BM 305 Biçimsel Diller ve Otomatlar (Formal Languages and Automata)

Hazırlayan: M.Ali Akcayol
Gazi Üniversitesi
Bilgisayar Mühendisliği Bölümü

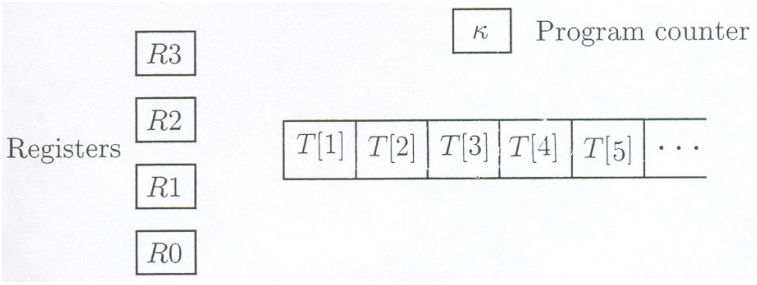
Konular

- Random Access Turing Machines
- Nondeterministic Turing Machines
- Unrestricted Grammars
- Undecidability
- The Church-Turing Thesis

Random Access Turing Machines

- Şimdiye kadar görülen Turing makineleri sıralı (sequential) hafızaya sahiptir.
- Buna karşılık gerçek bilgisayarlar rastgele erişimli (random access) hafızaya sahiptir.
- Rastgele erişimli Turing makinesi (Random Access Turing Machine-RATM) sabit sayıda kayıtçıya (registers) ve tek yönlü sınırsız tape ünitesine sahiptir.
- Her register ve tape hafıza alanı bir sayıyı saklar.
- Bu makine bir programa ait instruction'ları gerçek bilgisayarlardaki gibi sırayla çalıştırır.
- Çalıştırılan programda bir sonraki instruction adresini tutan program sayacı (counter) vardır.

Random Access Turing Machines



- Başlangıçta program counter 1, diğer register'lar 0 değerine sahiptir.
- RATM başlangıçta ilk instruction'la çalışmaya başlar.
- Her instruction'la register içerikleri ve/veya program counter ve/veya tape içeriği değişebilir.

Random Access Turing Machines

RATM için instruction kümesi

Instruction	Operand	Semantics
read	j	$R_0 := T[R_j]$
write	j	$T[R_j] := R_0$
store	j	$R_j := R_0$
load	j	$R_0 := R_j$
load	$= c$	$R_0 := c$
add	j	$R_0 := R_0 + R_j$
add	$= c$	$R_0 := R_0 + c$
sub	j	$R_0 := \max\{R_0 - R_j, 0\}$
sub	$= c$	$R_0 := \max\{R_0 - c, 0\}$
half		$R_0 := \lfloor \frac{R_0}{2} \rfloor$
jump	s	$\kappa := s$
jpos	s	if $R_0 > 0$ then $\kappa := s$
jzero	s	if $R_0 = 0$ then $\kappa := s$
halt		$\kappa := 0$

Random Access Turing Machines

RATM için instruction kümesi

- *j register numarasını gösterir ($1 \leq j < k$). (k register sayısıdır)*
- *T[i], i.tape alanının içeriğini gösterir. R_j, j.register içeriğini gösterir.*
- *$s \leq p$ programdaki instruction sıra numarasını gösterir.*
- *c doğal sayıdır.*
- *κ bir sonra çalıştırılacak instruction adresini saklar.*
- *Bütün instruction'lar aksi belirtilmediği sürece κ program sayacını bir artırır.*
- *Register 0 akümülator olarak kullanılır. Aritmetik ve lojik işlemlerde kullanılır.*
- *Makinenin çalışması bir halt instruction ile sonlandırılır.*

Random Access Turing Machines

Tanım:

RATM ***M = (k, Π)*** *şeklinde bir ikiliyle ifade edilir.*

- *k > 0 olmak üzere register sayısını gösterir.*
- *$\Pi = (\pi_1, \pi_2, ..., \pi_p,)$ olmak üzere sonlu sayıda instruction'a sahip programı gösterir. Burada her π_i bir instruction'ı gösterir.*
- *π_p son instruction'dır ve halt olduğu varsayılır.*
- *Program içinde başka halt instruction'ları olabilir.*

Random Access Turing Machines

Tanım:

RATM için ***konfigürasyon*** *k+2 tuple'dır ve*
$(\kappa, R_0, R_1, ..., R_{k-1}, T)$ şeklinde ifade edilir.

