Otomata Teorisi

Fırat İsmailoğlu, PhD

Hafta 6:
Pumping Lemma
İçerikten Bağımsız Diller (1. Bölüm)

Hafta 6
Plan

1. Olmayana Ergi Yöntemi
2. Güvercin Yuvası Prensibi
3. Pumping Lemma
4. İçerikten Bağımsız Diller
5. Grammar'ın Formal Gösterimi

Olmayana Ergi Yöntemi (Proof by Contradiction)

Matematikteki en temel ispatlama yöntemlerinden biridir.

OEY'de doğruluğunu ispatlamak istediğimiz hipotezin tersinin (zıttının) doğru olduğunu kabul ederiz. Bu kabulun elimizdeki verilerle bir çelişki oluşturduğunu ispatlayarak kabulumuzun yanlış olduğunu; yani hipotezimizin tersinin yanlış olduğunu gösteririz. Bu halde hipotezimiz doğrudur.

(Tersi yanlışsa kendi doğrudur)

ör. h = asal sayılar sonsuzdur.

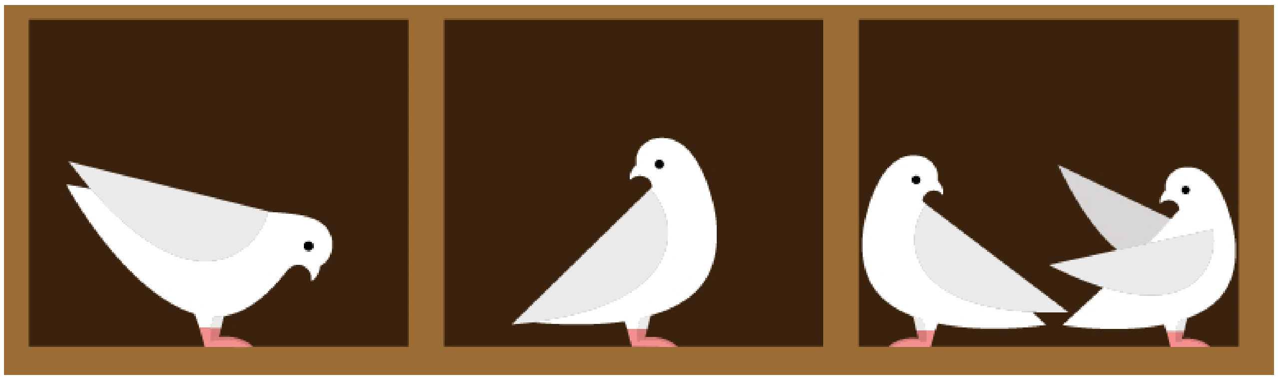
OEY. h'=asal sayılar sonludur. Su durumda p_{en} en son ve en büyük asal sayı olsun.

$k = 2 \times 3 \times \cdots \times p_{en} + 1$ şeklinde oluşturabileceğimiz bir sayı asal olur (kendinden ve 1 den başka tam sayı böleni yok).

k sayısı p_{en}'den büyüktür; su halde p_{en} en büyük asal sayı olamaz. Çeliski oluşur. h' yanlıştır; h doğrudur.

Güvercin Yuvası Prensibi (Pigeonhole Princible)

Eğer bir yerdeki güvercin sayısı, güvercin yuvası sayısından fazla ise, en az bir yuvada birden fazla güvercin vardır.



ör. Bir odadaki 13 kişiden en az iki kişi aynı ay doğmuştur.

13 güvercin (kişi), 12 güvercin yuvası (ay)

ör. İstanbul'da aynı sayıda saç teline sahip iki kişi vardır.

Bir insandaki toplam saç teli sayısı 0-500.000 arasındadir.

İstanbul'da milyonlarca kişi yaşar.

Milyonlarca kişiyi güvercin, saç teli sayısını güvercin yuvası sayısı olarak düşünürsek aynı sayıda saç teline sahip bir çok kişi bulunabilir.

Düzenli Olmayan Diller Örneği

ör. $L = \{0^n 1^n | n \geq 1\}$ dilini düşünelim. Bu dilin kelimeleri $01, 0011, 000111$ gibi belirli bir sayıdaki 0'dan sonra aynı sayıda 1 ile devam eden kelimelerdir. Bu dil düzenli değildir.

