*Link to the Video ->* <https://youtu.be/eK4L4F0Wm3A>

# Title of the project

### **Title: Formalising Alexa’s Language: A CKY-Based Validation**

**Course:** CSCI-549-01W Automata Theory

**Semester:** Summer I 2025

**Student:** Mukesh Ravichandran

**Contact:** mravichandr@leomail.tamuc.edu

**TABLE OF CONTENTS**

[Title of the project 1](#_r312nqnk3b7)

[Title: Formalising Alexa’s Language: A CKY-Based Validation 1](#_ihm1fwt7mr7a)

[Purpose 3](#_m1855u4f1cmb)

[Domain Description 3](#_jdn2m9xqddq5)

[Step 1 : List of Sentence 4](#_9w5p8xbb711z)

[Valid Sentence 4](#_uibxc5z40tby)

[Invalid Sentence 5](#_ggx6o83eg3v3)

[Logic Design of the Parser 5](#_45nn63sckaws)

[Technique Used : 5](#_jdn50wq7vsgz)

[Analyzing the Sentences 5](#_e2mcfxax6mky)

[Identifying Sentence Patterns 6](#_1mip6zq9ysti)

[Step 2 : Building a Context Free Grammar (CFG) 8](#_7jgysq2m6gr8)

[Lexical Production Rules 10](#_xd5oa52pedum)

[Step 3: Conversion of CFG to CNF 12](#_e632hwry29mb)

[What is CNF? 12](#_en0z61i3jku0)

[Why Convert to CNF? 12](#_bkrtr47xiosd)

[CNF Structural Rules 13](#_ef5yjtcb9583)

[Step 4 : Building the CKY table 16](#_h1ypnm3m9ht6)

[Valid Sentence 17](#_ux14yxq8n2px)

[Invalid Sentence 32](#_3onohybk4euy)

[Conclusion 36](#_ufm343ka42yt)

[Resources 36](#_pruxn9chvyu5)

## Purpose

For this project, I’m working on building a grammar that can handle voice commands used in home automation—like the ones we give to Alexa. For example, commands such as *“Alexa, wake me up at 6:30 AM”* or *“Turn off the kitchen lights”* might sound simple, but there’s a lot going on behind the scenes to figure out what the user actually wants and how to respond.

I’m using context-free grammar to represent how these kinds of commands are structured. This connects closely to topics in natural language processing and computational linguistics, which are important areas in computing. While the examples come from smart home settings, the goal is to show how we can break down natural language into something a computer can understand and act on. It also keeps the project focused on real-world language that’s both interesting and relevant to computing.

## Domain Description

My Own Alexa ! I wanted to create an alexa of my own , which is powered uses the CKY parsing technique. This domain is specially for the home automation, sentences and the scenario is similar to typical household , the sentence used in , i did some research on top sentences used .

When you interact with Alexa, your spoken commands are recorded and transcribed.

What i understood is the amazon does not releases any data due to policy, Hence I hit a roadblock of getting individualized or aggregate lists of the most frequently used sentences by Alexa users

Asking questions (e.g., "What is the weather today?", "What time is it?")

Requests for music (e.g., "Play [song name]")

| Category | Example Sentences |
| --- | --- |
| Questions | "What is the weather today?" |
| Music | "Play [song name]" |
| Jokes | "Tell me a joke" |
| Greetings | "Good morning, Alexa" |
| Facts | "Who wrote [book name]?" |
| Shopping | "Buy me [item]" |
| Smart Home Control | "Dim the lights" |
| Social Media | "YouTube" |
| Rhetorical/Unclear | "What is the meaning of life?" |

## Step 1 : List of Sentence

### Valid Sentence

Basic Control Commands

1. Alexa turn on the light.
2. Alexa turn off the fan.
3. Alexa lock the front door.
4. Alexa set the morning alarm.
5. Alexa dim the bedroom lights.
6. Alexa turn on the heater.
7. Alexa turn off the oven.
8. Alexa switch on the study lamp.

Time-based Command

1. Alexa wake me up at 6.

Contextual or Multi-Device Commands

1. Alexa play music in the kitchen.
2. Alexa open the living room window.
3. Alexa close the living room curtains.
4. Alexa start the robot vacuum.
5. Alexa increase the speaker volume.
6. Alexa decrease the fan speed.

Conditional or Routine Commands

1. Alexa stop the music right now.
2. Alexa read today’s news headlines.
3. Alexa play a relaxing music playlist.
4. Alexa pause the movie playback.
5. Alexa resume playing the podcast.

Query-style Commands

1. Alexa is the front door locked?
2. Alexa what’s the temperature inside?
3. Alexa is the living room light on?

### Invalid Sentence

1. Alexa turn on turn on today's (**Structural Failure-**Invalid sequence of phrases **)**
2. Alexa play music on the kitchen (**Contextual Failure - “on”** wrong grammatical**)**
3. Alexa lock the morning (**Malformed Phrase Failure-** Bad phrase**)**

# Logic Design of the Parser

## Technique Used :

The CKY (Cocke-Kasami-Younger) parsing algorithm is a classic parsing technique used for context-free grammars (CFGs) in Chomsky Normal Form (CNF). It determines whether a sentence can be generated by a given CFG and can also be used to construct a parse tree.

identifying every unique word (or group of words acting as a single unit) and categorizing it based on its grammatical function. These categories will become the *pre-terminal* symbols in our grammar, which directly produce the words (*terminals*).

## Analyzing the Sentences

The sentences are broken down , to find the structure "bottom-up" analysis of all 23 valid sentences.

examined each sentence and extracted its core components

* **Phrasal Verbs:** Verbs composed of multiple words (e.g., turn on, wake up) are treated as a single unit to simplify the rules.
* **Compound Modifiers:** Words that act as a single adjective (e.g., living room) are treated as a single unit.
* **Simplification:** In the sentence "Alexa resume playing the podcast," we simplify the structure to [Verb: resume] [Noun Phrase: the podcast] to fit our command pattern, keeping the grammar minimal.

Based on the analysis of all 23 sentences, we derived the following categories and their corresponding words:

#### **Identifying Sentence Patterns**

we now shift from a "bottom-up" to a "top-down" perspective. The goal of this step is to identify the recurring structures and patterns in the 23 valid sentences.