- *Burada $\kappa \in N$ program counter ve 0 ile p arasındadır.*
- *Halted konfigürasyonu için κ sıfırdır.*
- *R_j, j.register'in şimdiki değeridir. $(0 \leq j < k)$*
- *T, tape içeriğini gösterir, sonlu sayıda pozitif tamsayı çifti kümesidir ve $(N - \{0\} \times N - \{0\})$ $(i \geq 1)$ olmak üzere $(i, m) \in T$ şeklinde gösterilir. i.tape alanının içeriği m 'dir. $(m > 0)$*

Random Access Turing Machines

Tanım:

$M = (k, \Pi)$ bir RATM olsun. Bir konfigürasyon $C=(\kappa, R_0, R_1, ..., R_{k-1}, T)$ bir adım sonra $C'=(\kappa', R'_0, R_1, ..., R'_{k-1}, T')$ konfigürasyonuna geçiyorsa $C \vdash_M C'$ şeklinde gösterilir.

- *Eğer π_k instruction'ı **read j** şeklinde ise, $(j < k)$ R_0 register'ı tape biriminin R_j ile gösterilen alanının değerini alır. Tape alanı R_j register'ı tarafından adreslenir. Böylece $R'_0 = T(R_j)$ olur ve $T(R_j)$ unique bir değere sahiptir ve $(R_j, m) \in T$ dir. $\kappa' = \kappa + 1$ olur.*
- *Eğer π_k instruction'ı **add = c** şeklinde ise, $(c \geq 0)$ $R'_0 = R_0 + c$ ve $\kappa' = \kappa + 1$ olur.*
- *Eğer π_k instruction'ı **write j** şeklinde ise, $(j < k)$ $\kappa' = \kappa + 1$ olur. Tape biriminde R_j nin değeriyle belirtilen sıradaki alana R_0 yazılır.*
- *Eğer π_k instruction'ı **jpos s** şeklinde ise, $(1 \leq s \leq p)$ eğer $R_0 > 0$ ise $\kappa'=s$ olur. Eğer $R_0 \leq 0$ ise $\kappa' = \kappa + 1$ olur.*

$\vdash^*_M$ ilişkisi, $\vdash_M$ ilişkisinin reflexive, transitive closure'udur.

Random Access Turing Machines

Örnek (Çarpma):

*$M = (k, \Pi)$ bir RATM için iki sayının çarpımını yapan **mply** adında bir instruction ekleyelim.*

- *Register 0 x değerine ve Register 1 y değerine sahip olsun.*
- *RATM halt durumuna ulaşınca Register 0 içinde x.y değeri olacaktır.*
- *Çarpma işlemi ardarda toplama işlemleriyle yapılacaktır.*

Random Access Turing Machines

Örnek (Çarpma): (devam)

x y $x+x$ z w

$(1;5,3,0,0,0;\emptyset) \vdash (2;5,3,5,0,0;\emptyset) \vdash (3;3,3,5,0,0;\emptyset) \vdash (4;3,3,5,0,0;\emptyset) \vdash$
$(5;1,3,5,0,0;\emptyset) \vdash (6;1,3,5,1,0;\emptyset) \vdash (7;3,3,5,1,0;\emptyset) \vdash (8;2,3,5,1,0;\emptyset) \vdash$
$(9;1,3,5,1,0;\emptyset) \vdash (10;1,3,5,1,0;\emptyset) \vdash (11;0,3,5,1,0;\emptyset) \vdash$
$(12;5,3,5,1,0;\emptyset) \vdash (13;5,3,5,1,5;\emptyset) \vdash (14;5,3,5,1,5;\emptyset) \vdash$
$(15;10,3,5,1,5;\emptyset) \vdash (16;10,3,10,1,5;\emptyset) \vdash (17;1,3,10,1,5;\emptyset) \vdash$
$(18;1,1,10,1,5;\emptyset) \vdash (2;1,1,10,1,5;\emptyset) \vdash^{*} (18;0,0,20,0,15;\emptyset) \vdash$
$(2;0,0,20,0,15;\emptyset) \vdash (3;0,0,20,0,15;\emptyset) \vdash (19;0,0,20,0,15;\emptyset) \vdash$
$(20;15,0,20,0,15;\emptyset)\ ]$

- *Başlangıçta R0 = x ve R1 = y.*
- *Sonuçta R0 = x.y dir.*
- ***mply 1** işlemi gerçekleştirilir.*
- *Program her iterasyonda π_2-π_{18} arasındaki işlemleri yapar*

k.iterasyonda

- *Register 2 = $x2^k$*
- *Register 3 = $\lfloor y / 2^k \rfloor$*
- *Register 1 = $\lfloor y / 2^{k-1} \rfloor$*
- *Register 4 = $x .(y \bmod 2^k)$*

