CS 373: Theory of Computation

Gul Agha Mahesh Viswanathan

University of Illinois, Urbana-Champaign

Fall 2010

Part I

Closure Properties of Turing Machines

Boolean Operators

Boolean Operators

Proposition

Decidable languages are closed under union, intersection, and complementation.

Boolean Operators

Proposition

Decidable languages are closed under union, intersection, and complementation.

Proof.

Given TMs M_1, M_2 that decide languages L_1, and L_2

- A TM that decides $L_1 \cup L_2$: on input x, run M_1 and M_2 on x, and accept iff either accepts.

Boolean Operators

Proposition

Decidable languages are closed under union, intersection, and complementation.

Proof.

Given TMs M_1, M_2 that decide languages L_1, and L_2

- A TM that decides $L_1 \cup L_2$: on input x, run M_1 and M_2 on x, and accept iff either accepts. (Similarly for intersection.)

Boolean Operators

Proposition

Decidable languages are closed under union, intersection, and complementation.

Proof.

Given TMs M_1, M_2 that decide languages L_1, and L_2

- A TM that decides $L_1 \cup L_2$: on input x, run M_1 and M_2 on x, and accept iff either accepts. (Similarly for intersection.)
- A TM that decides $\overline{L_1}$: On input x, run M_1 on x, and accept if M_1 rejects, and reject if M_1 accepts. $\square$

Regular Operators

Proposition

Decidable languages are closed under concatenation and Kleene Closure.

Proof.

Given TMs M_1 and M_2 that decide languages L_1 and L_2.

- A TM to decide $L_1 L_2$:

Regular Operators

Proposition

Decidable languages are closed under concatenation and Kleene Closure.

Proof.

Given TMs M_1 and M_2 that decide languages L_1 and L_2.

- A TM to decide L_1L_2: On input x, for each of the $|x| + 1$ ways to divide x as yz: run M_1 on y and M_2 on z, and accept if both accept. Else reject.

Regular Operators

Proposition

Decidable languages are closed under concatenation and Kleene Closure.

Proof.

Given TMs M_1 and M_2 that decide languages L_1 and L_2.

- A TM to decide L_1L_2: On input x, for each of the $|x|+1$ ways to divide x as yz: run M_1 on y and M_2 on z, and accept if both accept. Else reject.
- A TM to decide L_1^*:

Regular Operators

Proposition

Decidable languages are closed under concatenation and Kleene Closure.

Proof.

Given TMs M_1 and M_2 that decide languages L_1 and L_2.

- A TM to decide $L_1 L_2$: On input x, for each of the $|x|+1$ ways to divide x as yz: run M_1 on y and M_2 on z, and accept if both accept. Else reject.
- A TM to decide L_1^*: On input x, if $x = \epsilon$ accept. Else, for each of the $2^{|x|-1}$ ways to divide x as $w_1 \ldots w_k$ $(w_i \neq \epsilon)$: run M_1 on each w_i and accept if M_1 accepts all. Else reject. $\square$

Inverse Homomorphisms

Proposition

Decidable languages are closed under inverse homomorphisms.

Inverse Homomorphisms

Proposition

Decidable languages are closed under inverse homomorphisms.

Proof.

Given TM M_1 that decides L_1, a TM to decide $h^{-1}(L_1)$ is:

Inverse Homomorphisms

Proposition

Decidable languages are closed under inverse homomorphisms.

Proof.

Given TM M_1 that decides L_1, a TM to decide $h^{-1}(L_1)$ is: On input x, compute $h(x)$ and run M_1 on $h(x)$; accept iff M_1 accepts. □

Homomorphisms

Proposition

Decidable languages are not closed under homomorphism

Homomorphisms

Proposition

Decidable languages are not closed under homomorphism

Proof.

We will show a decidable language L and a homomorphism h such that $h(L)$ is undecidable

Homomorphisms

Proposition

Decidable languages are not closed under homomorphism

Proof.

We will show a decidable language L and a homomorphism h such that $h(L)$ is undecidable

- Let $L = \{xy \mid x \in \{0,1\}^*, y \in \{a,b\}^*, x = \langle M, w \rangle$, and y encodes an integer n such that the TM M on input w will halt in n steps $\}$

Homomorphisms

Proposition

Decidable languages are not closed under homomorphism

Proof.