**Findings:**

Our analysis reveals a clear, hierarchical structure:

* A Sentence is composed of a WakeWord followed by a Verb Phrase or a Question Phrase.
* Verb Phrases and Question Phrases are built from our lexical categories (V\_CMD, V\_Q, NP, PP, etc.).
* Noun Phrases and Prepositional Phrases are themselves built from smaller, well-defined components.

**The Primary Sentence Structure** At the highest level, every single sentence follows one fundamental pattern: it begins with the wake word, which is then followed by a phrase that represents a complete thought (either a command or a question).

**Structure:** [WakeWord] [ActionablePhrase]

**Example:** Alexa + turn on the light

Commands: Alexa, [Action] [Object] [Extra Info].

Alexa, turn on the light. (Action + Object)

Alexa, play music in the kitchen. (Action + Object + Location)

Alexa, stop the music right now. (Action + Object + Time)

Alexa, wake me up at 6. (Action + Pronoun + Time)

Questions: Alexa, [Question Word] [Object] [State]?

Alexa, is the front door locked? (Yes/No Question)

Alexa, what’s the temperature inside? (Wh- Question)

**2. Types of Actionable Phrases** The [ActionablePhrase] component can be broken down into two distinct types:

* **A. Command Verb Phrases (VP):** These phrases instruct Alexa to perform an action. They are the most common pattern in our sentence list.
  + **Pattern:** [Verb] [Object] [Optional Extra Info]
  + **Examples:**
    - turn on the light -> V\_CMD + NP
    - play music in the kitchen -> V\_CMD + NP + PP
    - stop the music right now -> V\_CMD + NP + ADV
    - wake me up at 6 -> V\_CMD + PRO + PP
* **B. Question Phrases (QP):** These phrases ask Alexa for information about the state of an object.
  + **Pattern:** [Question Verb] [Object] [Optional State]
  + **Examples:**
    - is the front door locked -> V\_Q + NP + STATE
    - what’s the temperature inside -> V\_Q + NP + STATE
    - is the living room light on -> V\_Q + NP + STATE

**3. Common Sub-Structures (Phrase Patterns)** Within the command and question phrases, we identified smaller, reusable patterns:

* **A. Noun Phrases (NP):** This pattern describes the "object" of the action or question.
  + **Pattern:** (Determiner) + (Adjective) + Noun
  + **Examples:**
    - the light -> DET + N
    - the front door -> DET + ADJ + N
    - music -> N (The simplest form of an NP)
    - 6 -> NUM (A number can also function as a noun phrase in our context)
* **B. Prepositional Phrases (PP):** This pattern provides extra information, typically about location or time.
  + **Pattern:** Preposition + Noun Phrase
  + **Example:** in the kitchen -> P + NP

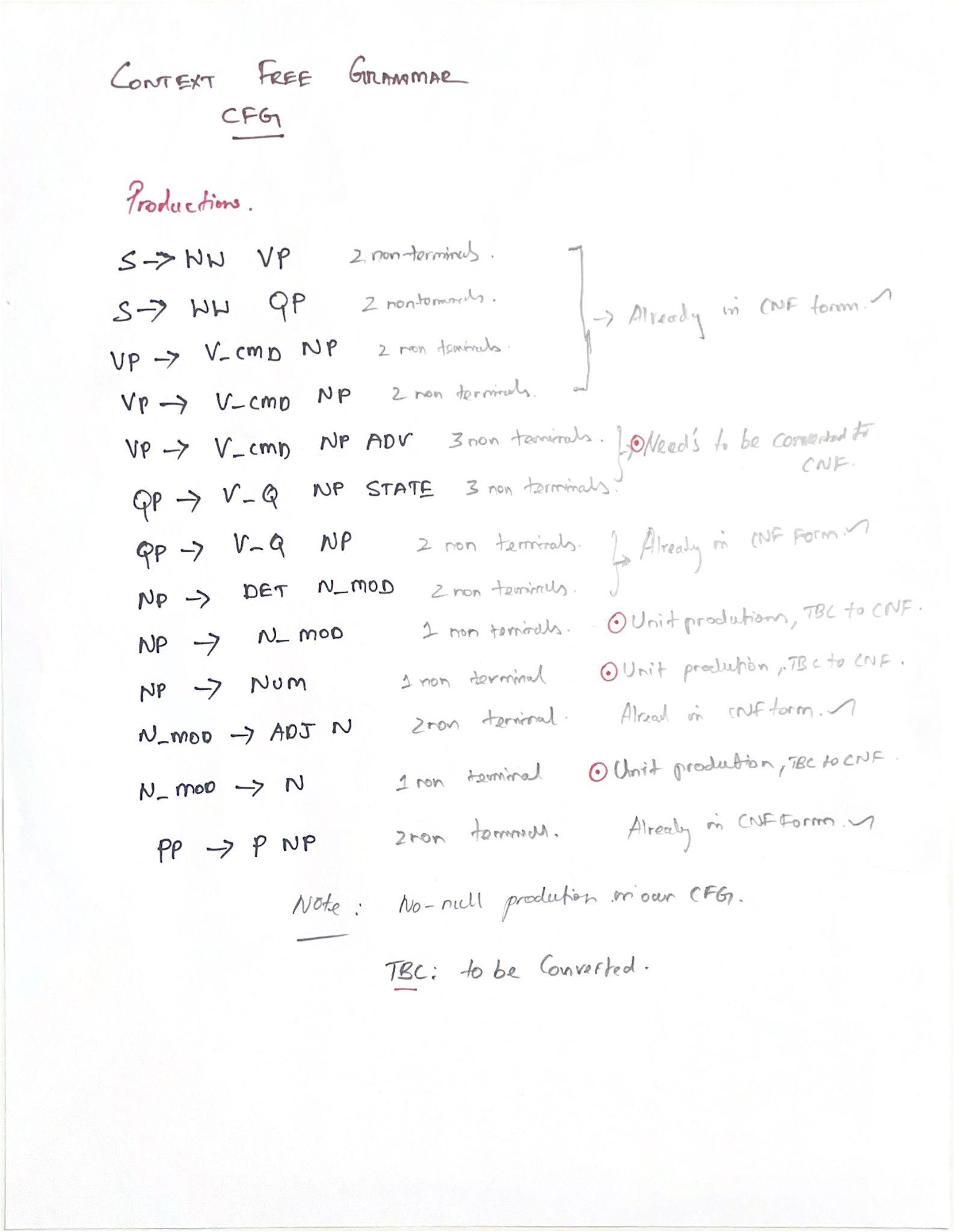
## Step 2 : Building a Context Free Grammar (CFG)

Context-Free Grammar capable of parsing a specific set of voice commands directed at the virtual assistant, "Alexa.

