K \times \Sigma \times \Gamma^* \times H) \times (K \times \Gamma^*)$.
For example $((p,a,\alpha,R), (q,\beta)) \in \delta$ then M is in state p when the input is 'a' and top of stack contains α. The tape head moves right and results in state change such that q with β on top of the stack. Such a pair $((p,a,\alpha,R), (q,\beta))$ is called a transition of M where:
p is current state,
a is input tape yet to be read,
α top of stack when in state p,
R is move the head right one position before changing the state,
q is next state
β is top of stack when in state q.
Δ is used to mark the end of stack, $ marks the start of tape and # marks the end. To push a symbol is to add it to the top of the stack, to pop a symbol is to remove it from the top of the stack; following functions are defined for the new machine. Whenever the input string or stack top is represented by e, it denotes that it is not to be consulted for the transition. For example, the transition:

- ((p,a,e,R), (p,a)) push 'a' on the stack and move right.
- ((p,a,b,R), (p,a)) push 'a' on the stack when top of stack is 'b' and move right.
- ((p,a,b,L), (p,e)) pop 'b' from stack when tape symbol is 'a' and move left.
- ((p,e,a,S), (p,e)) pop 'a' from stack and no head movement.
- ((p,e,e,R), (q,e)) state transition from p to q.
- ((p,e,a,R), (q,e)) state transition when top of stack is 'a'.
- ((p,a,b,R), (q,e)) state transition when top of stack is 'b' and input is 'a'.
- ((p,a,b,R), (q,e)) state transition when top of stack is 'b' and input is 'a' and push 'e' (no push).

Reflexive and transitive closure: If (p,x,α,m) and (q,y,ζ,m) are configuration of M (where m is any move on the tape, i.e, R,L,S), then (p,a,α,R) yields (q,ζ) in one step such that $(p,a,\alpha,R) \vdash_M (q,y,\zeta)$ if there is a transition $(p,x,\beta,R), (q,\gamma)) \in \delta$ such that x=ay, $\alpha=\beta\eta$ and $\zeta=\gamma\eta$ (for some $\eta \in \Gamma^*$).

Transitive closure of $\vdash_M$ is denoted by $\vdash_M^*$. M is said to accept a string $w \in \Sigma^*$ if and only if $(p,w,e,R) \vdash_M^* (q,e,e)$ for some state $q \in F$ and there is a sequence of configurations $C_0, C_1, \ldots, C_{n-1}, C_n$ where $(n \geq 0)$ such that $C_0 \vdash_M C_1 \vdash_M \ldots \vdash_M C_{n-1} \vdash_M C_n$, $C_0 = (p,w,e,R)$ and $C_n = (q,e,e)$. Any sequence of such configuration is called a computation by M.

Problem revisited: Use UM to simulate a TM that perform copy and paste operation $\{ww : w \in \{a,b\}^*\}$ that is to append the same string at the end of the string.

We are given with a string w=$abc# as input and we have to generate ww. The UM will perform the operation in the following way:

- Move to the right of input until # is reached.
- Move left.
- Read the symbol, move left and push the symbol on the stack (Repeat until $ is reached).
- Move to the right of input until # is reached.
- Move left.
- Read the symbol, move left and push the symbol on the stack (Repeat until $ is reached).

Now the stack contains ww that is, abcabc. The transitions of δ are as follows:

1. $((q_0,\$,e,R), (q_1,e))$
2. $((q_1,e,e,R), (q_1,e))$
3. $((q_1,e,\#,L), (q_2,e))$
4. $((q_2,a,e,L), (q_2,a))$
5. $((q_2,b,e,L), (q_2,b))$
6. $((q_2,e,e,L), (q_2,e))$
7. $((q_2,\$,e,R), (q_3,e))$
8. $((q_3,e,e,R), (q_3,e))$
9. $((q_3,e,\#,L), (q_4,e))$
10. $((q_4,a,e,L), (q_4,a))$
11. $((q_4,b,e,L), (q_4,b))$
12. $((q_4,e,e,L), (q_4,e))$
13. $((q_4,\$,e,S), (q_f,e))$

