Lecture slides of the course

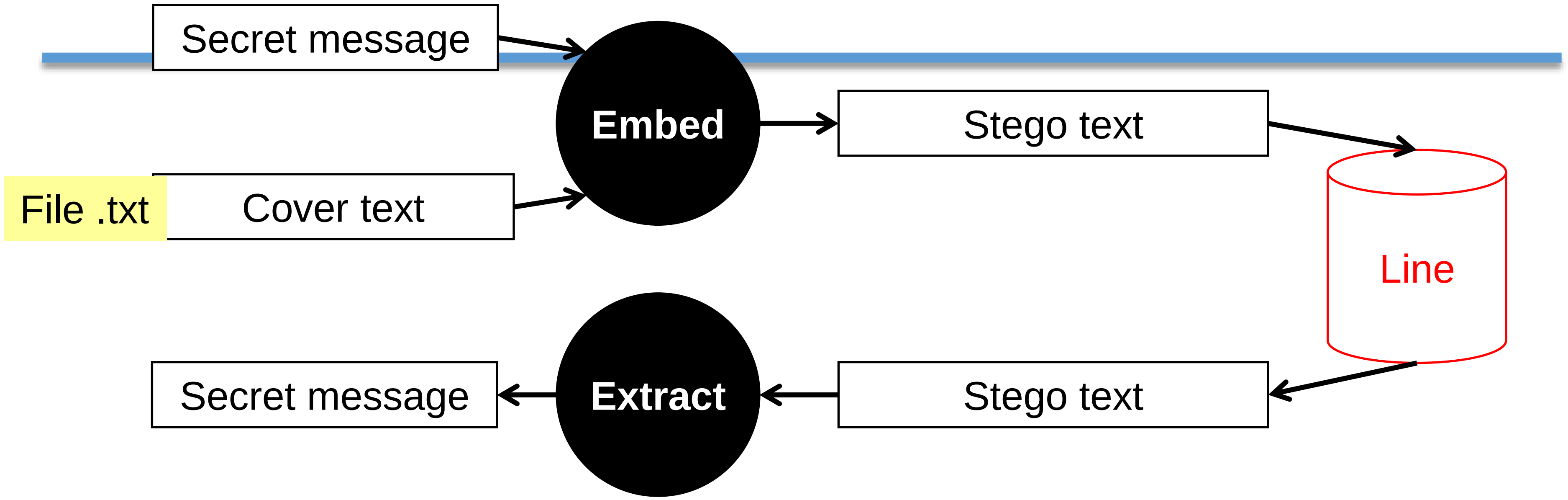
Information hiding & secret sharing

Text Steganography (P2)

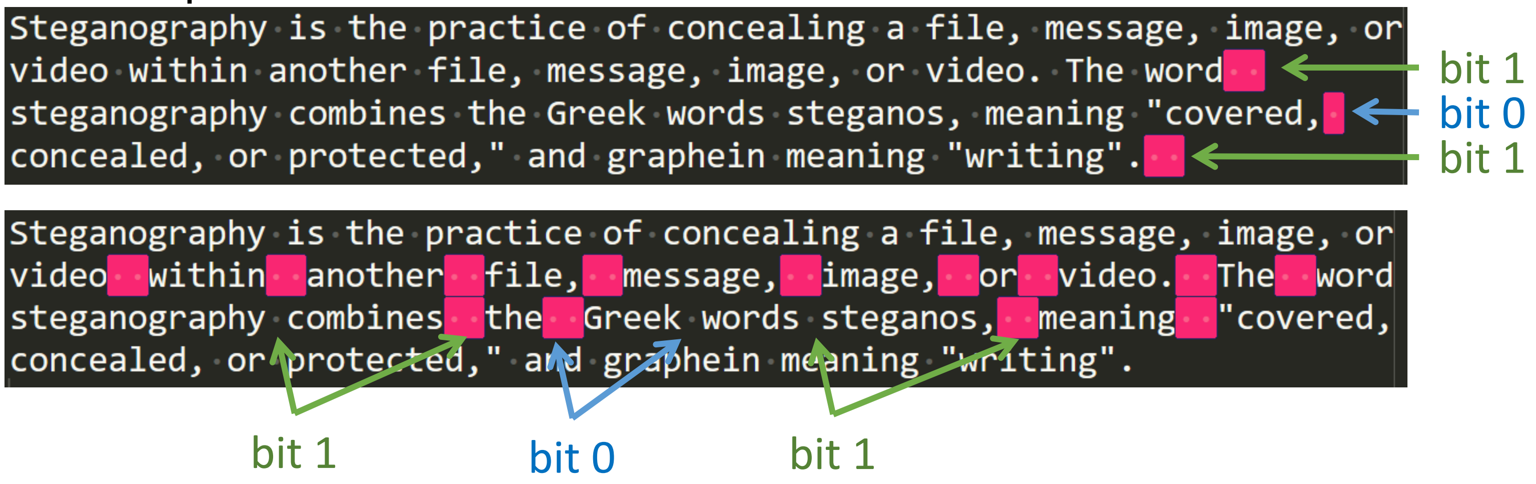
Phạm Trọng Nghĩa

ptnghia@fit.hcmus.edu.vn

Review



- White space methods



```
Steganography is the practice of concealing a file, message, image, or
video within another file, message, image, or video. The word
steganography combines the Greek words steganos, meaning "covered,
concealed, or protected," and graphein meaning "writing".
```