One CFG for all the sentence formal set of **production rules**. A production rule consists of a single non-terminal symbol on the left (representing a concept like a "sentence" or "noun phrase") and a sequence of symbols on the right, connected by the arrow -> which means "can be composed of."

The combination of all these rules forms our final Context-Free Grammar (CFG).

We will construct the rules in a top-down fashion, starting from the S (Sentence) symbol and progressively defining each component based on our analysis.



**1. Sentence-Level Rules (S)** These are the highest-level rules that define what constitutes a valid sentence in our language. Based on our analysis, a sentence is always a Wake Word followed by either a command (Verb Phrase) or a question (Question Phrase).

* S -> WW VP
* S -> WW QP

**2. Phrase-Level Rules (VP, QP, PP)** These rules define the structure of our core phrase types. They directly correspond to the patterns we found for commands and questions.

* VP -> V\_CMD NP (e.g., *lock the door*)
* VP -> V\_CMD NP PP (e.g., *play music in the kitchen*)
* VP -> V\_CMD NP ADV (e.g., *stop the music right now*)
* VP -> V\_CMD PRO PP (e.g., *wake me up at 6*)
* QP -> V\_Q NP STATE (e.g., *is the door locked*)
* QP -> V\_Q NP (e.g., *what's the temperature*)
* PP -> P NP (e.g., *in the kitchen*)

**3. Sub-Phrase and Word-Combination Rules (NP, N\_MOD)** These rules define how smaller components, like Noun Phrases, are built. We use the N\_MOD (Modified Noun) symbol to elegantly handle nouns that may or may not be preceded by an adjective.

* NP -> DET N\_MOD (e.g., *the front door*)
* NP -> N\_MOD (e.g., *music*)
* NP -> NUM (e.g., *6*)
* N\_MOD -> ADJ N (e.g., *front door*)
* N\_MOD -> N (e.g., *door*)

### 

### 

### **Lexical Production Rules**

These rules connect the grammatical categories from the structural rules to the actual words (terminals) defined in the lexicon. For brevity, rules with multiple terminals are combined using the | (or) operator.



## Step 3: Conversion of CFG to CNF

### **What is CNF?**

Chomsky Normal Form (CNF) is a standardized format for a Context-Free Grammar. A grammar is in CNF if all of its production rules adhere to one of two simple forms:

1. A -> B C (A non-terminal produces exactly two other non-terminals).
2. A -> 'a' (A non-terminal produces exactly one terminal/word).

The start symbol is the only exception and is not allowed to be on the right-hand side of any rule.

### **Why Convert to CNF?**

Converting a CFG to CNF is a crucial step for computational linguistics and parser implementation. The primary advantage of CNF is that its simple, predictable structure is required by many standard parsing algorithms, most notably the **CKY (Cocke-Kasami-Younger-) algorithm**. The CKY algorithm uses the binary nature of CNF rules to efficiently determine if a given string can be generated by the grammar. This standardization removes ambiguity and simplifies the logic required to build a parser.

**The Conversion Methodology**

**Eliminate NULL Productions - Not applicable here**

For completeness, a full CFG-to-CNF conversion also includes steps to eliminate NULL Values and isolate terminals. Our CFG was designed cleanly, so it contains no empty strings, and its terminals are already isolated in lexical rules, so those steps are not needed here

**Eliminate Unit Productions (UNIT)**

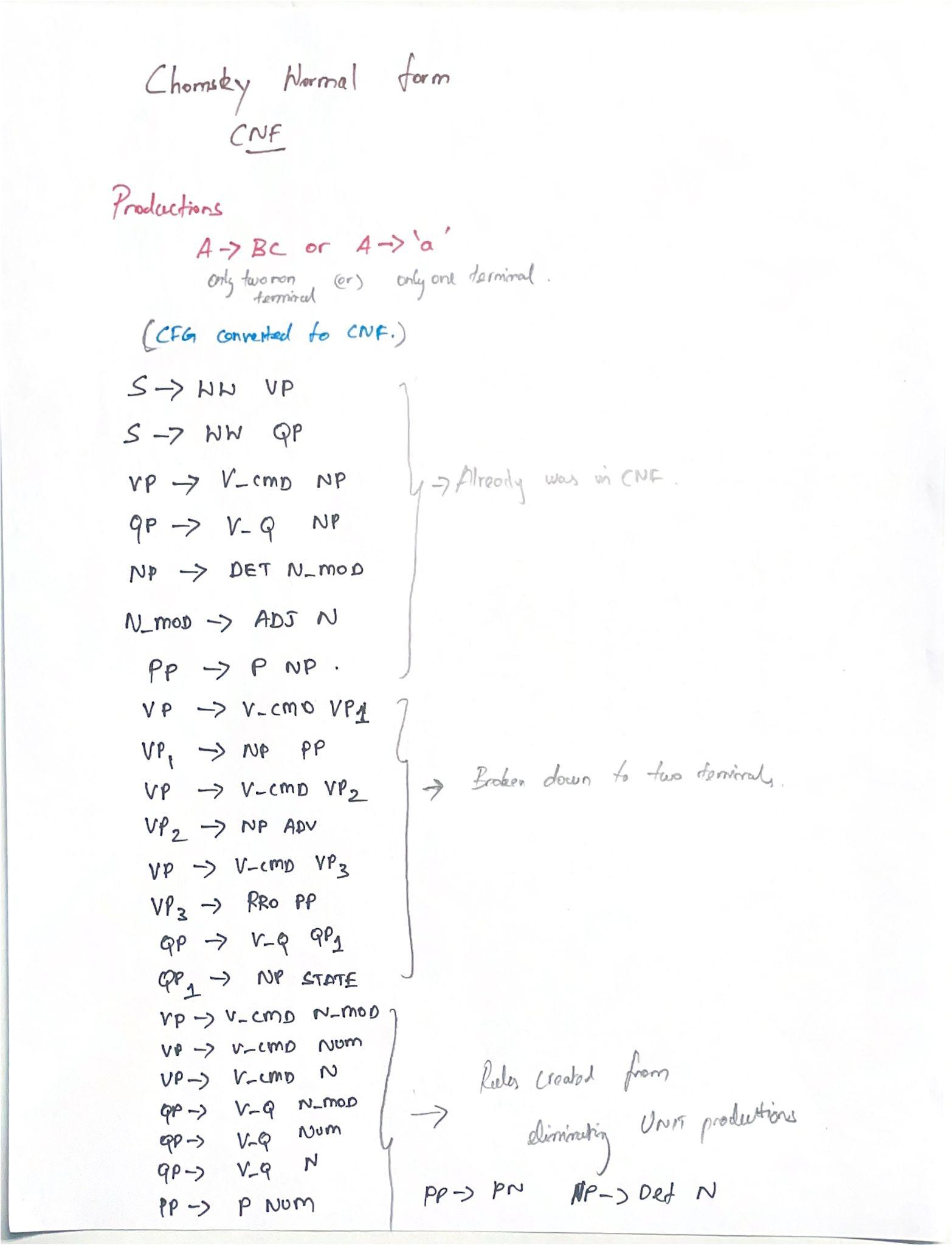
A unit production is a rule of the form A -> B, where one non-terminal simply points to another. These must be eliminated. The process involves replacing the unit rule by creating new rules that directly connect A to whatever B produces.