çözüm. OEY. Diyelim ki bu dil düzenli olsun, o halde bu dili tanıyan k tane duruma sahip bir DSO vardır.

Kelimeleri eşit sayıda 0 ve 1 den oluşan bu dilin kelimlerini kabul eden DSO'nun 1 harflerini okumaya başladığında daha önce kaç defa 0 okumuş olduğunu *hatırlaması* gerekir. Fakat DSO'ların hatırlama özelliği çok kısıtlıdır, hatta yoktur.

Bunun için DSO'nun her bir durumu o ana kadar kaç tane 0 harfi okundugunu temsil etsin (q_0 0 tane 0 okumayı, q_1 I tane 0 okumayı,...).

$w = 0^k 1^k$ kelimesini test edelim.

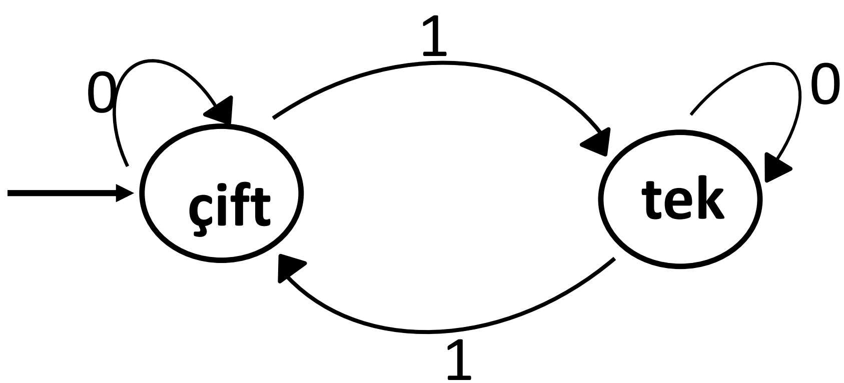
Bu DSO'da k tane 0 harfini okumak için, $k + 1$ tane durum ziyaret edilir $(\varepsilon, 0,00, \dots, 0^k)$. Bu durumda güvercin yuvası prensibi gereği en az bir durum birden fazla kez ziyaret edilmiştir.

Düzenli Olmayan Diller Örneği

Bu durumda 0'ları okuduktan sonra vardığımız durumun o ana kadar okunan 0 sayısını temsil ettiğinden emin olamayız; yani kaç 0 okuduğumuzu bilemeyiz.

Şu durumda böyle bir DSO tasarlayamayız, bu ise tasarlayabileceğimiz hipoteziyle çelişir. OEY gereği verilen L dili düzenli değildir.

1. Hafta notlarında, verilen kelimedeki toplam 1 sayısının tek mi çift mi olduğuna karar veren aşağıdaki otomatayı gostermistik. Bu otomada verilen kelimenin tamamıni hafızasinda tutmuyor, her defasında kelimenin yalnızca bir tane harfini okuyabiliyordu.



Pumping Lemma

Pumping lemma, düzenli diller için geçerlidir. Yani bir dil düzenli ise Pumping lemmayı sağlar.

Fakat Pumping lemma, bir dilin düzenli olduğunu değil düzenli olmadiğını göstermek için kullanılır.

Kısaca, bir dilin düzenli olmadığını göstermek için, OEY ile önce düzenli bir dil olduğu varsayılır. Ardından Pumping lemmayı sağlayamadığı gösterilerek bir çelişki ortaya konur. Böylece dilin düzenli olmadığı ispatlanır.

Pumping Lemma

L bir düzenli dil ise, L'nin bir p pumping uzunluğu vardır öyleki L'nin en az p uzunluğundaki her bir kelimesi ($\forall\, w \in L, |w| \geq p$), aşağıda verilen üç şartı sağlayarak $w = xyz$ şeklinde üç parçaya ayrılabilir:

1. $|y| > 0$,
2. $|xy| \leq p$,
3. her $i \geq 0$, için xy^{i}z de L'nin bir elemanıdır.

3. şart şöyle düşünülebilir: y'yi kaç defa tekrar edersek edelim (ne kadar şişirirsek şişirelim) $xy \ldots y\; z$, yine L'nin bir elamanıdır.