How UM works

No.	Initial state	Input Σ	Stack Γ	Move	New state	Transition δ
1	q_0	$\underline{\$}abc\#	Δ	$\$\underline{a}bc\#	q_1	1
2	q_1	$\$\underline{a}bc\#	Δ	$\$a\underline{b}c\#	q_1	2
3	q_1	$\$a\underline{b}c\#	Δ	$\$ab\underline{c}\#	q_1	2
4	q_1	$\$ab\underline{c}\#	Δ	$\$abc\underline{\#}	q_1	2
5	q_1	$\$abc\underline{\#}	Δ	$\$ab\underline{c}\#	q_2	3
6	q_2	$\$ab\underline{c}\#	c	$\$a\underline{b}c\#	q_2	6
7	q_2	$\$a\underline{b}c\#	b	$\$a\underline{b}c\#	q_2	5
8	q_2	$\$a\underline{b}c\#	a	$\$\underline{a}bc\#	q_2	4
9	q_2	$\$\underline{a}bc\#	e	$\$\underline{a}bc\#	q_3	7
10	q_3	$\$\underline{a}bc\#	e	$\$a\underline{b}c\#	q_3	8
11	q_3	$\$a\underline{b}c\#	e	$\$ab\underline{c}\#	q_3	8
12	q_3	$\$ab\underline{c}\#	e	$\$abc\underline{\#}	q_3	8
13	q_3	$\$abc\underline{\#}	e	$\$ab\underline{c}\#	q_4	9
14	q_4	$\$ab\underline{c}\#	c	$\$a\underline{b}c\#	q_4	12
15	q_4	$\$a\underline{b}c\#	b	$\$\underline{a}bc\#	q_4	11
16	q_4	$\$\underline{a}bc\#	a	$\$\underline{a}bc\#	q_4	10
17	q_4	$\$\underline{a}bc\#	e	$\$\underline{a}bc\#	q_f	13

THREE STACK PDA

We define a Three Stack PDA (3SPDA) in the following way:
Let this machine M be a 6-tuple $(K,\Sigma,\Gamma,\delta,s,F)$, where:
K is set of finite states,
Σ is an alphabet of the input tape,
Γ is an alphabet of the stack symbol (assume same alphabet for both)
s is the starting state ($s\in K$)
F is set of accepting states ($F\subseteq K$)
δ is the transition function, a finite subset of $(K\times\Sigma\times\Gamma^*\times\Gamma^*\times\Gamma^*)\times(K\times\Gamma^*\times\Gamma^*\times\Gamma^*)$.

We are given with a string w=abc as input and we have to generate ww. The 3SPDA will perform the operation in the following way:

- Read the input w into $stack_2$ and $stack_3$.
- Pop the contents of $stack_3$ and push them into $stack_1$ so that original order of input tape is obtained.
- Pop the contents of $stack_2$ and push them into $stack_1$.
- Now $stack_1$ contains our required string ww.

The transition of δ are as follow:

1. $((q_0,a,e,e,e), (q_0,e,a,a))$
2. $((q_0,b,e,e,e), (q_0,e,b,b))$
3. $((q_0,c,e,e,e), (q_0,e,c,c))$
4. $((q_0,e,e,e,e), (q_1,e,e,e))$
5. $((q_1,e,e,a,e), (q_1,a,e,e))$
6. $((q_1,e,e,b,e), (q_1,b,e,e))$
7. $((q_1,e,e,c,e), (q_1,c,e,e))$
8. $((q_1,e,e,e,e), (q_2,a,e,e))$
9. $((q_2,e,e,e,a), (q_2,a,e,e))$
10. $((q_2,e,e,e,b), (q_2,b,e,e))$
11. $((q_2,e,e,e,c), (q_2,c,e,e))$
12. $((q_2,e,e,e,e), (q_f,e,e,e))$

How 3SPDA works

No.	Initial state	Input Σ	$Stack_1$ Γ_1	$Stack_2$ Γ_2	$Stack_3$ Γ_3	New state	Transition δ
1	q_0	abc	e	e	e	-	-
2	q_0	bc	e	a	a	q_0	1
3	q_0	c	e	ba	ba	q_0	2
4	q_0	e	e	cba	cba	q_0	3
5	q_0	e	e	cba	cba	q_1	4
6	q_1	e	c	ba	cba	q_1	7
7	q_1	e	bc	a	cba	q_1	6
8	q_1	e	abc	e	cba	q_1	5
9	q_1	e	abc	e	cba	q_2	8
10	q_2	e	cabc	e	ba	q_2	11
11	q_2	e	bcabc	e	a	q_2	10
12	q_2	e	abcabc	e	e	q_2	9
13	q_2	e	abcabc	e	e	q_f	12

CONCLUSIONS

From the above discussion we conclude the following results:

- A UM is equivalent to a TM. That is any language accepted by a UM is accepted by some TM and any language accepted by a TM is accepted by some UM.
- Any language accepted by a 3SPDA is accepted by some TM and any language accepted by a TM is accepted by some 3SPDA (modification of Claim I).
- Any language accepted by a 3SPDA is accepted by some UM and any language accepted by a UM is accepted by some 3SPDA.
- Any language accepted by a PDA with a stack ($n\geq 2$) is also accepted by some TM (modification of Claim II).
- A 3SPDA takes more computational time than TM and UM as it requires reading the input tape completely once before performing any operation.

REFERENCES

1. Cohen, D.I.A., 1991. Introduction to Computer Theory. John Wiley and Sons, Inc. USA.
2. Harris, A., S. Michael and J. Roberts, 2001. Pushdown automata with two stacks. Report, Denison University, Granville, Ohio, USA.
3. Khiyal, M.S.H., 2004. Theory of Automata and Computation. National Book Foundation, Islamabad, Pakistan.