**Binarize Long Rules (BIN)**

Any production rule with more than two non-terminals on the right-hand side (e.g., A -> B C D) must be broken down into a cascade of rules that each have only two non-terminals.

Of course! Here are the contents of the "CNF Structural Rules (A -> B C)" listed clearly for your convenience.

### **CNF Structural Rules**



**1. Top-Level Sentence Rules:**

* S -> WW VP
* S -> WW QP

**2. Original Rules Already in Binary Format:**

* VP -> V\_CMD NP
* QP -> V\_Q NP
* NP -> DET N\_MOD
* N\_MOD -> ADJ N
* PP -> P NP

**3. Rules Created from Binarizing Long Rules :**

* VP -> V\_CMD VP1
* VP1 -> NP PP
* VP -> V\_CMD VP2
* VP2 -> NP ADV
* VP -> V\_CMD VP3
* VP3 -> PRO PP
* QP -> V\_Q QP1
* QP1 -> NP STATE

**4. Rules Created from Eliminating Unit Productions :**

* VP -> V\_CMD N\_MOD
* VP -> V\_CMD NUM
* VP -> V\_CMD N
* QP -> V\_Q N\_MOD
* QP -> V\_Q NUM
* QP -> V\_Q N
* PP -> P N\_MOD
* PP -> P NUM
* PP -> P N
* NP -> DET N

## Step 4 : Building the CKY table

CKY table to parse each sentence

To able to determine if the given sentence is able to satisfy the condition with respect to the grammar of the language

That is done by the process called parsing

Using rules of grammar determining of the sentence is able to pass

**Difference between Derivation and parsing**

Starting from S , we can work , and form sentence

However Parsing is reverse of it , given a sentence, we determine whether

Whether we will reach S,

**How do we conclude ?**

Given a sentence , if we are able to reach to the start symbol S , then it is valid

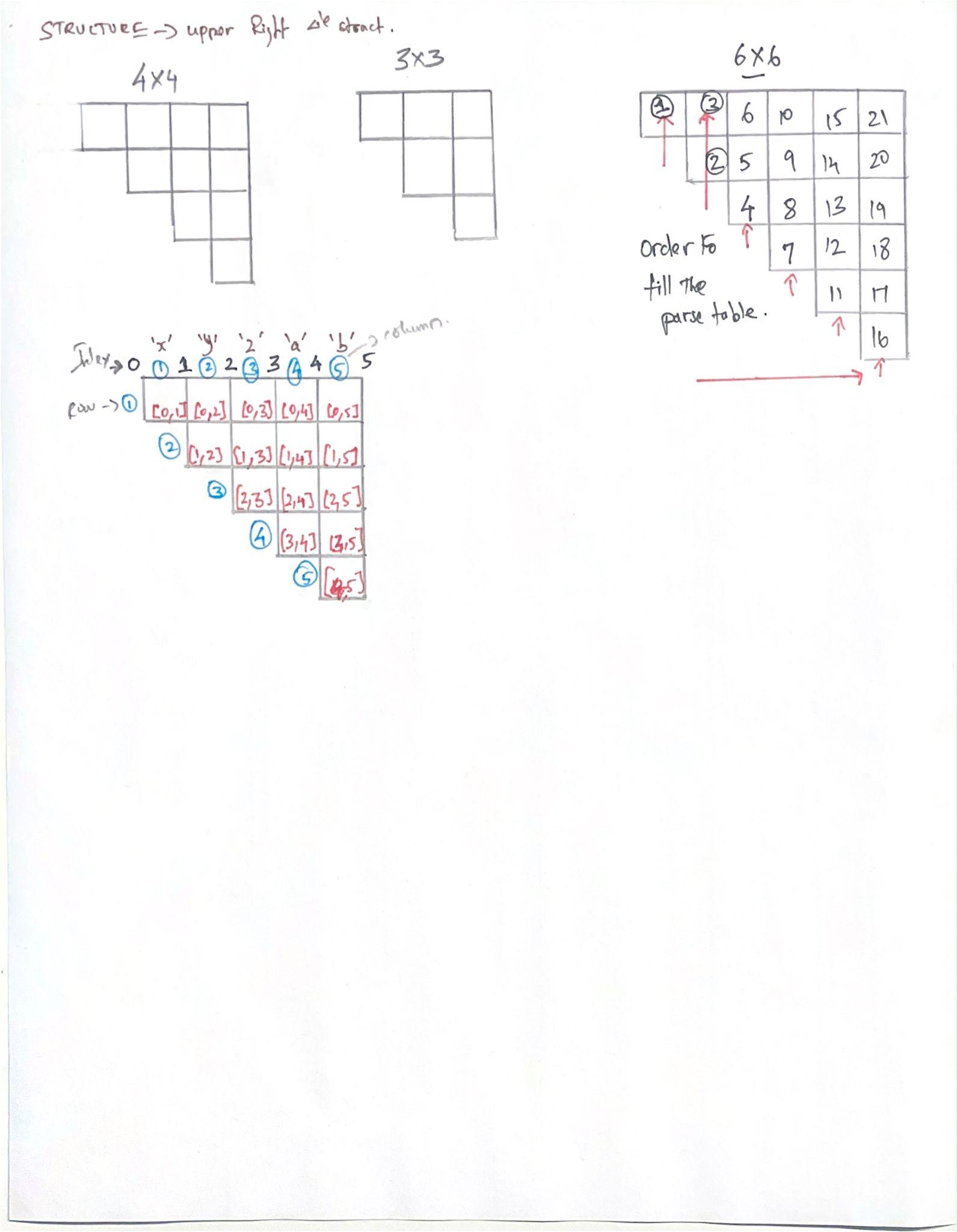
Then the sentence considering here is not valid