Pumping Lemma Bir Dilin Düzenli Olmdığını Göstermek

Pumping Lemma ile bir dilin düzenli olmadığı gösterilirken aşağıdaki aşamalar izlenir:

1. Dilin düzenli olduğu var sayılır.
2. Bir p pumping uzunluğuna sahip olmak zorundadır.
3. Bu dile ait p'den daha uzun bir w kelimesi bulunur.
4. w kelimesi $|y| > 0$ ve $|xy| \leq p$ olacak şekilde $w = xyz$ şeklinde üç parçaya bölünür.
5. Bazı i'ler için $xy^i z$ nin bu dilin elemanı olmadığı gösterilir.
6. Böylece Pumping lemmanın sağlanmadığı gösterilmiş olur, bu ise dilin düzenli olduğu varsayımı ile çelişir.

Pumping Lemma Örneği I

ör. $L = \{0^n 1^n | n \geq 1\}$ dilinin düzenli olmadığını gösterelim.

Bu dil düzenli olsun ve p Pumping uzunluğuna sahip olsun.

$w = 0^p 1^p \in L$ kelimesini ele alalım. Bu kelime şu şekildedir:

$$w = \underbrace{0 \ldots 00}_{p \text{ tane}}\underbrace{1 \ldots 1}_{p \text{ tane}}$$

$$w = \underbrace{\underbrace{0 \ldots 0}_{p-1}}_{x}\underbrace{\underbrace{0}_{1}}_{y}\underbrace{\underbrace{1 \ldots 1}_{p}}_{z}$$

$i = 2$ için $y^2 = 00$ olur. Bu durumda xy^2z

$$\underbrace{0 \ldots 0}_{x}\,\underbrace{00}_{y^2}\,\underbrace{1 \ldots 1}_{z}$$

olup $p + 1$ tane 0; p tane 1 den oluştuğundan L'nin bir elemanı olmaz.

Pumping Lemma Örneği 2

ör. $L = \{vv | v \in \{0,1\}^*\}$ dili düzenli değildir.

Bu dil düzenli olsun ve p Pumping uzunluğuna sahip olsun.

$w = 0^p10^p1$ kelimesini ele alalım, açıkça bu kelime p'den uzundur ve şu şekildedir:

$$w = \underbrace{0 \ldots 0}_{p \text{ tane}}1\underbrace{0 \ldots 0}_{p \text{ tane}}1.$$

Bu kelimeyi $x = 0^{p-1}, y = 0, z = 10^p1$ olacak şekilde üç parçaya ayıralım. Dikkat edin $|xy| = |0^p| \leq p$ ve $|y| = 1 > 0$ şartları sağlanıyor.