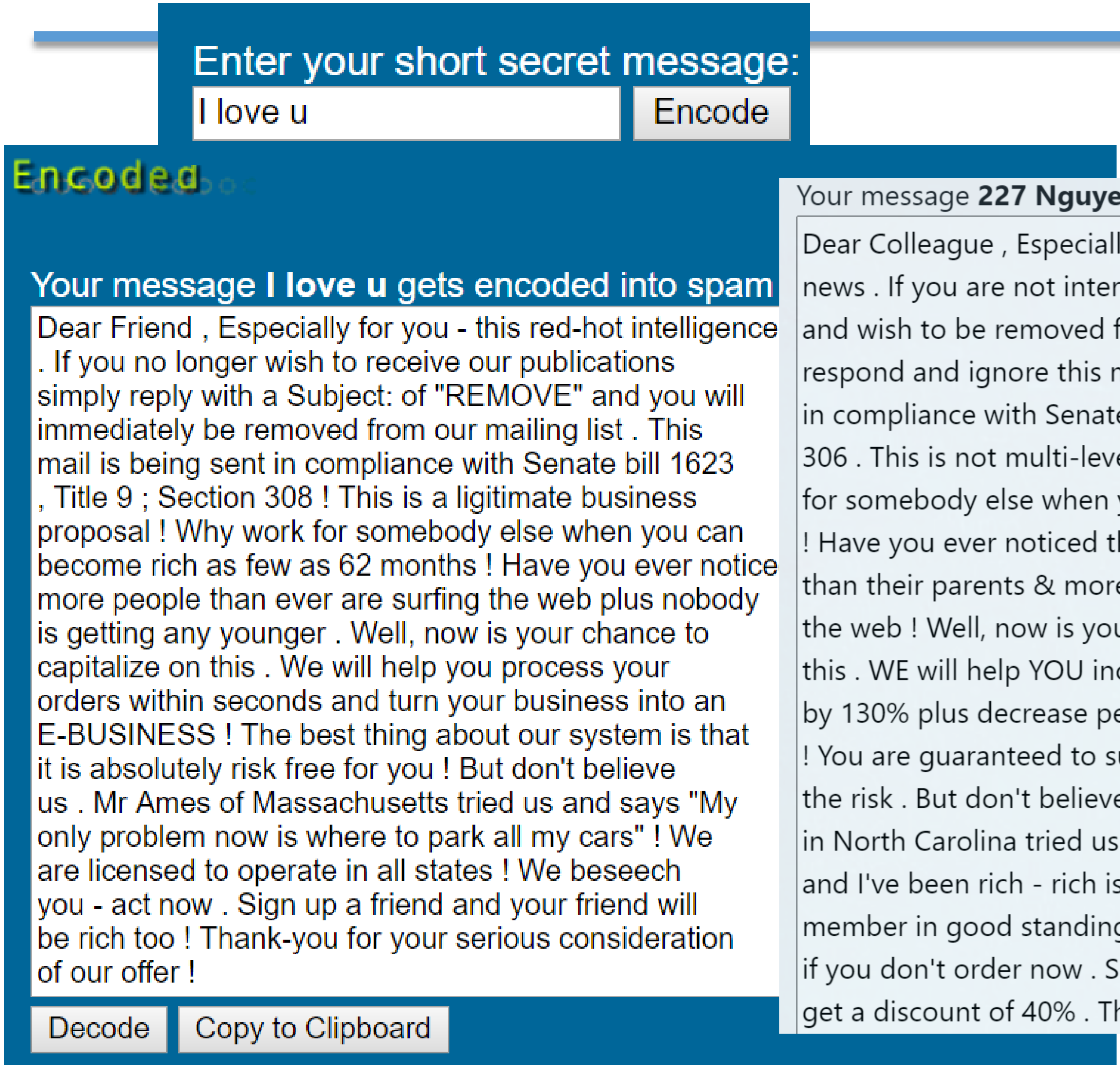
```
Steganography is the practice of concealing a file, message, image, or
video  within  another  file,  message,  image,  or  video.  The  word
steganography combines  the  Greek words steganos,  meaning  "covered,
concealed, or protected," and graphein meaning "writing".
```

- Syntatic methods
- Semantics methods

This session: hide secret information on text in a different approach

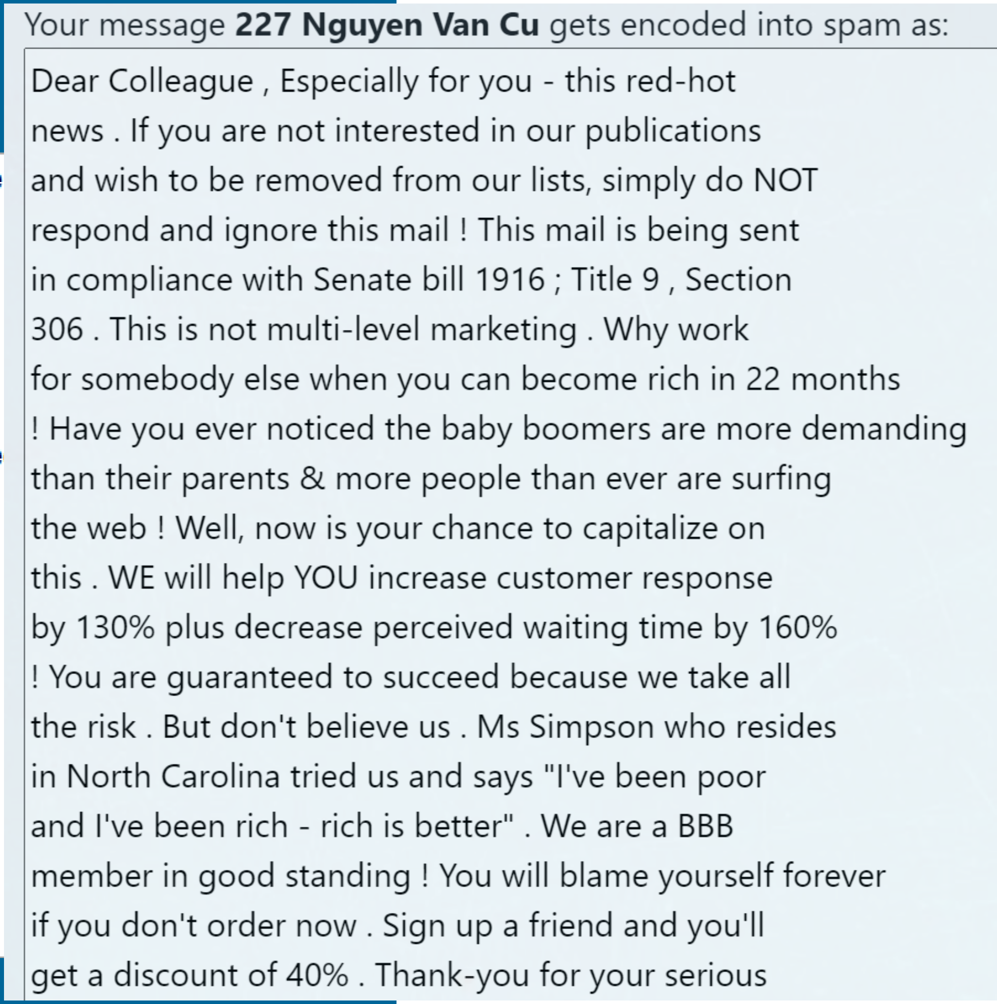
- The previous session: provide cover text and hide secret information in this document
- This session: from the secret, a cover text is generated with the secret in it
- Example: spammimic.com

Example: spammimic.com



Your message **I love u** gets encoded into spam

Dear Friend , Especially for you - this red-hot intelligence . If you no longer wish to receive our publications simply reply with a Subject: of "REMOVE" and you will immediately be removed from our mailing list . This mail is being sent in compliance with Senate bill 1623 , Title 9 ; Section 308 ! This is a ligitimate business proposal ! Why work for somebody else when you can become rich as few as 62 months ! Have you ever notice more people than ever are surfing the web plus nobody is getting any younger . Well, now is your chance to capitalize on this . We will help you process your orders within seconds and turn your business into an E-BUSINESS ! The best thing about our system is that it is absolutely risk free for you ! But don't believe us . Mr Ames of Massachusetts tried us and says "My only problem now is where to park all my cars" ! We are licensed to operate in all states ! We beseech you - act now . Sign up a friend and your friend will be rich too ! Thank-you for your serious consideration of our offer !