CKY is based on CNF Grammar

Given sentence is valid if the start Symbol S is reachable

Parsing - Parsing exactly opposite of what is derivation

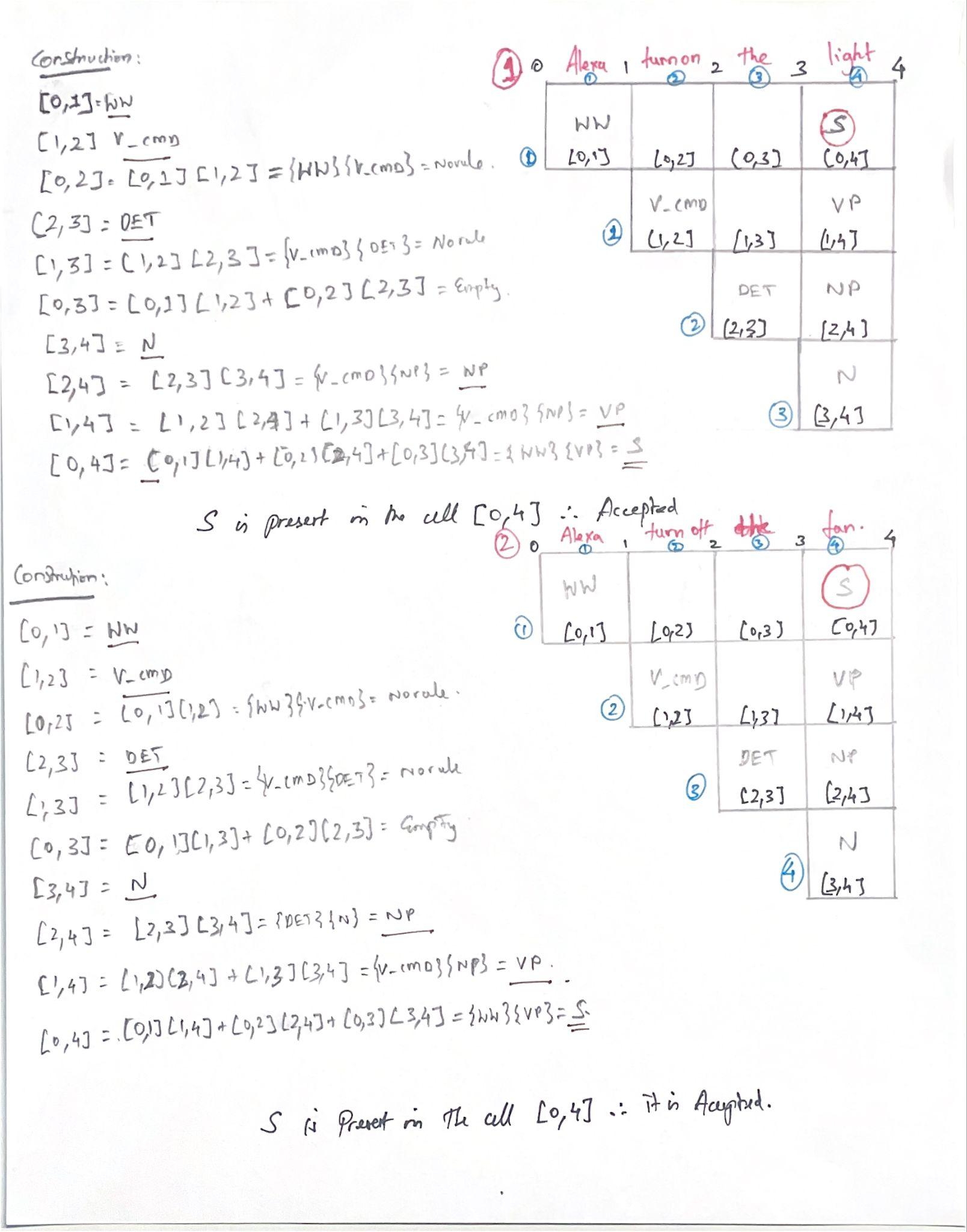