i yine 2 olsun (yada herhangi bir pozitif sayı). Şu halde xy^2z

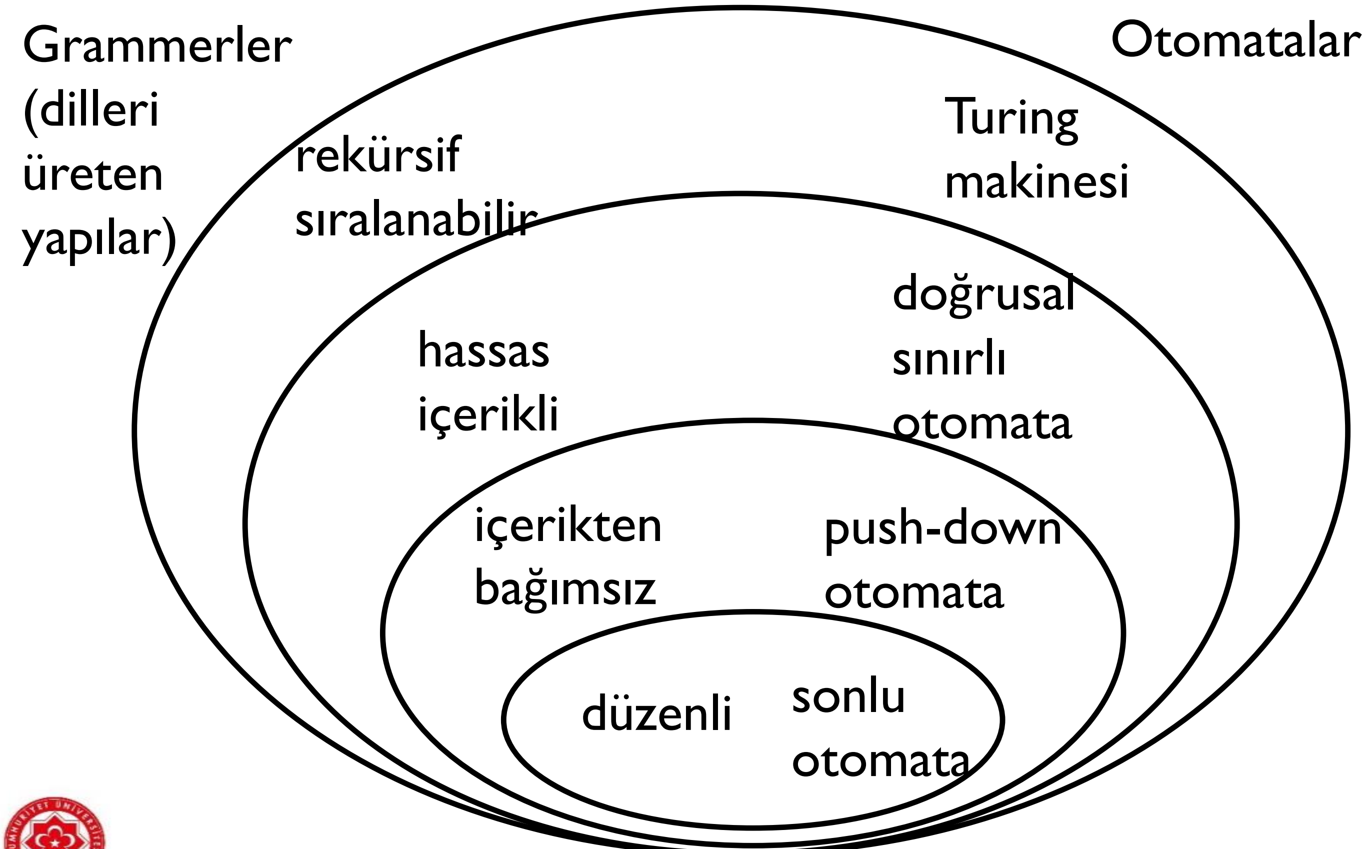
$$\underbrace{0 \ldots 0}_{x}\underbrace{00}_{y^2}\underbrace{10 \ldots 01}_{z}$$

iki eşit parçadan oluşmadığı için L'nin bir elemanı değildir. Bu halde Pumping lemma sağlanamadığı için L düzenli değildir.

İçerikten Bağımsiz Diller (Context-Free Languages)

$L = \{0^n1^n | n \geq 1\}$ dilinin düzenli olmadığını gördük; bu yüzden düzenli bir ifade ile yada bir sonlu otomata tarafından tanınamaz.

Bu tür dillerin tanınması için daha geniş bir kavram olan içerikten bağımsız grammer (context-free grammar) kullanacağız.



Chomsky Hiyerarşisi

İçerikten Bağımsiz Diller Örneği

Palindromlar dilini düşünelim. Palindrom sondan yada baştan okunduğunda (soldan saga yada sağdan sola) okunduğunda aynı olan kelimelere denir, örneğin küçük, radar..

Pumping lemma kullanılarak görülebileceği gibi bu dil düzensizdir.

$\Sigma = \{0,1\}$ alfabesi kullanılarak oluşturulan palindromlarin (örneğin $010, 110011, 11111..$) dili asağıdaki şekilde rekürsif (yinelemeli) olarak üretilebilir:

1. Temel: $\varepsilon, 0$ ve 1 palindromdur.
2. Tümervarım: Her w palindromu için $0w0$ ve $1w1$ de palindrom olur.

$L_{pal} = \{\varepsilon, 0,1,0\varepsilon 0,1\varepsilon 1,000,101,010,111,0000,1001,0110,1111,$
$00000,10001, 01010,11011,00100,10101,01110,11111, \ldots\}$

İçerikten Bağımsiz Diller Örneği

Bu palindromlar içerikten bağımsız grammer kullanılarak bir dizi kurallarla da oluşturulabilir:

$$\left.\begin{array}{ll} \mathit{1.} & P \longrightarrow \varepsilon \\ \mathit{2.} & P \longrightarrow 0 \\ \mathit{3.} & P \longrightarrow 1 \end{array}\right\}\ \text{Temel}$$

$$\left.\begin{array}{ll} \mathit{4.} & P \longrightarrow 0P0 \\ \mathit{5.} & P \longrightarrow 1P1 \end{array}\right\}\ \text{Tümevarım}$$

İlk uç kural temel'i oluşturur. Bu kurallarda, okların sağ tarafında bir değişken yoktur.