Your message **227 Nguyen Van Cu** gets encoded into spam as:

Dear Colleague , Especially for you - this red-hot news . If you are not interested in our publications and wish to be removed from our lists, simply do NOT respond and ignore this mail ! This mail is being sent in compliance with Senate bill 1916 ; Title 9 , Section 306 . This is not multi-level marketing . Why work for somebody else when you can become rich in 22 months ! Have you ever noticed the baby boomers are more demanding than their parents & more people than ever are surfing the web ! Well, now is your chance to capitalize on this . WE will help YOU increase customer response by 130% plus decrease perceived waiting time by 160% ! You are guaranteed to succeed because we take all the risk . But don't believe us . Ms Simpson who resides in North Carolina tried us and says "I've been poor and I've been rich - rich is better" . We are a BBB member in good standing ! You will blame yourself forever if you don't order now . Sign up a friend and you'll get a discount of 40% . Thank-you for your serious

Linguistic Steganography - CFG

- All of the embedding methods discussed so far, hide the secret information in a specific cover by applying an embedding algorithm.
- There exists steganographic applications that generate a digital object only for the purpose of being a cover for secret communication.
- We will see one such application that is based on **context-free grammars** (CFG)

Hiding data in artificially generated text - Requirement

- The letter frequencies in the text must resemble those of a natural language.
  - For English: E and T should be the most-common letters, and Z and Q should be the rarest.
- Most words in the text must be found in a good dictionary.
  - Any text may include some words, such as proper names, slang, and scientific terms, that may not be found in a given dictionary
  - If a computerized check finds too many such words, itshould flag the text as suspicious.
- The sentences in the text must be syntactically correct.
  - If an automatic syntax checker finds, for example, two consecutive verbs in the text, it should become suspicious.

CFG Review

- A context-free grammar (CFG) is a set of rewriting rules that can be **explicit** or **recursive**.
- The rules are used to generate strings of various patterns.
- The set of all strings generated by a particular CFG is the languagegenerated by the CFG.
- This set may be finite or (if the rules are recursive) infinite.
- The strings are considered sentences in the language.

Example: CFG

- **Start → noun verb**
- **noun →** Alice | Bob
- **verb →** is sending | is receiving

- The bold word (**Start**, **noun**, **verb**): called nonterminal symbol
- **Start** is special nonterminal symbol: start symbol
- Words not bolded (Alice, Bob, is, sending, receiving) called terminal symbol
- A rule of CFG has the form: L → R with L is a nonterminal symbol, R is string consist of nonterminal symbol or terminal symbol or both; the meaning of rule L → R is L can be expand into R
    - If we have rule L → R1, L → R2, ... Then can be shorted into: L → R1 | R2 | ...

A CFG consists of the following

1. A set of ***terminal*** symbols: These are the characters and words (the alphabet) that constitute the sentences generated by the grammar.
2. A set of ***nonterminal*** symbols: These are placeholders for patterns of terminal and nonterminal symbols.
   - In our examples, the nonterminals are typeset in boldface.
3. A set of ***productions***: These are rules for replacing (or rewriting) nonterminal symbols in a string with other nonterminal or terminal symbols.
   - A production has the form L→R where L is the nonterminal symbol that's replaced by the string R of nonterminal or terminal symbols.
4. A ***start*** symbol: a special nonterminal. The process of generating a string by the grammar should start with a production that has this symbol on its left-hand side

How to generate a string

1. Use the start symbol as the initial nonterminal.
2. Select a production that has the start symbol on the left-hand side and use it to replace the start symbol with the right-hand side of the production. This is the text generated so far.
3. Select a nonterminal symbol in the text, find a production that has this non-terminal on the left side, and replace the nonterminal with the right hand side of theproduction.
4. Repeat Step 3 until the resulting text consists of just terminal symbols.

Generate string from a CFG – Ex 1

```
<program> →begin <stmt_list> end
<stmt_list>→<stmt> | <stmt> ; <stmt_list>
<stmt>     →<var> = <expr>
<var>      →a | b | c | d
<expr>     →<term> + <term> | <term> - <term>
<term>     →<var> | const
```

How to check if a string is a sentence of the language?

Derivation

- Let consider the sentence: “begin a = b + 7 end”

```
<program>  =>begin <stmt_list> end => begin <stmt> end
           =>begin <var> = <expr> end
           =>begin a = <expr>  end
           =>begin a = <term> + <term> end
           =>begin a = <var> + <term> end
           =>begin a = b + <term> end
           =>begin a = b + const end
```