All the tables are manually hand drawn

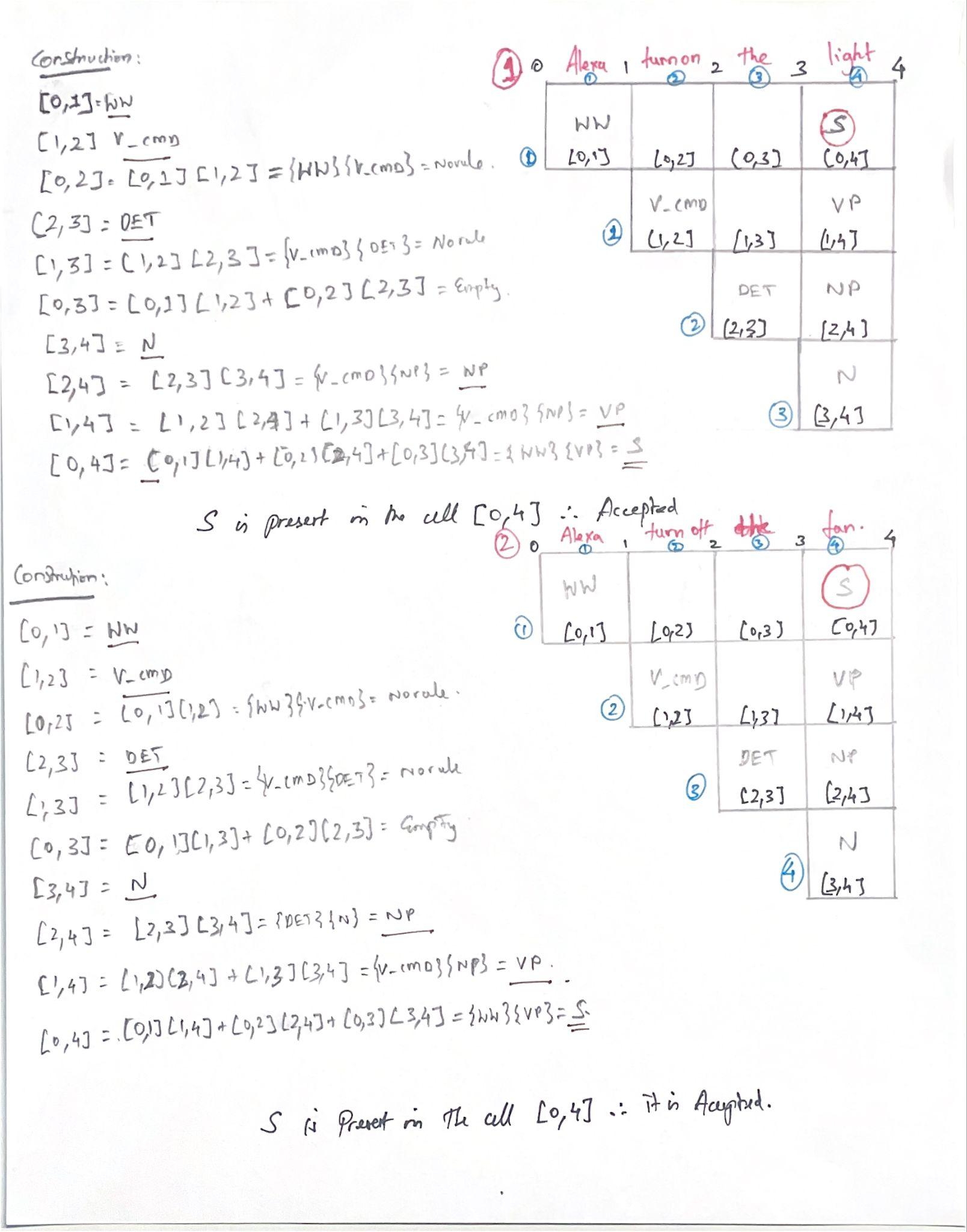


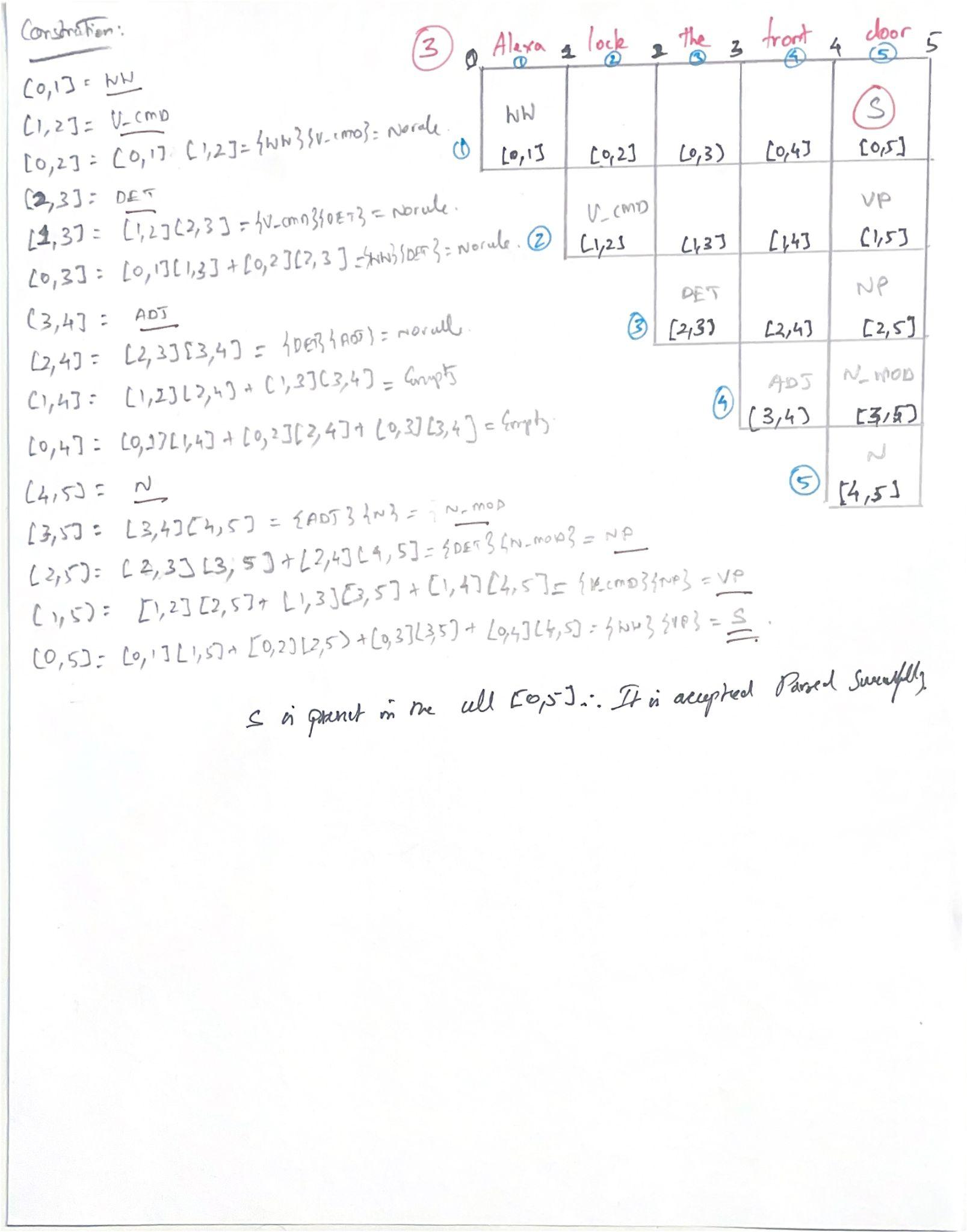
Showing the working and filling out the table