We will show a decidable language L and a homomorphism h such that $h(L)$ is undecidable

- Let $L = \{xy \mid x \in \{0,1\}^*, y \in \{a,b\}^*, x = \langle M, w \rangle$, and y encodes an integer n such that the TM M on input w will halt in n steps $\}$
- L is decidable: can simply simulate M on input w for n steps

Homomorphisms

Proposition

Decidable languages are not closed under homomorphism

Proof.

We will show a decidable language L and a homomorphism h such that $h(L)$ is undecidable

- Let $L = \{xy \mid x \in \{0,1\}^*, y \in \{a,b\}^*, x = \langle M, w \rangle$, and y encodes an integer n such that the TM M on input w will halt in n steps $\}$
- L is decidable: can simply simulate M on input w for n steps
- Consider homomorphism h: $h(0) = 0$, $h(1) = 1$, $h(a) = h(b) = \epsilon$.

Homomorphisms

Proposition

Decidable languages are not closed under homomorphism

Proof.

We will show a decidable language L and a homomorphism h such that $h(L)$ is undecidable

- Let $L = \{xy \mid x \in \{0,1\}^*, y \in \{a,b\}^*, x = \langle M, w \rangle$, and y encodes an integer n such that the TM M on input w will halt in n steps $\}$
- L is decidable: can simply simulate M on input w for n steps
- Consider homomorphism h: $h(0) = 0$, $h(1) = 1$, $h(a) = h(b) = \epsilon$.
- $h(L) =$

Homomorphisms

Proposition

Decidable languages are not closed under homomorphism

Proof.

We will show a decidable language L and a homomorphism h such that $h(L)$ is undecidable

- Let $L = \{xy \mid x \in \{0,1\}^*, y \in \{a,b\}^*, x = \langle M, w \rangle$, and y encodes an integer n such that the TM M on input w will halt in n steps $\}$
- L is decidable: can simply simulate M on input w for n steps
- Consider homomorphism h: $h(0) = 0$, $h(1) = 1$, $h(a) = h(b) = \epsilon$.
- $h(L) = \text{HALT}$ which is undecidable. $\square$

Boolean Operators

Proposition

R.E. languages are closed under union, and intersection.

Boolean Operators

Proposition

R.E. languages are closed under union, and intersection.

Proof.

Given TMs M_1, M_2 that recognize languages L_1, L_2

Boolean Operators

Proposition

R.E. languages are closed under union, and intersection.

Proof.

Given TMs M_1, M_2 that recognize languages L_1, L_2

- A TM that recognizes $L_1 \cup L_2$: on input x, run M_1 and M_2 on x in parallel, and accept iff either accepts.

Boolean Operators

Proposition

R.E. languages are closed under union, and intersection.

Proof.

Given TMs M_1, M_2 that recognize languages L_1, L_2

- A TM that recognizes $L_1 \cup L_2$: on input x, run M_1 and M_2 on x in parallel, and accept iff either accepts. (Similarly for intersection; but no need for parallel simulation) $\square$

Complementation

Proposition

R.E. languages are not closed under complementation.

Proof.

A_{TM} is r.e. but $\overline{A_{\text{TM}}}$ is not. $\square$

Regular Operations

Proposition

R.E languages are closed under concatenation and Kleene closure.

Proof.

Given TMs M_1 and M_2 recognizing L_1 and L_2

- A TM to recognize $L_1 L_2$:

Regular Operations

Proposition

R.E languages are closed under concatenation and Kleene closure.

Proof.

Given TMs M_1 and M_2 recognizing L_1 and L_2

- A TM to recognize L_1L_2: On input x, do in parallel, for each of the $|x|+1$ ways to divide x as yz: run M_1 on y and M_2 on z, and accept if both accept. Else reject.

Regular Operations

Proposition

R.E languages are closed under concatenation and Kleene closure.

Proof.

Given TMs M_1 and M_2 recognizing L_1 and L_2

- A TM to recognize L_1L_2: On input x, do in parallel, for each of the $|x|+1$ ways to divide x as yz: run M_1 on y and M_2 on z, and accept if both accept. Else reject.
- A TM to recognize L_1^*:

Regular Operations

Proposition

R.E languages are closed under concatenation and Kleene closure.