	1.	store 2
	2.	load 1
	3.	jzero 19
$z = y / 2$	4.	half
	5.	store 3
	6.	load 1
	7.	sub 3
	8.	sub 3
	9.	jzero 13
$y - z - z$	10.	load 4
	11.	add 2
$w = w + x$	12.	store 4
	13.	load 2
$x = x + x$	14.	add 2
	15.	store 2
	16.	load 3
$y = z$	17.	store 1
	18.	jump 2
	19.	load 4
	20.	halt

Random Access Turing Machines

Örnek: $R_1 := R_2 + R_1 - 1$

1. *load 1*
2. *add 2*
3. *sub = 1*
4. *store 1*

Örnek: while x > 0 do x:= x – 3 *(x = Register 1)*

1. *load 1*
2. *jzero 6*
3. *sub = 3*
4. *store 1*
5. *jump 1*

Random Access Turing Machines

Örnek (Çarpma): $w := x \cdot y$ *değerini hesaplar.*

```
w:= 0
while y > 0 do
        begin
                z:= half(y)
                if  y – z – z ≠ 0 then w:= w + x
                x:= x + x
                y:= z
        end
halt
```

$y \rightarrow R_1$, $x \rightarrow R_2$, $z \rightarrow R_3$ *ve* $w \rightarrow R_4$ *göstermektedir.*

Random Access Turing Machines

Tanım: $\mathbf{E}$, Σ *ve* $\{0, 1, \ldots, |\Sigma|-1\}$ *arasında bir bijection olsun* ($\mathbf{E}(\sqcup) = 0$). $w = a_1a_2\ldots a_n \in (\Sigma - \sqcup)^*$ *girişi için bir RATM* $M = (k, \Pi)$ *nin başlangıç konfigürasyonu* $(\kappa, R_0, \ldots, R_{k-1}, T)$ *olsun.*

- *Burada,* $\kappa = 1$, $R_j = 0$, *ve* $T = \{(1, \mathbf{E}(a_1)), (2, \mathbf{E}(a_2)), \ldots, (n, \mathbf{E}(a_n))\}$
- M, $w \in \Sigma^*$ *girişi için halt durumuna ulaştığında* $R_0=1$ *ise bu string'i kabul eder* $R_0=0$ *ise bu string'i red eder.*

Random Access Turing Machines

Örnek: $L = \{a^n b^n c^n : n \geq 0\}$ *dilini tanıyan RATM programını yazınız.*

acount:= bcount:= ccount:= 0, n:= 1

while T[n] = 1 do n:= n + 1, acount:= acount + 1

while T[n] = 2 do n:= n + 1, bcount:= bcount + 1

while T[n] = 3 do n:= n + 1, ccount:= ccount + 1

if acount = bcount = ccount and T[n] = 0 then accept

else reject

- *E(a) = 1, E(b) = 2, E(c) = 3*
- *accept için "load =1, halt" ve reject için "load =0, halt" yazılabilir.*

Nondeterministic Turing Machines

Bir nondeterministic Turing makinesi $(K, \Sigma, \Delta, s, H)$ *şeklinde quintuple olarak ifade edilir.*

- K, Σ, s, H *standart Turing makineleriyle aynıdır.*
- Δ *geçiş ilişkisi* $((K - H) \times \Sigma) \times ((K \times (\Sigma \cup \{\leftarrow, \rightarrow\})$ *kümesinin alt kümesidir.*
- $\vdash_M^*$ *ilişkisi,* $\vdash_M$ *ilişkisinin reflexive, transitive closure'udur.*
- *Bir konfigürasyon* $\vdash_M$ *için birden fazla konfigürasyona geçilebilir.*
- M, $w \in (\Sigma - \{\triangleright, \sqcup\})^*$ *girişini kabul eder eğer* $h \in H$, $a \in \Sigma$ *ve* $u, v \in \Sigma^*$ *için* $(s, \triangleright \underline{\sqcup} w) \vdash_M^* (h, \triangleright \sqcup \underline{u} \underline{a} v)$ *ise.*

Unrestricted Grammars

- *Context-free gramerler terminal (Σ) ve nonterminallerin (V-Σ) kümesi olan bir alfabeye sahiptir.*
- $A \rightarrow u$ *($A \in V$-Σ, ve $u \in V^*$).*
- *Bir context-free gramer başlangıç string S ile başlayıp sürekli sol taraftaki nonterminal yerine kuralların sağ tarafını değiştirmektedir.*
- *CFG'de bütün kuralların sol tarafları bir nonterminale sahiptir.*
- *Bir **unrestricted** gramerde ise kuralların sol tarafı en az bir tane nonterminale sahip olmak kaydıyla bir string olabilir.*
- *Sonuçta üretilen string CFG'deki gibi sadece terminallerden oluşabilir.*

Unrestricted Grammars

Tanım: Bir unrestricted grammer $G = (V, \Sigma, R, S)$ şeklinde quadruple ile gösterilir.