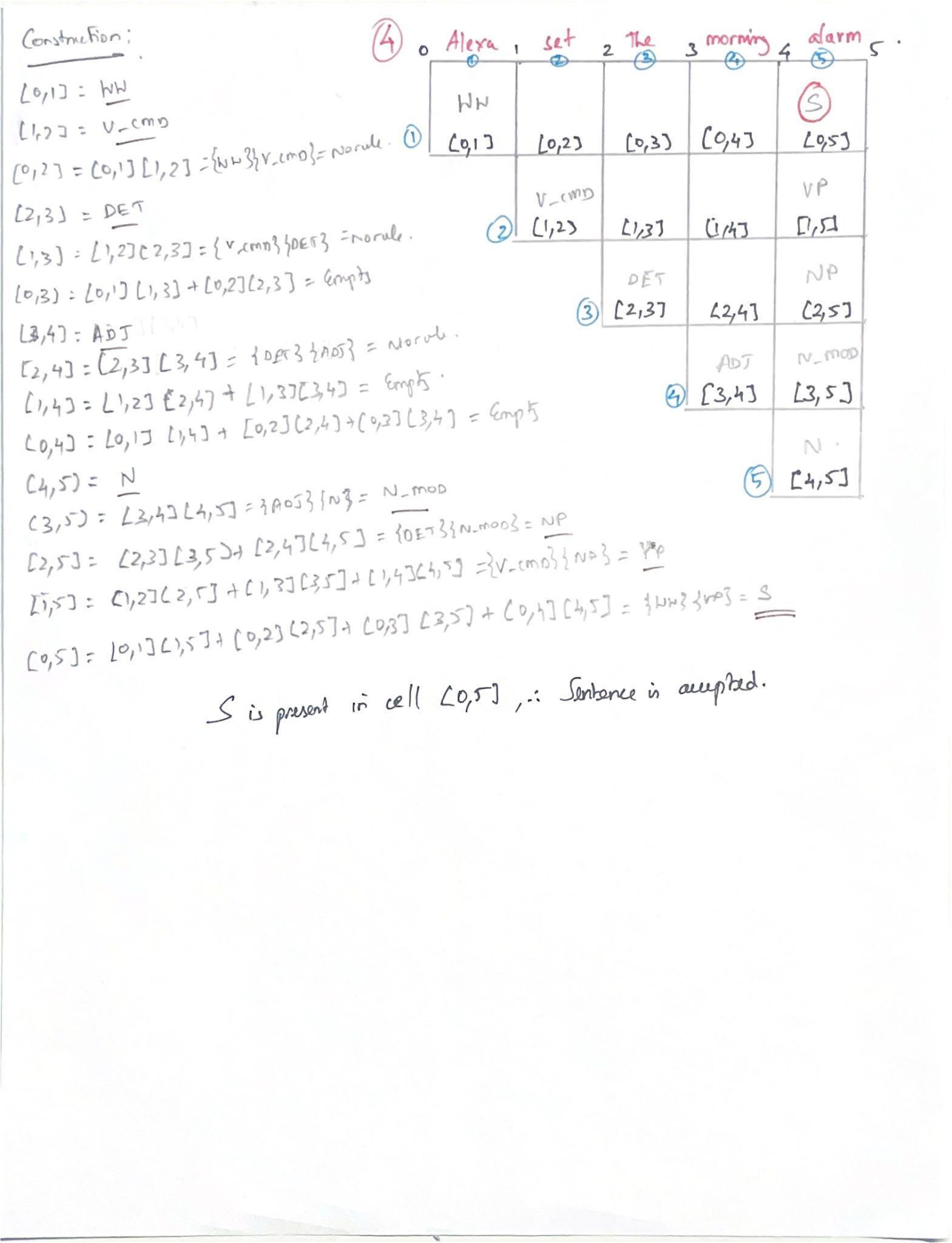
### Valid Sentence

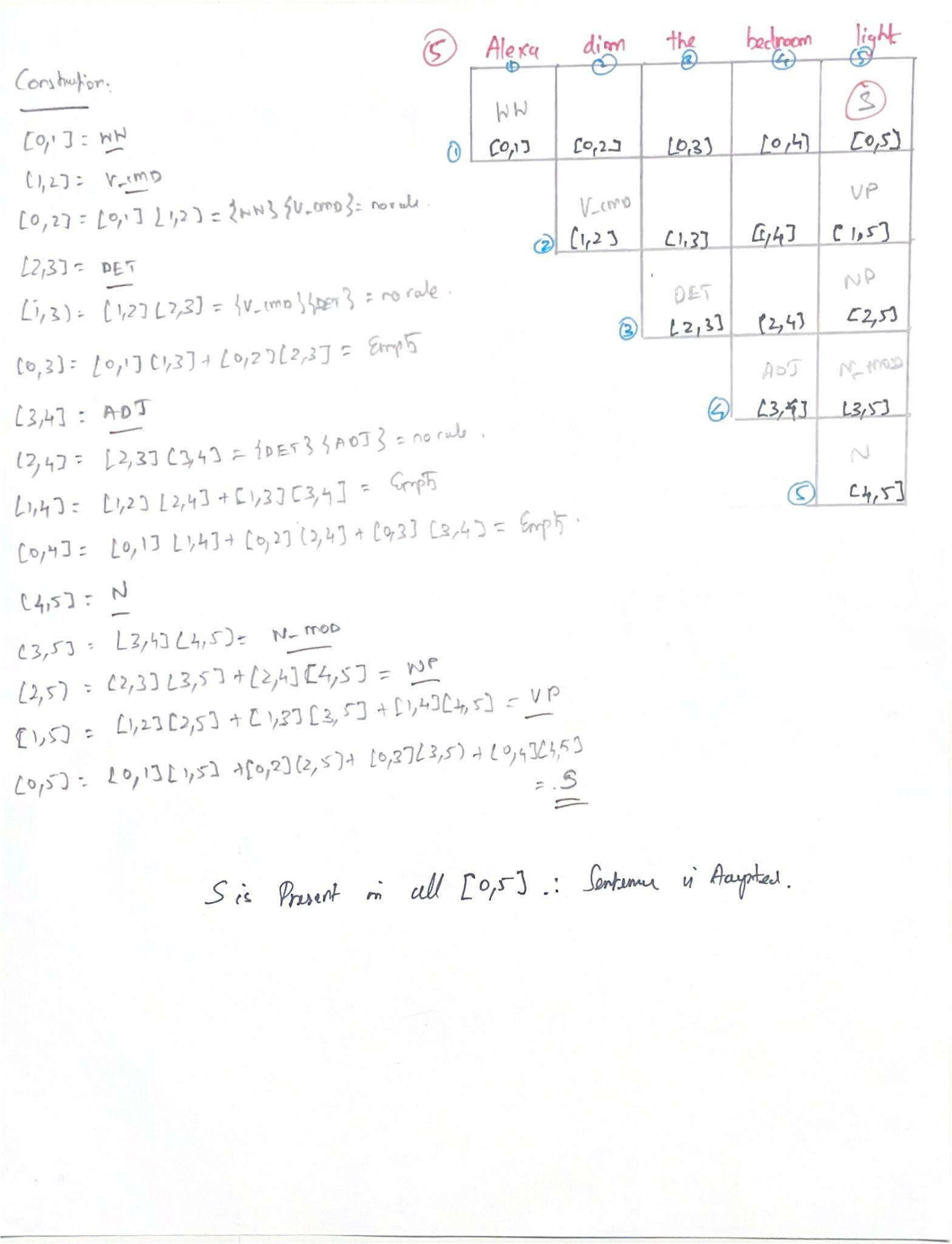
The analysis demonstrates the bottom-up nature of the parser, showing how it first identifies individual words, then systematically combines them