Proof.

Given TMs M_1 and M_2 recognizing L_1 and L_2

- A TM to recognize $L_1 L_2$: On input x, do in parallel, for each of the $|x|+1$ ways to divide x as yz: run M_1 on y and M_2 on z, and accept if both accept. Else reject.
- A TM to recognize L_1^*: On input x, if $x = \epsilon$ accept. Else, do in parallel, for each of the $2^{|x|-1}$ ways to divide x as $w_1 \ldots w_k$ ($w_i \neq \epsilon$): run M_1 on each w_i and accept if M_1 accepts all. Else reject. $\square$

Homomorphisms

Proposition

R.E. languages are closed under both inverse homomorphisms and homomorphisms.

Proof.

Let TM M_1 recognize L_1.

- A TM to recognize $h^{-1}(L_1)$:

Homomorphisms

Proposition

R.E. languages are closed under both inverse homomorphisms and homomorphisms.

Proof.

Let TM M_1 recognize L_1.

- A TM to recognize $h^{-1}(L_1)$:On input x, compute $h(x)$ and run M_1 on $h(x)$; accept iff M_1 accepts.

Homomorphisms

Proposition

R.E. languages are closed under both inverse homomorphisms and homomorphisms.

Proof.

Let TM M_1 recognize L_1.

- A TM to recognize $h^{-1}(L_1)$:On input x, compute $h(x)$ and run M_1 on $h(x)$; accept iff M_1 accepts.
- A TM to recognize $h(L_1)$:

Homomorphisms

Proposition

R.E. languages are closed under both inverse homomorphisms and homomorphisms.

Proof.

Let TM M_1 recognize L_1.

- A TM to recognize $h^{-1}(L_1)$:On input x, compute $h(x)$ and run M_1 on $h(x)$; accept iff M_1 accepts.
- A TM to recognize $h(L_1)$: On input x, start going through all strings w, and if $h(w) = x$, start executing M_1 on w

Homomorphisms

Proposition

R.E. languages are closed under both inverse homomorphisms and homomorphisms.

Proof.

Let TM M_1 recognize L_1.

- A TM to recognize $h^{-1}(L_1)$:On input x, compute $h(x)$ and run M_1 on $h(x)$; accept iff M_1 accepts.
- A TM to recognize $h(L_1)$: On input x, start going through all strings w, and if $h(w) = x$, start executing M_1 on w, using dovetailing to interleave with other executions of M_1.

Homomorphisms

Proposition

R.E. languages are closed under both inverse homomorphisms and homomorphisms.

Proof.

Let TM M_1 recognize L_1.

- A TM to recognize $h^{-1}(L_1)$:On input x, compute $h(x)$ and run M_1 on $h(x)$; accept iff M_1 accepts.
- A TM to recognize $h(L_1)$: On input x, start going through all strings w, and if $h(w) = x$, start executing M_1 on w, using dovetailing to interleave with other executions of M_1. Accept if any of the executions accepts. $\square$

Part II

Grammars

Parenthesis Matching

Problem: Describe the the set of arithmetic expressions with correctly matched parenthesis.

Parenthesis Matching

Problem: Describe the the set of arithmetic expressions with correctly matched parenthesis.

Solution: Ignoring numbers and variables, and focussing only on parenthesis, correctly matched expressions can be defined as

Parenthesis Matching

Problem: Describe the the set of arithmetic expressions with correctly matched parenthesis.
Solution: Ignoring numbers and variables, and focussing only on parenthesis, correctly matched expressions can be defined as

- The ϵ is a valid expression

Parenthesis Matching

Problem: Describe the the set of arithmetic expressions with correctly matched parenthesis.
Solution: Ignoring numbers and variables, and focussing only on parenthesis, correctly matched expressions can be defined as

- The ϵ is a valid expression
- A valid string ($\neq \epsilon$) must either be

Parenthesis Matching

Problem: Describe the the set of arithmetic expressions with correctly matched parenthesis.
Solution: Ignoring numbers and variables, and focussing only on parenthesis, correctly matched expressions can be defined as

- The ϵ is a valid expression
- A valid string ($\neq \epsilon$) must either be
    - The concatenation of two correctly matched expressions, or

Parenthesis Matching

Problem: Describe the the set of arithmetic expressions with correctly matched parenthesis.
Solution: Ignoring numbers and variables, and focussing only on parenthesis, correctly matched expressions can be defined as

- The ϵ is a valid expression
- A valid string ($\neq \epsilon$) must either be
    - The concatenation of two correctly matched expressions, or
    - It must begin with (and end with) and moreover, once the first and last symbols are removed, the resulting string must correspond to a valid expression.