Sonraki iki kural tümevarıma denk gelir. Örneğin 4. kural alacağımız her bir w palindromu için $0w0$ da bir palindrom olacağını ortaya koyar.

Grammerin Formal Gösterimi

Bir grammer 4-li sıradır ve $\mathrm{G} = (V, \Sigma, R, S)$ ile gösterilir. Burada:

1. V değişkenler kümesidir. (Non-terminaller de denir).

2. Σ terminaller, yani değişmezler kümesidir (alfabenin elemanlari tarafından oluşturulur).

3. R türetim kuralları kümesidir. Değişkenlerden nasıl kelimeler türetileceğini orataya koyan kurallardır. R'deki kurallar

$$A \longrightarrow w$$

formundadır. Burada $A \in V, w \in (V \cup \Sigma)^*$.

ör. $P \longrightarrow 0$

P değişkeni 0'ı bir veya birden fazla adımda türetebilir.

4. S başlangıç değişkenidir. Her zaman 1. kuralda okun sol tarafında yer alir.

Not. V ve Σ ayrık kümelerdir: $V \cap \Sigma = \emptyset$.

Grammerin Formal Gösterimi

Bu şekilde gösterilen G içerikten bağımsız grammeri ile oluşturulan dile içerikten bağımsız dil denir. Formal olarak şu şekilde gösterilir:

$$L(G) = \{w \in \Sigma^* | \, S \xrightarrow{*} w\}.$$

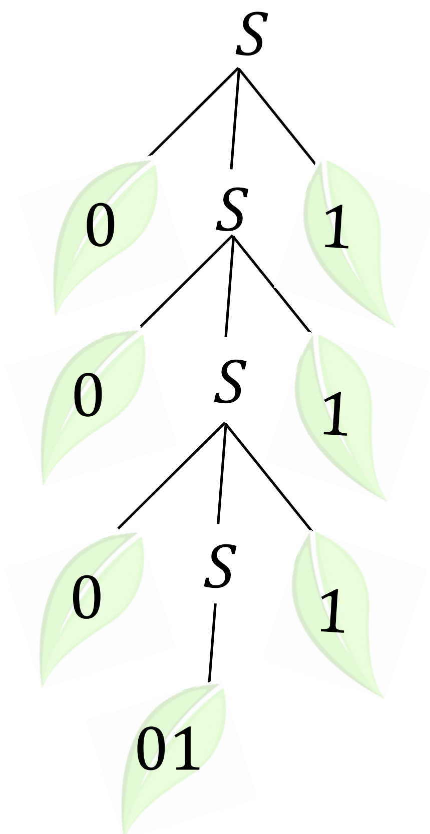
Not 1. Burada $w \in \Sigma^*$ olduğundan, üretilen kelimeler terminallerden olusur, icinde değişken olmaz!

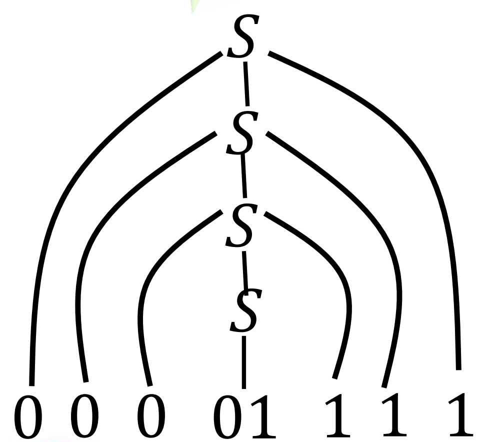
Not 2. $S \xrightarrow{*} w$, w kelimesine baslangıç değişkeninden sıfır, bir yada bir kaç türetim adımı ile ulaşılabileceğini gösterir.

or. $\mathrm{G} = (V, \Sigma, R, S)$ grammeri için

$V = \{S\}, \Sigma = \{0,1\}, S$ başlangıç değişkeni olsun ve R şöyle olsun:

$$S \longrightarrow 0S1$$
$$S \longrightarrow 01$$





Bu ağacın yapraklarından oluşan 00001111 kelimesi bu grammer tarafından üretilmiş bir kelimedir. Bu ağaca türetim ağacı (parse tree) denir. (Türetim ağacı asağıdaki şekilde de gösterilebilir.)