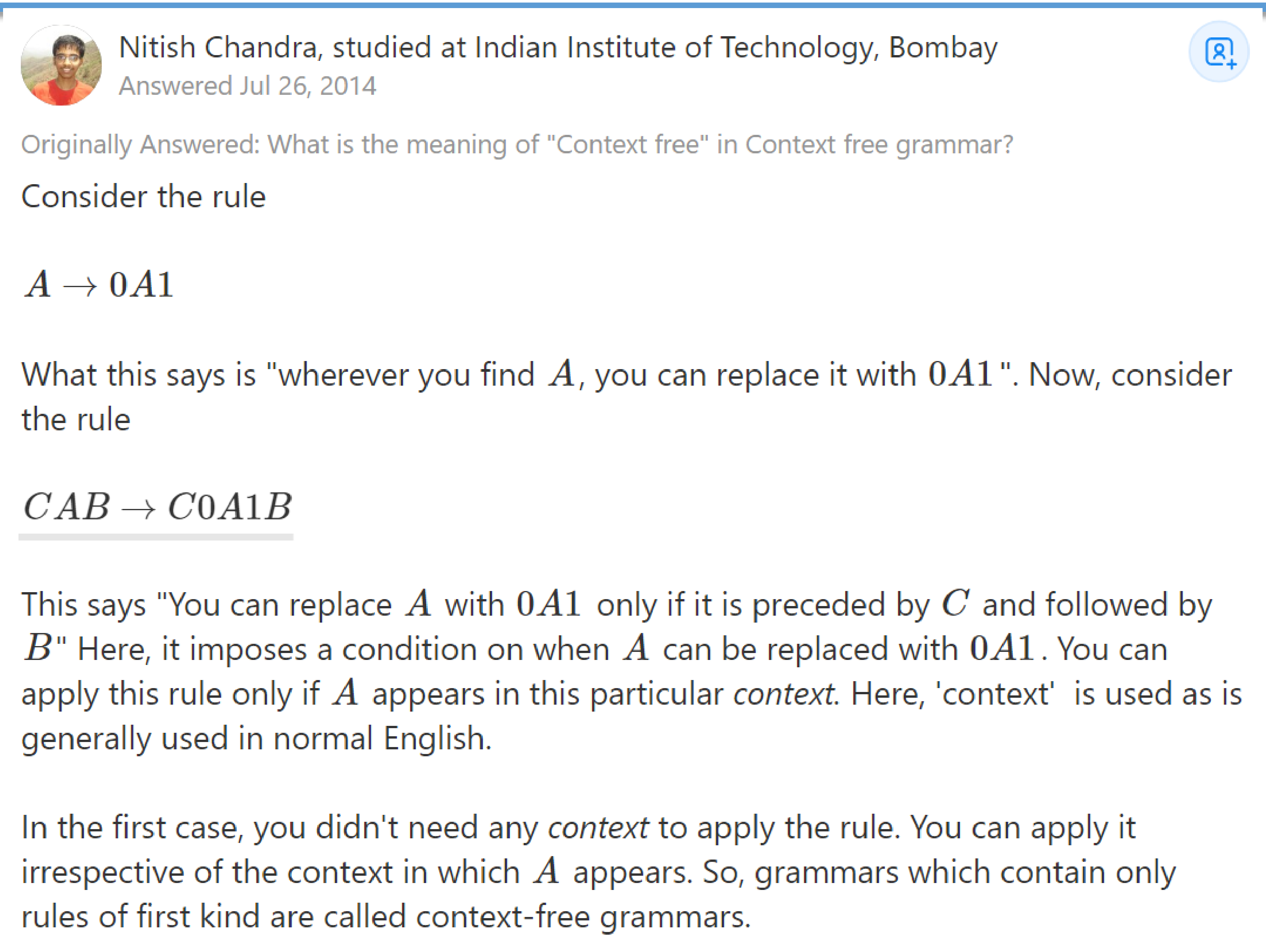
Generate string from a CFG – Ex 2

- Given CFG
  - **Start → noun verb**
  - **noun →** Alice | Bob
  - **verb →** is sending | is receiving
- Generate string
  - **Start**
  - **noun verb** (rule: **Start → noun verb**)
  - Alice **verb** (rule : **noun →** Alice)
  - Alice is receiving (rule : **verb → is receiving**)
- With a different way of choosing nouns and verbs, a different string will be produced

Generate string from a CFG – Ex 3

- CFG
  - **Start → expression**
  - **expression → number | expression + expression | expression - expression**
  - **number → digit | numberdigit**
  - **digit →** 0 | 1 | ... | 9
- Generate string:
  - **Start**
  - **expression** (rule: **Start → expression**)
  - **expression + expression** (rule: **expression → expression + expression**)
  - **expression – expression + expression** (rule: **expression → expression - expression**)
  - **number – expression + expression** (rule: **expression → number**)
  - **numberdigit – expression + expression** (rule: **number → numberdigit**)
  - **digitdigit – expression + expression** (rule: **number → digit**)
  - 27 – **expression + expression** (rule: **digit** → 2, **digit** → 7)
  - ...
  - 27 – 9 + 123
- How many sequences can be derived from this CFG?

Why call context free



Nitish Chandra, studied at Indian Institute of Technology, Bombay
Answered Jul 26, 2014

Originally Answered: What is the meaning of "Context free" in Context free grammar?

Consider the rule

$$A \to 0A1$$

What this says is "wherever you find A, you can replace it with $0A1$". Now, consider the rule

$$CAB \to C0A1B$$

This says "You can replace A with $0A1$ only if it is preceded by C and followed by B" Here, it imposes a condition on when A can be replaced with $0A1$. You can apply this rule only if A appears in this particular *context*. Here, 'context' is used as is generally used in normal English.

In the first case, you didn't need any *context* to apply the rule. You can apply it irrespective of the context in which A appears. So, grammars which contain only rules of first kind are called context-free grammars.

Linguistic Steganography - CFG

The idea of converting secret information into a harmless text was proposed by Peter Wayner (1992), based on the context-free syntax CFG (Context-Free Grammar).

Content:

- CFG → text
- Embedded: secret message + CFG → cover text
- Extract: cover text + CFG → secret message

How to use CFG to generate text containing secret bits?

Idea

- During sequence generation from the CFG, for a nonterminal symbol, there can be many alternatives → use these options to embed bits
- Example with rule L → R1 | R2:
  - Bit 0 = R1
  - Bit 1 = R2
- Example with rule L → R1 | R2 | R3 | R4:
  - 2 bit 00 = R1
  - 2 bit 01 = R2
  - 2 bit 10 = R3
  - 2 bit 11 = R4

How to use CFG to generate text containing secret bits?

Example 1

Given CFG

- **Start → noun verb**
- **noun →** Alice | Bob | Fred | Barney
- **verb →** went fishing **where** | went bowling **where**
- **where →** in Iowa | in Minnesota

Secret bits: 1101

Embedding:

- **Start**
  1101
- **noun verb** 1101
- Barney **verb** 1101
- Barney went fishing **where** 1101
- Barney went fishing in Minnesota 1101

How to use CFG to generate text containing secret bits?

Example 2

Given CFG

- **Start** → **noun verb**
- **noun** → Alice | Bob | Fred | Barney
- **verb** → went fishing **where** | went bowling **where**
- **where** → in Iowa | in Minnesota

Secret bits : 11

<u>Embedding</u>

- **Start** 11
- **noun verb** 11
- Barney **verb** 11

Embedded bit is exhausted, but generation sequence is not complete yet
➔what to do?

How to use CFG to generate text containing secret bits?

Example 2

Given CFG

- **Start** → **noun verb**
- **noun** → Alice | Bob | Fred | Barney
- **verb** → went fishing **where** | went bowling **where**
- **where** → in Iowa | in Minnesota

Secret bits : 11

Embedding

- **Start** 11
- **noun verb** 11
- Barney **verb** 11100

One way is to keep embedding **one bit 1 and many bit 0** until the generation sequence is completed
When extracting, we will get a bit string ending in 100... and can easily cut this tail

How to use CFG to generate text containing secret bits?

Example 2

Given CFG

- **Start** → **noun verb**
- **noun** → Alice | Bob | Fred | Barney
- **verb** → went fishing **where** | went bowling **where**
- **where** → in Iowa | in Minnesota

Secret bits : 11

<u>Embedding</u>

- **Start** 11
- **noun verb** 11
- Barney **verb** 11100...
- Barney went bowling **where** 11100...
- Barney went bowling in Iowa 11100...

How to use CFG to generate text containing secret bits?

Example 3

Given CFG

- **Start** → **noun verb**
- **noun** → Alice | Bob | Fred | Barney | Mary
- **verb** → went fishing **where** | went bowling **where**
- **where** → in Iowa | in Minnesota

noun has 5 options
→ how many bits can be embedded?

How to use CFG to generate text containing secret bits?

Example 3

Given CFG

- **Start → noun verb**
- **noun →** Alice | Bob | Fred | Barney | Mary
- **verb →** went fishing **where** | went bowling **where**
- **where →** in Iowa | in Minnesota

One way is to just use the first 4 options and embed 2 bits

How to extract secret bit from cover text containing secret bit?

Idea

- CFG required
- The problem to be solved is to find the path from the start symbol to the text string containing the secret bit; Once we have found this path, we can easily know the embedded secret bits
- How to find the way?
- One way is to use DFS (Depth First Search)

How to extract secret bit from cover text containing secret bit?