Parenthesis Matching

Grammar

Taking E to be the set of correct expressions, the inductive definition can be succinctly written as

$$\begin{aligned} E &\rightarrow \epsilon \\ E &\rightarrow EE \\ E &\rightarrow (E) \end{aligned}$$

English Sentences

English sentences can be described as

$$
\begin{aligned}
\langle S\rangle &\rightarrow \langle NP\rangle\langle VP\rangle\\
\langle NP\rangle &\rightarrow \langle CN\rangle \mid \langle CN\rangle\langle PP\rangle\\
\langle VP\rangle &\rightarrow \langle CV\rangle \mid \langle CV\rangle\langle PP\rangle\\
\langle PP\rangle &\rightarrow \langle P\rangle\langle CN\rangle\\
\langle CN\rangle &\rightarrow \langle A\rangle\langle N\rangle\\
\langle CV\rangle &\rightarrow \langle V\rangle \mid \langle V\rangle\langle NP\rangle\\
\langle A\rangle &\rightarrow \text{a} \mid \text{the}\\
\langle N\rangle &\rightarrow \text{boy} \mid \text{girl} \mid \text{flower}\\
\langle V\rangle &\rightarrow \text{touches} \mid \text{likes} \mid \text{sees}\\
\langle P\rangle &\rightarrow \text{with}
\end{aligned}
$$

English Sentences

Examples

$$\overbrace{\underbrace{\text{a}}_{\text{article}}\ \underbrace{\text{boy}}_{\text{noun}}}^{\text{noun-phrs}}\ \overbrace{\underbrace{\text{sees}}_{\text{verb}}}^{\text{verb-phrs}}$$

English Sentences

Examples

$$\overbrace{\underbrace{\text{a}}_{\text{article}}\ \underbrace{\text{boy}}_{\text{noun}}}^{\text{noun-phrs}}\ \overbrace{\underbrace{\text{sees}}_{\text{verb}}}^{\text{verb-phrs}}$$

$$\overbrace{\underbrace{\text{the}}_{\text{article}}\ \underbrace{\text{boy}}_{\text{noun}}}^{\text{noun-phrs}}\ \overbrace{\underbrace{\text{sees}}_{\text{verb}}\ \underbrace{\text{a flower}}_{\text{noun-phrs}}}^{\text{verb-phrs}}$$

Applications

Such rules (or grammars) play a key role in

- Parsing programming languages and natural languages
- Markup Languages like HTML and XML.
- Modelling software

Grammars

Definition

A grammar is $G = (V, \Sigma, R, S)$ where

Grammars

Definition

A grammar is $G = (V, \Sigma, R, S)$ where

- V is a finite set of variables also called nonterminals or syntactic categories. Each variable represents a language.

Grammars

Definition

A grammar is $G = (V, \Sigma, R, S)$ where

- V is a finite set of variables also called nonterminals or syntactic categories. Each variable represents a language.
- Σ is a finite set of symbols, disjoint from V, called terminals, that form the strings of the language.

Grammars

Definition

A grammar is $G = (V, \Sigma, R, S)$ where

- V is a finite set of variables also called nonterminals or syntactic categories. Each variable represents a language.
- Σ is a finite set of symbols, disjoint from V, called terminals, that form the strings of the language.
- R is a finite set of rules or productions. Each production is of the form $\alpha \rightarrow \beta$ where $\alpha, \beta \in (V \cup \Sigma)^*$

Grammars

Definition

A grammar is $G = (V, \Sigma, R, S)$ where

- V is a finite set of variables also called nonterminals or syntactic categories. Each variable represents a language.
- Σ is a finite set of symbols, disjoint from V, called terminals, that form the strings of the language.
- R is a finite set of rules or productions. Each production is of the form $\alpha \rightarrow \beta$ where $\alpha, \beta \in (V \cup \Sigma)^*$
- $S \in V$ is the start symbol; it is the variable that represents the language being defined. Other variables represent auxiliary languages that are used to define the language of the start symbol.