ör. $\mathrm{G} = (\{S\}, \{a, b\}, R, S)$ ve R kuralları:

$$S \longrightarrow aSb$$
$$S \longrightarrow SS$$
$$S \longrightarrow \varepsilon$$

gibi olsun. Not burada kurallar

$$S \longrightarrow aSb|SS|\varepsilon$$

şeklinde daha kısa olarak da verilebilir.

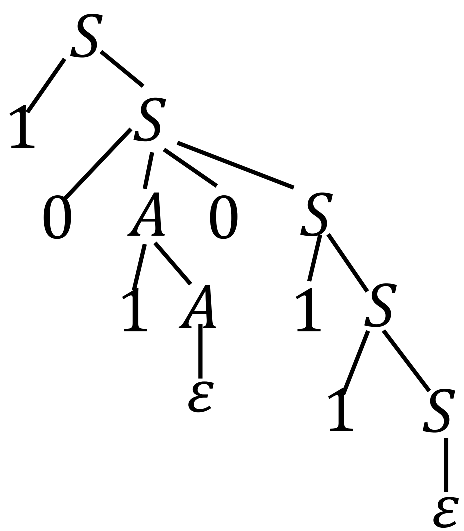
Bu grammerin ürettigi kelimeler $abab, aaabbb, aababab$ gibi kelimelerdir.

Burada a harfini sol paranetez ('('), b harfini sağ parantez (')'), olarak düşünürsek, bu grammerin ürettigi dilin aslında düzgün olarak ic ice geçmiş parentezler dili olduğunu görürüz.

ör. $\mathrm{G} = (\{S\}, \{0, 1\}, R, S)$ ve R kuralları

$$S \longrightarrow 1S \mid 0A0S \mid \varepsilon$$
$$A \longrightarrow 1A|\varepsilon$$

grammeri çift sayida 0 iceren kelimerlerin dilini üretir. Örneğin 101011 kelimesi şu sekilde üretilebilir.



Not: Dikkat edin, burada yalnızca değişkenler (A, S) den türeme yapılır: terminaller $(0,1)$ türemez. Ve sonuç kelimesi yalnızca terminalleri içerir.

Neden 'İçerikten Bağımsız' Diyoruz?

İçerikten bağimsiz dillerde türetim kurallarının sol tarafında yalnızca bir tane değişken bulunur. Örneğin:

$$A \longrightarrow b$$

yani hiç bir şart aramaksızın A değişkenin olduğu yeri b ile değistirebiliriz. Sonuçta bu durumda kelimenin içeriğinden bağımsızdir.

Fakat şöyle kurallarımız olsaydı:

$$Ac \longrightarrow b$$
$$BA \longrightarrow c$$

A değişkenini değiştirmek için, A'nin sağına ve soluna bakmamız gerekirde değiştirmeden önce. Yani örneğin ancak A'dan sonra c terminali gelmek koşuluyla A yerine b yazabiliriz. Şu halde kelimenin içeriğine bağımlıyız.

En Sol ve En Sağ Türetim (Leftmost and Rightmost Derivations)

Eğer bir kelime türetiminde, her bir adımda kelimenin içindeki değişkenlerden en solda olan değişken değiştiriliyorsa bu türetime en sol türetim, en sağda bulunan değişken değiştiriliyorsa bu türetime en sağ türetim denir.

ör. $\mathrm{G} = (\{A, B, S\}, \{0, 1\}, R, S)$ ve R kuralları

$$1.\ S \longrightarrow AB$$
$$2.\ A \longrightarrow aaA$$
$$3.\ A \longrightarrow \varepsilon$$
$$4.\ B \longrightarrow Bb$$
$$5.\ B \longrightarrow \varepsilon$$

yukarıdaki şekilde numaralandırılmış olarak verilsin. Bu grammerden aab kelimesi en sol ve en sağ türetimle iki şekilde türetilebilir:

$$S \xrightarrow{1} AB \xrightarrow{2} aaAB \xrightarrow{3} aaB \xrightarrow{4} aaBb \xrightarrow{5} aab$$
$$S \xrightarrow{1} AB \xrightarrow{4} ABb \xrightarrow{2} aaABb \xrightarrow{5} aaAb \xrightarrow{3} aab$$