Example 1

Given CFG

- **Start → noun verb**
- **noun →** Alice | Bob | Fred | Barney
- **verb →** went fishing **where |** went bowling **where**
- **where →** in Iowa **|** in Minnesota

Given a text containing a secret bit : “Barney went bowling in Iowa”

$\boxed{\textbf{Start}}$

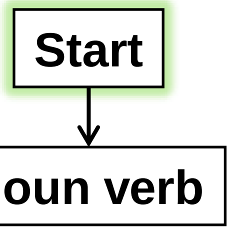
How to extract secret bit from cover text containing secret bit?

Example 1

Given CFG

- **Start → noun verb**
- **noun →** Alice | Bob | Fred | Barney
- **verb →** went fishing **where** | went bowling **where**
- **where →** in Iowa | in Minnesota

Given a text containing a secret bit : “Barney went bowling in Iowa”


$$\begin{tikzcd}\boxed{\textbf{Start}} \arrow[d] \\ \boxed{\textbf{noun verb}}\end{tikzcd}$$

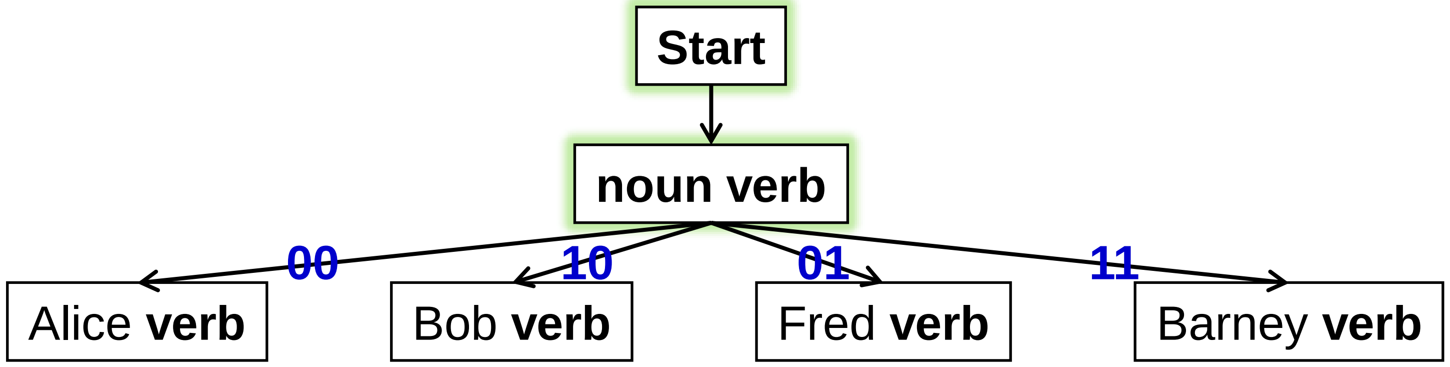
How to extract secret bit from cover text containing secret bit?

Example 1

Given CFG

- **Start** → **noun verb**
- **noun** → Alice | Bob | Fred | Barney
- **verb** → went fishing **where** | went bowling **where**
- **where** → in Iowa | in Minnesota

Given a text containing a secret bit : “Barney went bowling in Iowa”



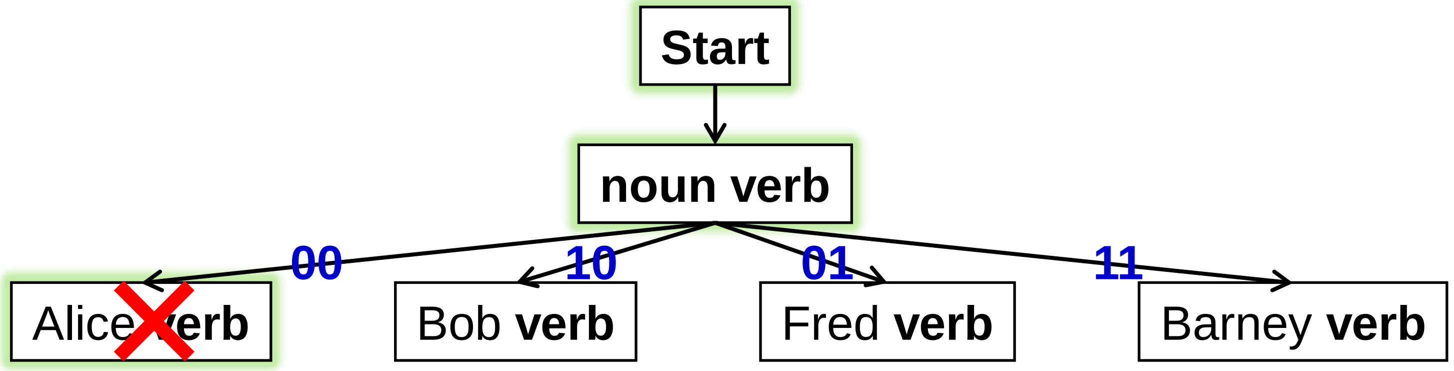
How to extract secret bit from cover text containing secret bit?

Example 1

Given CFG

- **Start → noun verb**
- **noun →** Alice | Bob | Fred | Barney
- **verb →** went fishing **where** | went bowling **where**
- **where →** in Iowa | in Minnesota

Given a text containing a secret bit : “Barney went bowling in Iowa”



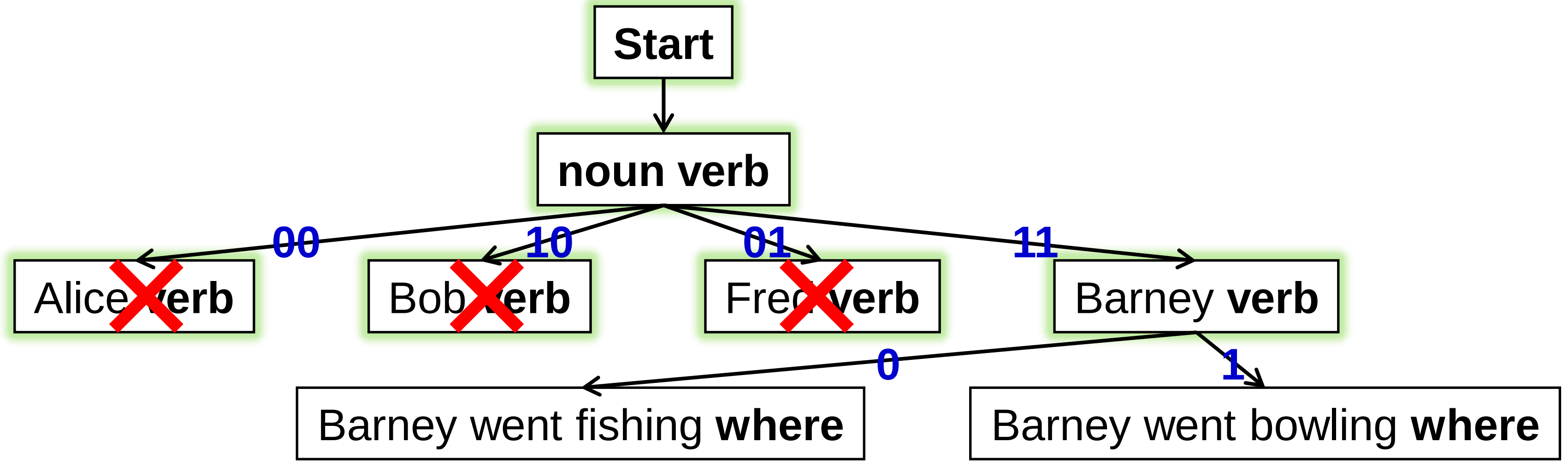
How to extract secret bit from cover text containing secret bit?

