Lecture 27   Ana. of Alg (1/2552)   16/9/09

5.2 Turing Machines

A Turing machine ประกอบด้วย

a) **tape** มีขอบเขตอนันต์ทั้งสองด้าน (ทางซ้าย / ทางขวา)

ที่ซึ่งสามารถแบ่งออกมาในรูป squares ภายใต้การกำหนด หมายเลข ดังรูป 5.2.1

...	⊔	H	E	L	L	O	⊔	...
...	-3	-2	-1	0	1	2	3	...

รูป 5.2.1 : A Turing machine tape

ในแต่ละ square จะบรรจุ symbol (a single character) จาก Alphabet (character set) ที่ machine recognizes

i.e. เราสามารถสมมุติว่า Alphabet บรรจุ '0', '1' และ '⊔' blank

b) a **tape head** ที่สามารถ reading a symbol จาก a square บน tape หรือสามารถ writing a symbol บน a square หรือ moving ไปยังตำแหน่งอื่นที่อยู่ห่างจาก

square ปัจจุบัน โดย moving ทีละ square อาจจะเป็น
ทางซ้าย หรือขวา ของ ตำแหน่งปัจจุบัน

c) a finite list ของ states ที่ซึ่ง ทุกช่วงขณะ machine จะอยู่ใน state ใด state หนึ่ง ของ states เหล่านี้

โดย states ทั้งหมดของ machine ประกอบด้วย

- state ปกติ (regular states): $q_1, \dots q_s$

และ

- special states :

q_0: the initial state

q_Y: the final state ใน a problem ที่ซึ่ง คำตอบ เป็น 'yes'

q_N: the final state ใน a problem ที่ซึ่ง คำตอบ เป็น 'No'

d) a program (หรือ program module) ที่เป็นหน่วยที่ควบคุม machine เพื่อทำงาน ขั้นตอนต่างๆ ของ งานที่ต้องการ

ต่อไปจะพิจารณา the program module โดยสมมุติว่า ณ ช่วงขณะหนึ่ง machine อยู่ที่ state q (ที่ไม่ใช่ q_Y หรือ q_N)

และ symbol ที่เพิ่งได้อ่าน tape คือ 'symbol' แล้วเกิดอะไร

the pair (q, symbol) กำหนด program module จะ decide

(i) state ใด ที่ machine จะไป i.e q' และ:

(ii) symbol ใด ที่ machine จะเขียนทับ(write) ลงบน square ใน tape ณ ตำแหน่งปัจจุบัน และ:

(iii) tape head จะเคลื่อนที่ (move) หนึ่ง square ไปทางซ้าย หรือ ขวา

ดังนั้น 1 step ของ a program module จะสามารถแทนได้ในรูปของฟังก์ชัน ได้ดังนี้

$$\Phi : \{q_1, \cdots, q_s, q_0, q_Y, q_N\} \times A \cup \{\sqcup\} \rightarrow \{q_1, \cdots, q_s, q_0, q_Y, q_N\} \times A \cup \{\sqcup\} \times \{-1, +1\}$$

คือ

$$(\text{state}, \text{symbol}) \mapsto (\text{newstate}, \text{newsymbol}, \text{increment}) \quad (S2.1)$$

ที่ ถ้า state q_Y หรือ q_N การทำงานจะเสร็จสิ้น และ machine หยุดการทำงาน

ใน machine เราสามารถแบ่งออกเป็น 2 ส่วน ส่วนแรกคือ

hardware และส่วนสอง คือ software

หน้าที่ของ programmer การเขียน a programe สำหรับ a Turing machine เพื่อบอกให้ machine ทำแต่ละขั้นตอน ตลอดจนการเปลี่ยน state ที่สอดคล้องกับ (5.2.1)

A Turing machine คล้ายกับ a table ที่ชี้ทุกๆ pair (state, symbol) machine จะต้องทำอะไร โดยที่ programer เป็นผู้กำหนด newstate และ new symbol ตลอดจน increment

การเริ่มต้นทำงานใน a Turing machine จะมีการป้อน input string $n \in A \cup \{\sqcup\}^B$ เมื่อ $A \cup \{\sqcup\}^n$ เป็นเซตของ word ที่มี length เป็น B ซึ่ง n จะแสดงถึงปัญหาที่เกิดต้องการ solve และทำการ write n ใน squares 1, 2, ..., B ของ tape ขณะที่ tape head จะอยู่ที่ตำแหน่ง 1 ของ square และ machine จะถูก put into state q_0 โดย

program module ที่ programmer จัดเตรียม จะถูกใส่เข้าไป ใน slot จากนั้นการทำงานจะเริ่มที่

machine อ่าน symbol ใน square 1 ขณะอยู่ที่ state q_0 และเมื่ออ่าน symbol แล้ว machine จะ consult the program module เพื่อค้นหาว่าจะทำอะไรต่อไป

คำสั่งใน program จะเขียน (write) a newsymbol ใน square 1 จากนั้น move head ของ Tape ไปที่ square 0 หรือ square 2 อย่างใดอย่างหนึ่ง แล้วเข้าสู่ newstate q' กระบวนการเหล่านี้จะมีการทำซ้ำไปเรื่อยๆ จนกระทั่ง machine จะเข้าสู่ state q_Y หรือ state q_N ณ state นี้ การทำงานจะสิ้นสุดลง ตามผลลัพธ์ของ decision problem i.e Yes / no

ต.ย.ของ a Turing machine ที่สามารถจำลองขึ้นได้ ได้แก่ pidgin-Pascal simulation ของ a Turing Machine โดย program นี้ประกอบด้วย 2 ส่วนหลัก