- V *bir alfabe*
- $\Sigma \subseteq V$ *terminaller kümesi*
- $S \in V - \Sigma$ *başlangıç sembolü*
- *R kurallar kümesi,* $R \subseteq (V^*(V - \Sigma)V^*) \times V^*$
- $u \rightarrow v$ *yazılabilir eğer* $(u, v) \in R$ *ise*
- $u \Rightarrow_G v$ *yazılabilir eğer* $w_1, w_2 \in V^*, u' \rightarrow v' \in R$ *için* $u = w_1 u' w_2$, $v = w_1 v' w_2$ *ise*
- $\Rightarrow^*_G$, $\Rightarrow_G$ *geçişlerinin reflexive, transitive, closure'udur.*
- *Bir* $w \in \Sigma$ *string'i G grameri tarafından üretilir eğer* $S \Rightarrow^*_G w$ *ise*

Unrestricted Grammars

Örnek: $G = (V, \Sigma, R, S)$ grameri $L = \{a^n b^n c^n : n \geq 1\}$ dilini üretir.

$V = \{S, a, b, c, A, B, C, D, T_a, T_b, T_c\}$ ve $\Sigma = \{a, b, c\}$

$R =$ {	$S \rightarrow ABCS,$	İlk iki kural $(ABC)^n T_c$ üretir
	$S \rightarrow T_c,$	
	$CA \rightarrow AC,$	Sonraki üç kural A,B,C'leri sıralar
	$BA \rightarrow AB,$	$A^n B^n C^n Tc$ oluşur
	$CB \rightarrow BC,$	
	$CT_c \rightarrow T_c c,$	Diğer kurallar C yerine c, B yerine b,
	$CT_c \rightarrow T_b c,$	A yerine a yazar
	$BT_b \rightarrow T_b b,$	
	$BT_b \rightarrow T_a b,$	
	$AT_a \rightarrow T_a a,$	
	$T_a \rightarrow e$ }	

Unrestricted Grammars

Örnek: (devam) $a^2 b^2 c^2$ *string'inin üretilişi aşağıdadır.*

$R =$ {	$S \rightarrow ABCS,$	$S \Rightarrow_G ABCS$
	$S \rightarrow T_c,$	$\Rightarrow_G ABCABCS \Rightarrow_G ABACBCS$
	$CA \rightarrow AC,$	$\Rightarrow_G AABCBCS \Rightarrow_G AABBCCS$
	$BA \rightarrow AB,$	$\Rightarrow_G AABBCCT_c \Rightarrow_G AABBCT_c c$
	$CB \rightarrow BC,$	$\Rightarrow_G AABBT_b cc \Rightarrow_G AABT_b bcc$
	$CT_c \rightarrow T_c c,$	$\Rightarrow_G AAT_a bbcc \Rightarrow_G AT_a abbcc$
	$CT_c \rightarrow T_b c,$	$\Rightarrow_G T_a aabbcc \Rightarrow_G aabbcc$
	$BT_b \rightarrow T_b b,$	
	$BT_b \rightarrow T_a \upsilon,$	
	$AT_a \rightarrow T_a a,$	
	$T_a \rightarrow e$ }	

Undecidability

Church-Turing thesis

- *Bu derste farklı hesaplama süreçleri için farklı matematiksel modeller verilmiştir.*
- *Özellikle bir dilin decide, semidecide edilmesi ve language generator ve hesaplama fonksiyonları için modeller geliştirilmiştir.*
- *Bir Turing makinesinin özelliklerinin artırılması (RATM) aslında hesaplama kabiliyetini arttırmamaktadır.*
- *Bir hesaplama makinesi bir algoritmanın gerçekleştirilmesi için tasarlanır.*

Undecidability

Church-Turing thesis (devam)

- *Bütün girişler için halt durumuna ulaşan Turing makineleri algoritmadır. Bu prensip **Church-Turing thesis** olarak adlandırılır.*
- *Semidecide yapan Turing makineleri algoritma değillerdir.*
- *Church-Turing tezinin doğruluğu matematiksel olarak ispatlanamamıştır, yanlışlığıda ispatlanamamıştır.*
- *Church-Turing tezine göre, Turing makinesiyle gerçekleştirilemeyen hesaplamalar **undecidable** olarak adlandırılır.*

Ödev

- Problemleri çözünüz 4.4.2 (sayfa 221)
- Problemleri çözünüz 4.6.1a (sayfa 232)