Example 1

Given CFG

- **Start → noun verb**
- **noun →** Alice | Bob | Fred | Barney
- **verb →** went fishing **where** | went bowling **where**
- **where →** in Iowa | in Minnesota

Given a text containing a secret bit : “Barney went bowling in Iowa”



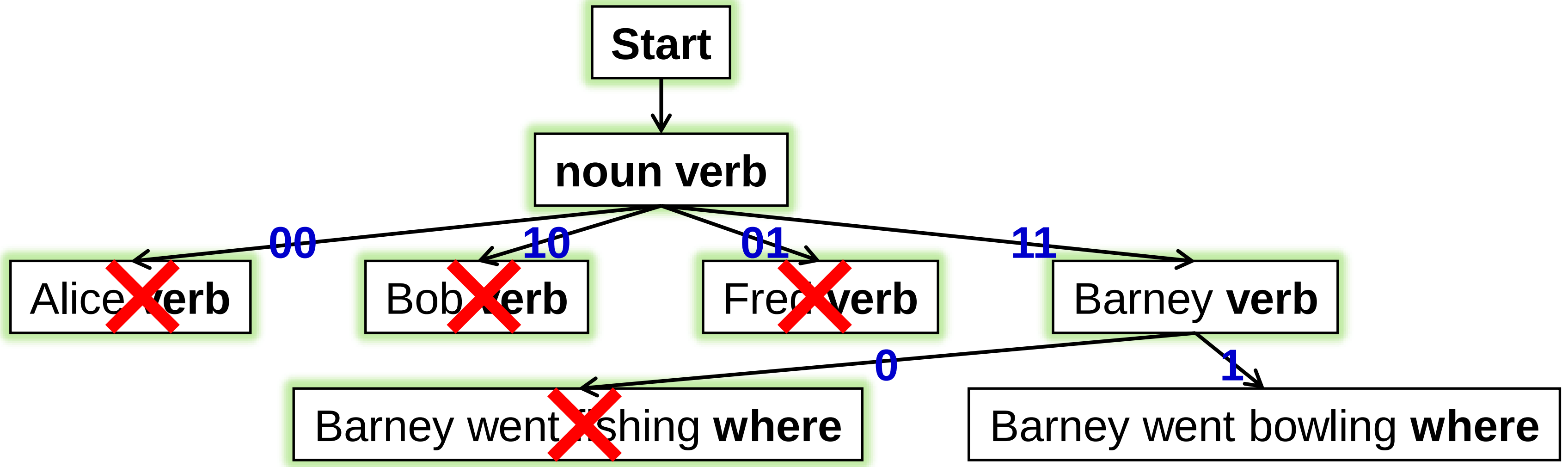
How to extract secret bit from cover text containing secret bit?

Example 1

Given CFG

- **Start** → **noun verb**
- **noun** → Alice | Bob | Fred | Barney
- **verb** → went fishing **where** | went bowling **where**
- **where** → in Iowa | in Minnesota

Given a text containing a secret bit : "Barney went bowling in Iowa"



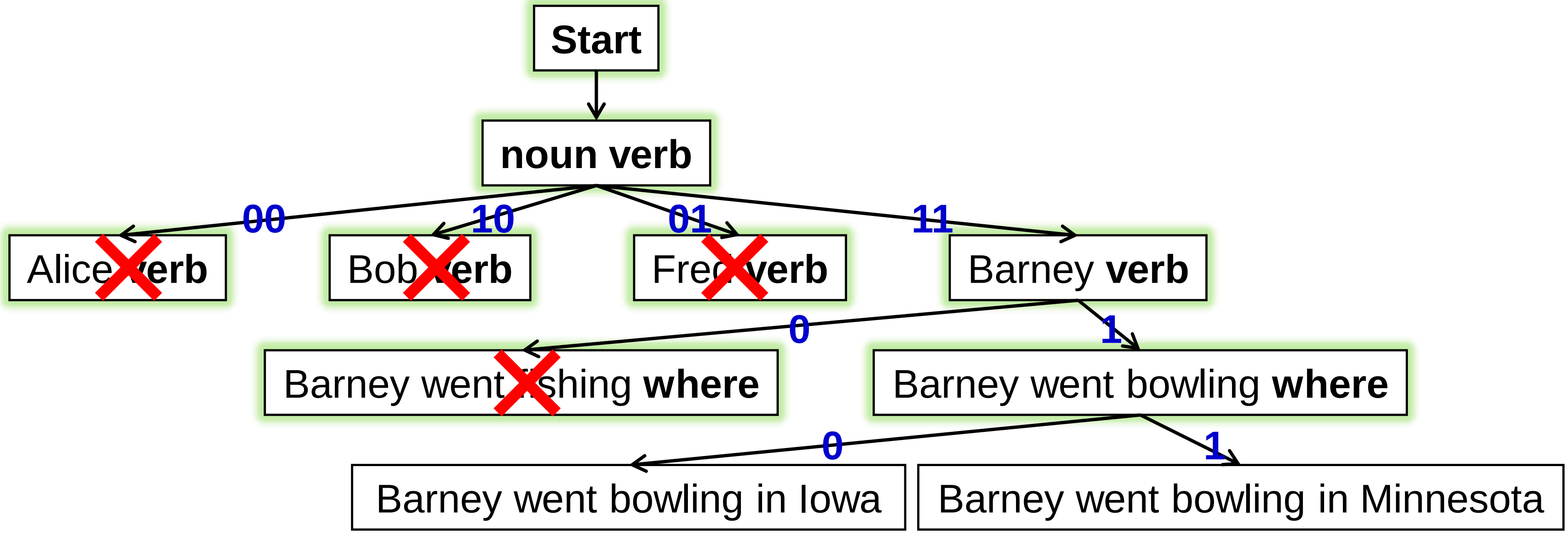
How to extract secret bit from cover text containing secret bit?