Example of a CFG

Example

Let $G_{\text{par}} = (V, \Sigma, R, S)$ be

- $V = \{E\}$
- $\Sigma = \{(,)\}$
- $R = \{E \to \epsilon, E \to EE, E \to (E)\}$
- $S = E$

Palindromes

Example

A string w is a palindrome if $w = w^R$.

Palindromes

Example

A string w is a palindrome if $w = w^R$. For example, madaminedenimadam(“Madam, in Eden, I’m Adam”)

Palindromes

Example

A string w is a palindrome if $w = w^R$. For example, madaminedenimadam("Madam, in Eden, I'm Adam")
$G_{\mathrm{pal}} = (\{S\}, \{0,1\}, R, S)$ defines palindromes over $\{0,1\}$, where R is

$$S \to \epsilon$$
$$S \to 0$$
$$S \to 1$$
$$S \to 0S0$$
$$S \to 1S1$$

Palindromes

Example

A string w is a palindrome if $w = w^R$. For example, madaminedenimadam("Madam, in Eden, I'm Adam")

$G_{\mathrm{pal}} = (\{S\}, \{0,1\}, R, S)$ defines palindromes over $\{0,1\}$, where R is

$$
\begin{aligned}
S &\rightarrow \epsilon \\
S &\rightarrow 0 \\
S &\rightarrow 1 \\
S &\rightarrow 0S0 \\
S &\rightarrow 1S1
\end{aligned}
$$

Or more briefly, $R = \{S \rightarrow \epsilon \mid 0 \mid 1 \mid 0S0 \mid 1S1\}$

Arithmetic Expressions

Consider the language of all arithmetic expressions (E) built out of integers (N) and identifiers (I), using only $+$ and $*$

Arithmetic Expressions

Consider the language of all arithmetic expressions (E) built out of integers (N) and identifiers (I), using only $+$ and $*$
$G_{\mathrm{exp}} = (\{E, I, N\}, \{a, b, 0, 1, (,), +, *, -\}, R, E)$ where R is

$$\begin{aligned}
E &\to I \mid N \mid E + E \mid E * E \mid (E) \\
I &\to a \mid b \mid Ia \mid Ib \\
N &\to 0 \mid 1 \mid N0 \mid N1 \mid -N \mid +N
\end{aligned}$$

More Examples

Example

Consider the grammar G with $\Sigma = \{a, b, c\}$, $V = \{S, B, C, H\}$ and

$$
\begin{array}{lll}
S \rightarrow aSBC \mid aBC & CB \rightarrow HB & HB \rightarrow HC \\
HC \rightarrow BC & aB \rightarrow ab & bB \rightarrow bb \\
bC \rightarrow bc & cC \rightarrow cc &
\end{array}
$$

More Examples

Example

Consider the grammar G with $\Sigma = \{a, b, c\}$, $V = \{S, B, C, H\}$ and

$$\begin{array}{lll} S \rightarrow aSBC \mid aBC & CB \rightarrow HB & HB \rightarrow HC \\ HC \rightarrow BC & aB \rightarrow ab & bB \rightarrow bb \\ bC \rightarrow bc & cC \rightarrow cc & \end{array}$$

Consider the grammar G with $\Sigma = \{a\}$ with

$$\begin{array}{lll} S \rightarrow \$Ca\# \mid a \mid \epsilon & Ca \rightarrow aaC & \$D \rightarrow \$C \\ C\# \rightarrow D\# \mid E & aD \rightarrow Da & aE \rightarrow Ea \\ \$E \rightarrow \epsilon & & \end{array}$$

Derivation

Expand the start symbol using one of its rules. Then expand the resulting string by replacing one of its substrings that matches the LHS of a rule by the RHS. Repeat until you get a string of terminals.

Derivation

Expand the start symbol using one of its rules. Then expand the resulting string by replacing one of its substrings that matches the LHS of a rule by the RHS. Repeat until you get a string of terminals.