The procedure turmach ใช้สำหรับรับ input
B
a string $x \in A \cup \{\sqcup\}$ และให้ output เป็น set ของ Boolean variable i.e. accept เป็น Boolean variable โดย output จะขึ้นกับ state ของการสิ้นสุดของ machine

(i.e. q_Y หมายถึง accept เป็น True, q_N หมายถึง accept เป็น False)

โดยกระบวนการนี้จะอยู่ส่วนของ hardware part ของ Turing machine

Procedure gonextto เป็น program module ของ machine ช่วยเหลือเคลื่อนที่ไปตามลักษณะของ Job หรือปัญหาที่ต้องการ solve โดย input ของ gonextto ประกอบด้วย state ปัจจุบันของ machine และ symbol ที่เพิ่งอ่านบน tape ได้ outputs ประกอบด้วย newstate, new symbol, increment position โดย tape head จะ write new symbol บน current square จากนั้น increment (±1)

Procedure turmach (B: integer; x: array [1...B]; accept: Boolean);

{ simulates Turing machine action บน input string x

ความยาว B }

{ write input string บน tape ที่มี B squares ตั้งแต่

square 1 ถึง square B }

for square := 1 to B do

tape[square] := n[square];

{ บันทึก ของ แต่ ของการ write บน tape }

q_0

~~state~~ := 0; square := 1;

q_Y q_N

while state ≠ 'Y' และ state ≠ 'N' do

{ read symbol ใน current tape square }

If square < leftmost หรือ square > rightmost

then symbol := '⊔'

else symbol := tape[square]

{ ask program module สำหรับ state transition }

gonextto(state, symbol, newstate, newsymbol, increment);

```
        state := newstate;
    {update ข้อมูล และ: write new symbol}
     if square > rightmost then leftmost := square;
        tape[square] := new symbol;
    {move tape head}
        square := square + increment;
    end; {while}
    accept := {state = 'Y', state = 'N'}
end. {turmach}
```

5.3 Cook's Theorem

NP-complete problems ว่าเป็น hardest problem ใน NP เพื่อ ถ้า Q' เป็น decision problem ใดๆ ใน NP และ Q เป็น อัน NP-complete problem และ instance ใดๆ ของ Q' เป็น polynomially reducible ต่อ instance ของ Q

มีอีก NP-complete problem ที่ซึ่ง ยังไม่สามารถหาได้ ความกระทั่ง ว่า หรือไม่ any single problem ควรจะเป็น key ต่อ polynomial time solvability ของ ทุกปัญหาใน class NP ที่ซึ่งเกิดหลากหลายได้ถูกพบใน discrete structure ได้แก่ graph, networks, games ตลอดจนกับ ปัญหาที่เกี่ยวข้อง กับ optimization problem เช่น algebraic structures, formal logic และ อื่นๆ เป็นต้น

ต่อไป เป็น satisfiability problem, เป็น the first problem ที่ได้รับการพิสูจน์แล้วเป็น NP-Complete โดย stephen Cook ใน 1971

เริ่มต้นด้วย a list ของ Boolean variable $x_1, \dots, x_n$

ให้ $X = \{x_1, \dots, x_n\}$ เมื่อ x_i = True หรือ False อย่างใดอย่างหนึ่ง

not x_i

ให้ A literal แทนด้วย x_i หรือ $\bar{x}_i$ เมื่อ $x_i \in X$

ดังนั้นจะมี possible literal จำนวน $2n$ ตัว

ให้ A clause แทนด้วย a set ของ literals

$$L = X \cup \{\bar{x} : x \in X\}$$

กฎของ game ประกอบด้วย เรา assign ค่า True (T)

หรือ False (F) ให้กับ ตัวแปร แต่ละตัว

แต่ละ clauses จะมีการสืบทอด ค่าความจริง จาก

process ซึ่ง สามารถ หาได้ ดังต่อไปนี้

A clause มีค่าเป็น 'T' ก็ต่อเมื่อ at least one ใน

literals ใน clause มีค่าเป็น 'T' และ clause มี

ค่าเป็น 'F' สำหรับกรณีอื่น

<u>Def</u> A set ของ clauses เป็น satisfiable

ถ้า มี การ assignment ของ truth values ต่อ ตัวแปรต่างๆ

ที่ทำให้ ทุก clauses เป็น true

<u>The satisfiability problem</u> (SAT)

กำหนด a set ของ clauses ปัญหาคือ มี a set of

truth values (= T หรือ F), หนึ่ง เซต สำหรับ แต่ละ variable

ที่ซึ่ง ทุก clause ประกอบด้วย หนึ่ง literal ที่มีค่าเป็น T

Ex พิจารณา the set x_1, x_2, x_3 ของ variables

เกิดขึ้นได้ list ของ 4 clauses ดังต่อไปนี้

$$\{\overset{T}{x_1}, \overset{F}{\bar{x}_2}\}, \{\overset{T}{x_1}, \overset{F}{x_3}\}, \{\overset{T}{x_2}, \overset{T}{\bar{x}_3}\}, \{\overset{F}{\bar{x}_1}, \overset{F}{x_3}\}$$

ถ้า true values ของ (x_1, x_2, x_3) เป็น (T, T, F)

แต่ใน 4 clauses เกิดได้ true values เป็น

$(T, T, T, F,)$

ทำให้เกิดว่า $\{\{x_1, \bar{x}_2\}, \{x_1, x_3\}, \{x_2, \bar{x}_3\}, \{\bar{x}_1, x_3\}\}$ ไม่เป็น satisfiable