Example 1

Given CFG

- **Start → noun verb**
- **noun** → Alice | Bob | Fred | Barney
- **verb** → went fishing **where** | went bowling **where**
- **where** → in Iowa | in Minnesota

Given a text containing a secret bit : “Barney went bowling in Iowa”



Start
noun verb
00
10
01
11
Alice verb
Bob verb
Fred verb
Barney verb
0
1
Barney went fishing where
Barney went bowling where
0
1
Barney went bowling in Iowa
Barney went bowling in Minnesota

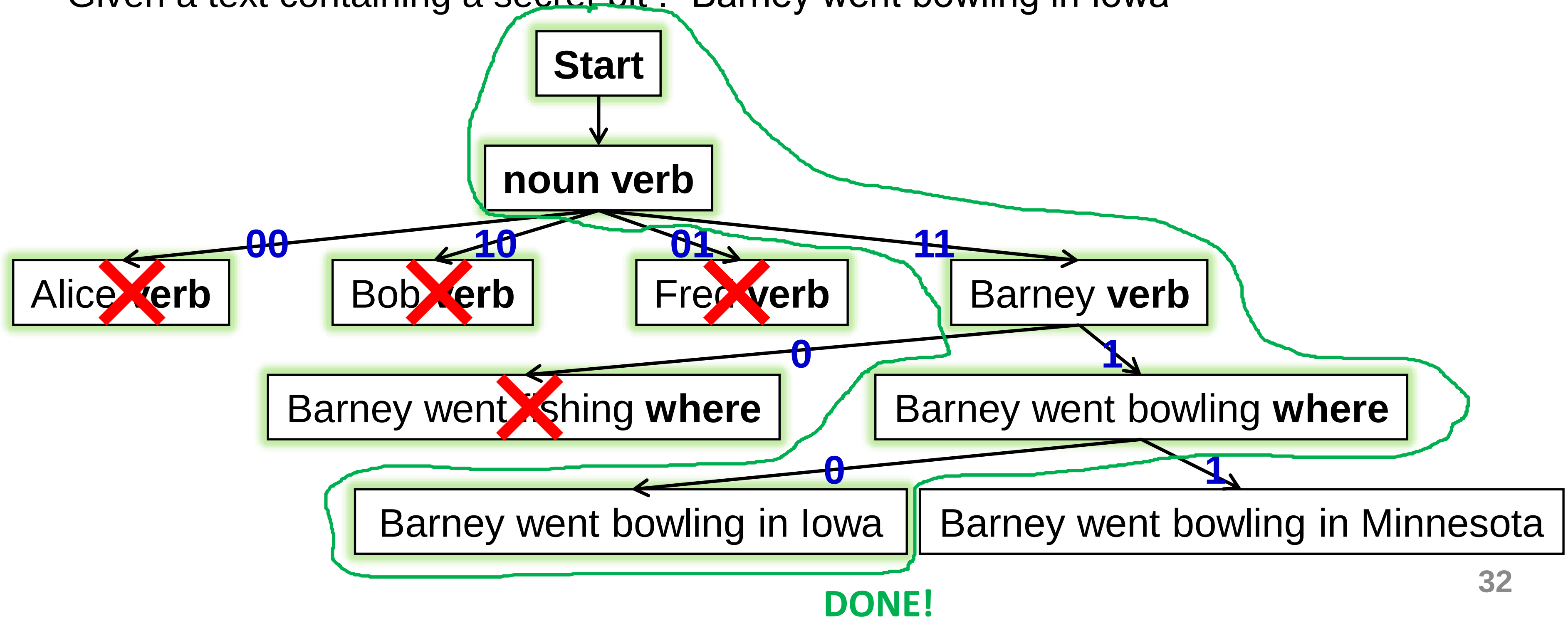
How to extract secret bit from cover text containing secret bit?

Example 1

Given CFG

- **Start** → **noun verb**
- **noun** → Alice | Bob | Fred | Barney
- **verb** → went fishing **where** | went bowling **where**
- **where** → in Iowa | in Minnesota

Given a text containing a secret bit : “Barney went bowling in Iowa”


Start
noun verb
00
10
01
11
Alice verb
Bob verb
Fred verb
Barney verb
0
1
Barney went fishing where
Barney went bowling where
0
1
Barney went bowling in Iowa
Barney went bowling in Minnesota
DONE!

How to extract secret bit from cover text containing secret bit?

Example 2

Given CFG

- **Start → name action | whobe where**
- **name →** Alice | Bob
- **action →** is here | is there
- **whobe →** Alice is | Bob was
- **where →** here | there

Embed bit string 101:

- **Start** 101
- **whobe where** 101
- Alice is **where** 101
- Alice is there 101

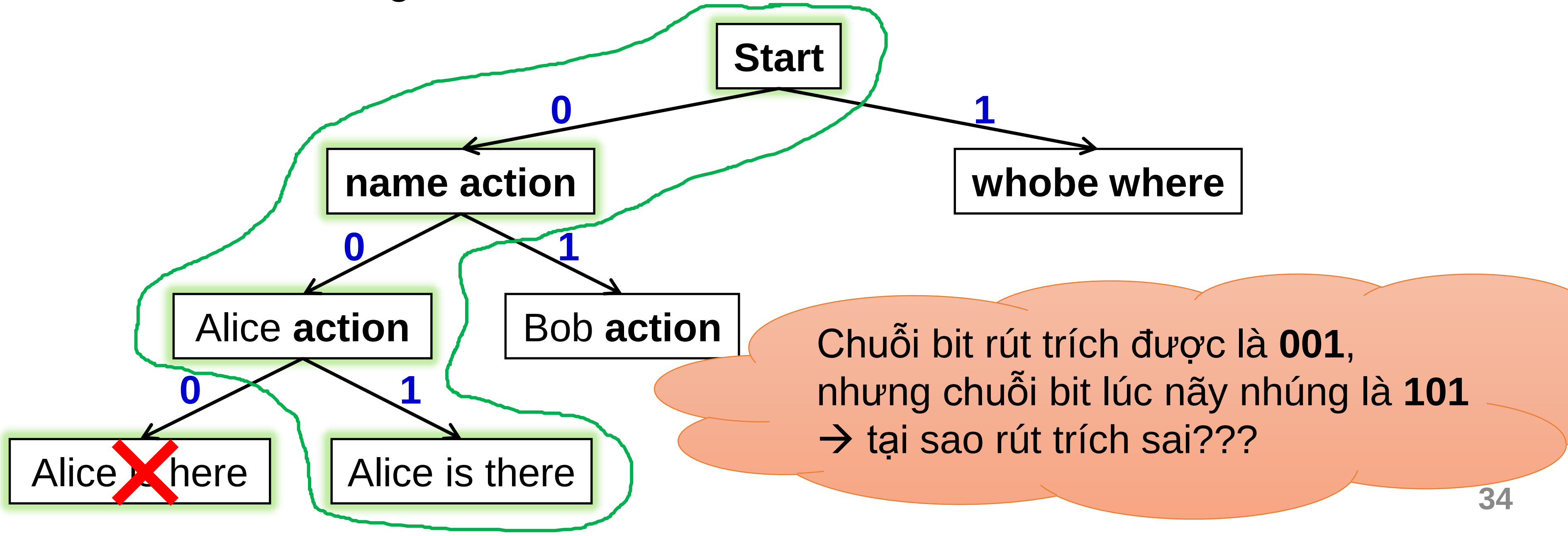
How to extract secret bit from cover text containing secret bit?