For the rules

$$
\begin{aligned}
E &\to I \mid N \mid E + E \mid E * E \mid (E) \\
I &\to a \mid b \mid Ia \mid Ib \\
N &\to 0 \mid 1 \mid N0 \mid N1 \mid -N \mid +N
\end{aligned}
$$

we have

Derivation

Expand the start symbol using one of its rules. Then expand the resulting string by replacing one of its substrings that matches the LHS of a rule by the RHS. Repeat until you get a string of terminals.

For the rules

$$
\begin{aligned}
E &\to I \mid N \mid E+E \mid E*E \mid (E) \\
I &\to a \mid b \mid Ia \mid Ib \\
N &\to 0 \mid 1 \mid N0 \mid N1 \mid -N \mid +N
\end{aligned}
$$

we have

$$
\begin{aligned}
E &\Rightarrow E*E \Rightarrow E*N \Rightarrow E*-N \Rightarrow E*-N \Rightarrow E*-1 \\
&\Rightarrow (E)*-1 \Rightarrow (E+E)*-1 \Rightarrow (E+I)*-1 \\
&\Rightarrow (E+a)*-1 \Rightarrow (I+a)*-1 \Rightarrow (a+a)*-1
\end{aligned}
$$

Formal Definition

Definition

Let $G = (V, \Sigma, R, S)$ be a grammar. We say $\gamma_1 \alpha \gamma_2 \Rightarrow_G \gamma_1 \beta \gamma_1$, where $\gamma_1, \gamma_2 \alpha, \beta \in (V \cup \Sigma)^*$ if $\alpha \to \beta$ is a rule of G.

Formal Definition

Definition

Let $G = (V, \Sigma, R, S)$ be a grammar. We say $\gamma_1\alpha\gamma_2 \Rightarrow_G \gamma_1\beta\gamma_1$, where $\gamma_1, \gamma_2\alpha, \beta \in (V \cup \Sigma)^*$ if $\alpha \to \beta$ is a rule of G.
We say $\alpha \overset{*}{\Rightarrow}_G \beta$ if either $\alpha = \beta$ or there are $\alpha_0, \alpha_1, \ldots \alpha_n$ such that

$$\alpha = \alpha_0 \Rightarrow_G \alpha_1 \Rightarrow_G \alpha_2 \Rightarrow_G \cdots \Rightarrow_G \alpha_n = \beta$$

Formal Definition

Definition

Let $G = (V, \Sigma, R, S)$ be a grammar. We say $\gamma_1\alpha\gamma_2 \Rightarrow_G \gamma_1\beta\gamma_1$, where $\gamma_1, \gamma_2\alpha, \beta \in (V \cup \Sigma)^*$ if $\alpha \rightarrow \beta$ is a rule of G.
We say $\alpha \overset{*}{\Rightarrow}_G \beta$ if either $\alpha = \beta$ or there are $\alpha_0, \alpha_1, \ldots \alpha_n$ such that

$$\alpha = \alpha_0 \Rightarrow_G \alpha_1 \Rightarrow_G \alpha_2 \Rightarrow_G \cdots \Rightarrow_G \alpha_n = \beta$$

Notation

When G is clear from the context, we will write $\Rightarrow$ and $\overset{*}{\Rightarrow}$ instead of $\Rightarrow_G$ and $\overset{*}{\Rightarrow}_G$.

Language of a Grammar

Definition

The language of a grammar $G = (V, \Sigma, R, S)$, denoted $L(G)$ is the collection of strings over the terminals derivable from S using the rules in R.

Language of a Grammar

Definition

The language of a grammar $G = (V, \Sigma, R, S)$, denoted $L(G)$ is the collection of strings over the terminals derivable from S using the rules in R. In other words,

$$L(G) = \{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\}$$

Example I

Example

Consider the grammar G with $\Sigma = \{a, b, c\}$, $V = \{S, B, C, H\}$ and

$$
\begin{array}{lll}
S \rightarrow aSBC \mid aBC & CB \rightarrow HB & HB \rightarrow HC \\
HC \rightarrow BC & aB \rightarrow ab & bB \rightarrow bb \\
bC \rightarrow bc & cC \rightarrow cc &
\end{array}
$$

Example I

Example

Consider the grammar G with $\Sigma = \{a, b, c\}$, $V = \{S, B, C, H\}$ and

$$\begin{array}{lll} S \rightarrow aSBC \mid aBC & CB \rightarrow HB & HB \rightarrow HC \\ HC \rightarrow BC & aB \rightarrow ab & bB \rightarrow bb \\ bC \rightarrow bc & cC \rightarrow cc & \end{array}$$

Some derivations of G are

$$S \Rightarrow aBC \Rightarrow abC \Rightarrow abc$$

Example I

Example

Consider the grammar G with $\Sigma = \{a, b, c\}$, $V = \{S, B, C, H\}$ and

$$
\begin{array}{lll}
S \to aSBC \mid aBC & CB \to HB & HB \to HC \\
HC \to BC & aB \to ab & bB \to bb \\
bC \to bc & cC \to cc &
\end{array}
$$

Some derivations of G are

$$S \Rightarrow aBC \Rightarrow abC \Rightarrow abc$$

$$
\begin{aligned}
S &\Rightarrow aSBC \Rightarrow aaSBCBC \Rightarrow aaaBCBCBC \Rightarrow aaaBHBCBC \Rightarrow aaaBHCCBC \\
&\Rightarrow aaaBBCCBC \Rightarrow aaaBBCHBC \Rightarrow aaaBBCHCC \Rightarrow aaaBBCBCC \\
&\Rightarrow aaaBBHBCC \Rightarrow aaaBBHCCC \Rightarrow aaaBBBCCC \Rightarrow aaabBBCCC \\
&\Rightarrow aaabbBCCC \Rightarrow aaabbbCCC \Rightarrow aaabbbcCC \Rightarrow aaabbbccC \Rightarrow aaabbbccc
\end{aligned}
$$

Example I

Example

Consider the grammar G with $\Sigma = \{a, b, c\}$, $V = \{S, B, C, H\}$ and

$$
\begin{array}{lll}
S \rightarrow aSBC \mid aBC & CB \rightarrow HB & HB \rightarrow HC \\
HC \rightarrow BC & aB \rightarrow ab & bB \rightarrow bb \\
bC \rightarrow bc & cC \rightarrow cc &
\end{array}
$$

Some derivations of G are

$$S \Rightarrow aBC \Rightarrow abC \Rightarrow abc$$

$$
\begin{aligned}
S &\Rightarrow aSBC \Rightarrow aaSBCBC \Rightarrow aaaBCBCBC \Rightarrow aaaBHBCBC \Rightarrow aaaBHCCBC \\
&\Rightarrow aaaBBCCBC \Rightarrow aaaBBCHBC \Rightarrow aaaBBCHCC \Rightarrow aaaBBCBCC \\
&\Rightarrow aaaBBHBCC \Rightarrow aaaBBHCCC \Rightarrow aaaBBBCCC \Rightarrow aaabBBCCC \\
&\Rightarrow aaabbBCCC \Rightarrow aaabbbCCC \Rightarrow aaabbbcCC \Rightarrow aaabbbccC \Rightarrow aaabbbccc
\end{aligned}
$$

$$L(G) = \{a^n b^n c^n \mid n \geq 0\}$$

Example II

Example

Consider the grammar G with $\Sigma = \{a\}$ with

$$
\begin{array}{lll}
S \rightarrow \$Ca\# \mid a \mid \epsilon & Ca \rightarrow aaC & \$D \rightarrow \$C \\
C\# \rightarrow D\# \mid E & aD \rightarrow Da & aE \rightarrow Ea \\
\$E \rightarrow \epsilon & &
\end{array}
$$

Example II

Example

Consider the grammar G with $\Sigma = \{a\}$ with

$$
\begin{array}{lll}
S \rightarrow \$Ca\# \mid a \mid \epsilon & Ca \rightarrow aaC & \$D \rightarrow \$C \\
C\# \rightarrow D\# \mid E & aD \rightarrow Da & aE \rightarrow Ea \\
\$E \rightarrow \epsilon & &
\end{array}
$$

The following are derivations in this grammar

$$S \Rightarrow \$Ca\# \Rightarrow \$aaC\# \Rightarrow \$aaE \Rightarrow \$aEa \Rightarrow \$Eaa \Rightarrow aa$$

Example II

Example

Consider the grammar G with $\Sigma = \{a\}$ with

$$\begin{array}{lll} S \rightarrow \$Ca\# \mid a \mid \epsilon & Ca \rightarrow aaC & \$D \rightarrow \$C \\ C\# \rightarrow D\# \mid E & aD \rightarrow Da & aE \rightarrow Ea \\ \$E \rightarrow \epsilon & & \end{array}$$