Example 2

Given CFG

- **Start → name action | whobe where**
- **name →** Alice | Bob
- **action →** is here | is there
- **whobe →** Alice is | Bob was
- **where →** here | there

Extract from string “Alice is there”



Chuỗi bit rút trích được là **001**, nhưng chuỗi bit lúc nãy nhúng là **101** → tại sao rút trích sai???

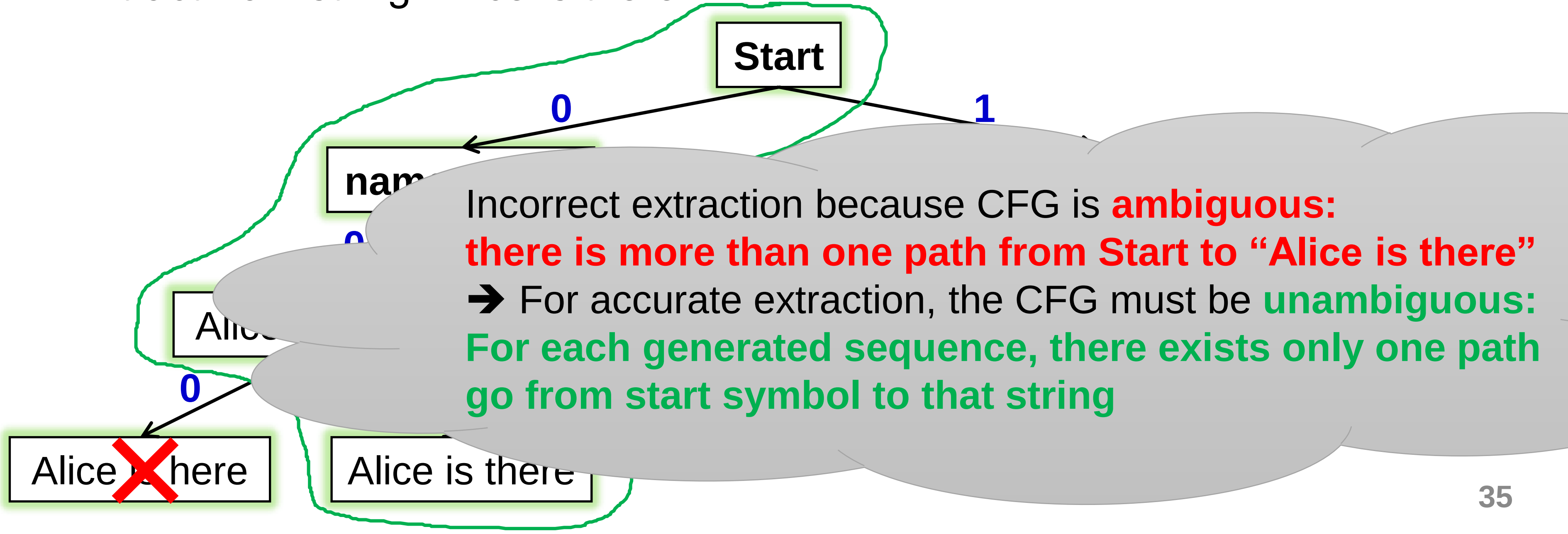
How to extract secret bit from cover text containing secret bit?

Example 2

Given CFG

- **Start → name action | whobe where**
- **name →** Alice | Bob
- **action →** is here | is there
- **whobe →** Alice is | Bob was
- **where →** here | there

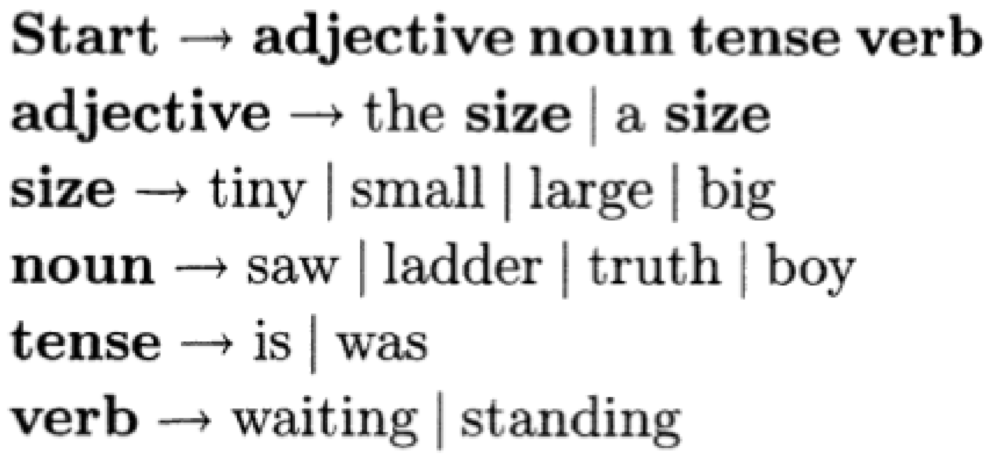
Extract from string “Alice is there”



Incorrect extraction because CFG is **ambiguous:**
there is more than one path from Start to “Alice is there”
➔ For accurate extraction, the CFG must be **unambiguous:**
For each generated sequence, there exists only one path go from start symbol to that string

Quiz 1

- 0100110



Start → **adjective noun tense verb**

adjective → the **size** | a **size**

size → tiny | small | large | big

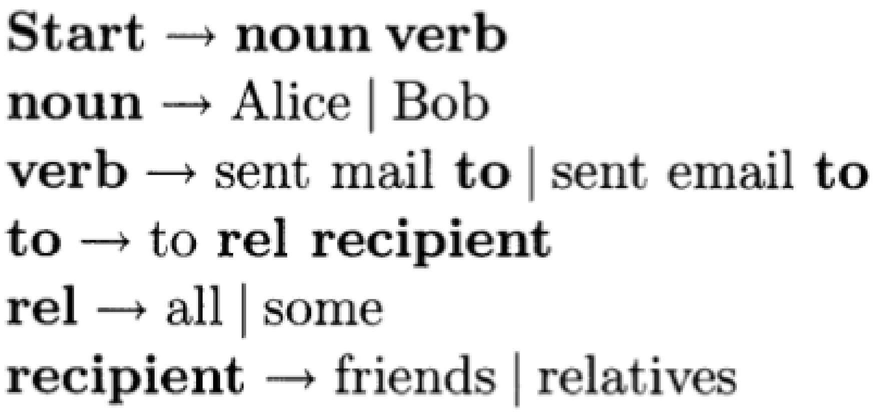
noun → saw | ladder | truth | boy

tense → is | was

verb → waiting | standing

Quiz 2

- “Alice sent email to all relatives.”



Start → **noun verb**
noun → Alice | Bob
verb → sent mail **to** | sent email **to**
to → to **rel recipient**
rel → all | some
recipient → friends | relatives

Analyze

Analyze on the method of hiding secret information on documents using CFG

→ To achieve high invisibility and large capacity, it takes a lot of time to design the CFG ☹