The following are derivations in this grammar

$$S \Rightarrow \$Ca\# \Rightarrow \$aaC\# \Rightarrow \$aaE \Rightarrow \$aEa \Rightarrow \$Eaa \Rightarrow aa$$

$$S \Rightarrow \$Ca\# \Rightarrow \$aaC\# \Rightarrow \$aaD\# \Rightarrow \$aDa\# \Rightarrow \$Daa\# \Rightarrow \$Caa\#$$

Example II

Example

Consider the grammar G with $\Sigma = \{a\}$ with

$$
\begin{array}{lll}
S \rightarrow \$Ca\# \mid a \mid \epsilon & Ca \rightarrow aaC & \$D \rightarrow \$C \\
C\# \rightarrow D\# \mid E & aD \rightarrow Da & aE \rightarrow Ea \\
\$E \rightarrow \epsilon & &
\end{array}
$$

The following are derivations in this grammar

$$S \Rightarrow \$Ca\# \Rightarrow \$aaC\# \Rightarrow \$aaE \Rightarrow \$aEa \Rightarrow \$Eaa \Rightarrow aa$$

$$
\begin{aligned}
S &\Rightarrow \$Ca\# \Rightarrow \$aaC\# \Rightarrow \$aaD\# \Rightarrow \$aDa\# \Rightarrow \$Daa\# \Rightarrow \$Caa\# \\
&\Rightarrow \$aaCa\# \Rightarrow \$aaaaC\# \Rightarrow \$aaaaE \Rightarrow \$aaaEa \Rightarrow \$aaEaa
\end{aligned}
$$

Example II

Example

Consider the grammar G with $\Sigma = \{a\}$ with

$$
\begin{array}{lll}
S \to \$Ca\# \mid a \mid \epsilon & Ca \to aaC & \$D \to \$C \\
C\# \to D\# \mid E & aD \to Da & aE \to Ea \\
\$E \to \epsilon & &
\end{array}
$$

The following are derivations in this grammar

$$S \Rightarrow \$Ca\# \Rightarrow \$aaC\# \Rightarrow \$aaE \Rightarrow \$aEa \Rightarrow \$Eaa \Rightarrow aa$$

$$
\begin{aligned}
S &\Rightarrow \$Ca\# \Rightarrow \$aaC\# \Rightarrow \$aaD\# \Rightarrow \$aDa\# \Rightarrow \$Daa\# \Rightarrow \$Caa\# \\
&\Rightarrow \$aaCa\# \Rightarrow \$aaaaC\# \Rightarrow \$aaaaE \Rightarrow \$aaaEa \Rightarrow \$aaEaa \\
&\Rightarrow \$aEaaa \Rightarrow \$Eaaaa \Rightarrow aaaa
\end{aligned}
$$

Example II

Example

Consider the grammar G with $\Sigma = \{a\}$ with

$$
\begin{array}{lll}
S \rightarrow \$Ca\# \mid a \mid \epsilon & Ca \rightarrow aaC & \$D \rightarrow \$C \\
C\# \rightarrow D\# \mid E & aD \rightarrow Da & aE \rightarrow Ea \\
\$E \rightarrow \epsilon & &
\end{array}
$$

The following are derivations in this grammar

$$S \Rightarrow \$Ca\# \Rightarrow \$aaC\# \Rightarrow \$aaE \Rightarrow \$aEa \Rightarrow \$Eaa \Rightarrow aa$$

$$
\begin{aligned}
S \quad & \Rightarrow \$Ca\# \Rightarrow \$aaC\# \Rightarrow \$aaD\# \Rightarrow \$aDa\# \Rightarrow \$Daa\# \Rightarrow \$Caa\# \\
& \Rightarrow \$aaCa\# \Rightarrow \$aaaaC\# \Rightarrow \$aaaaE \Rightarrow \$aaaEa \Rightarrow \$aaEaa \\
& \Rightarrow \$aEaaa \Rightarrow \$Eaaaa \Rightarrow aaaa
\end{aligned}
$$

$L(G) = \{a^i \mid i \text{ is a power of } 2\}$