into larger phrases (like Noun Phrases and Verb Phrases) according to our CNF rules. A successful parse is confirmed when the final cell, representing the entire sentence, contains the Start Symbol S.

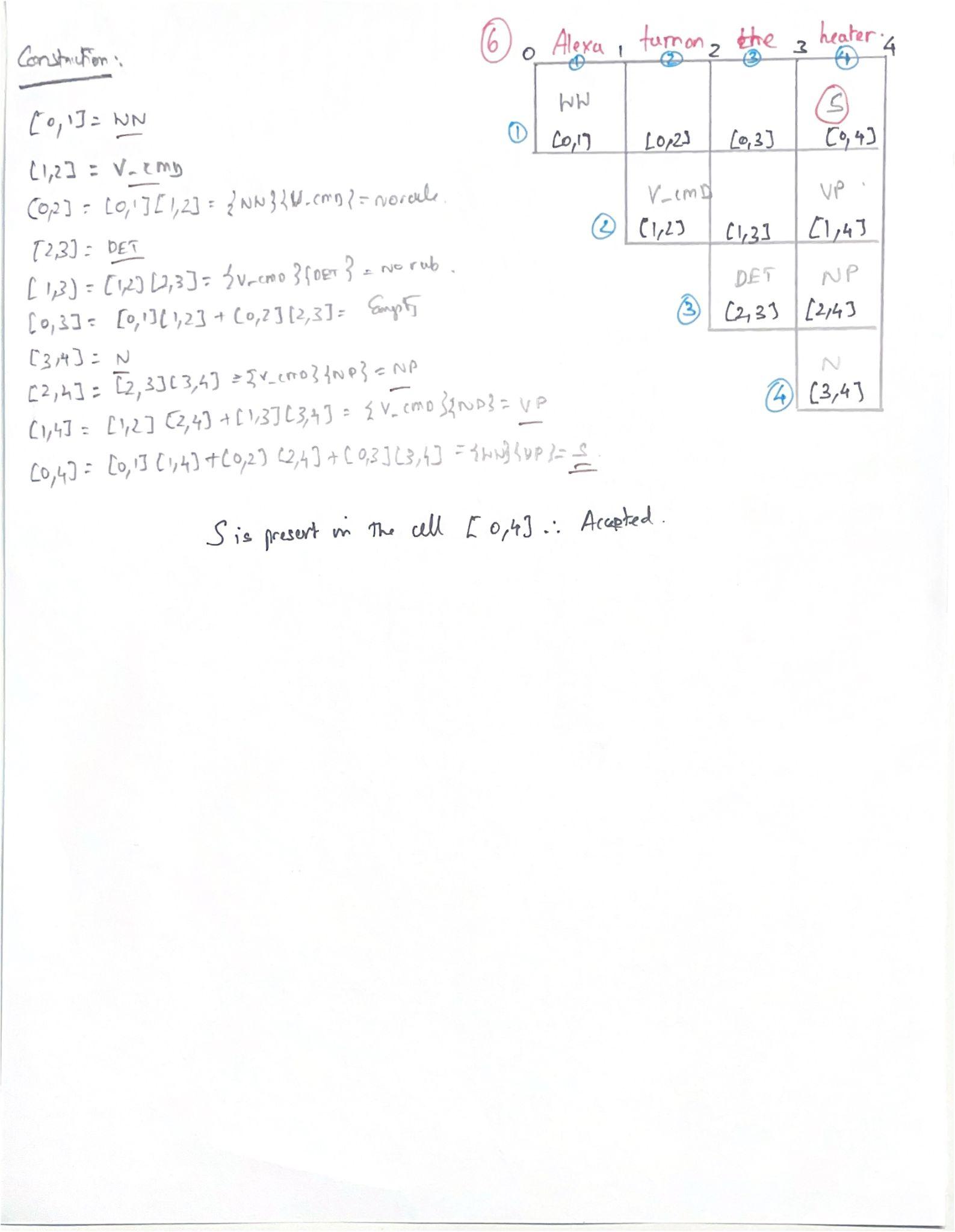


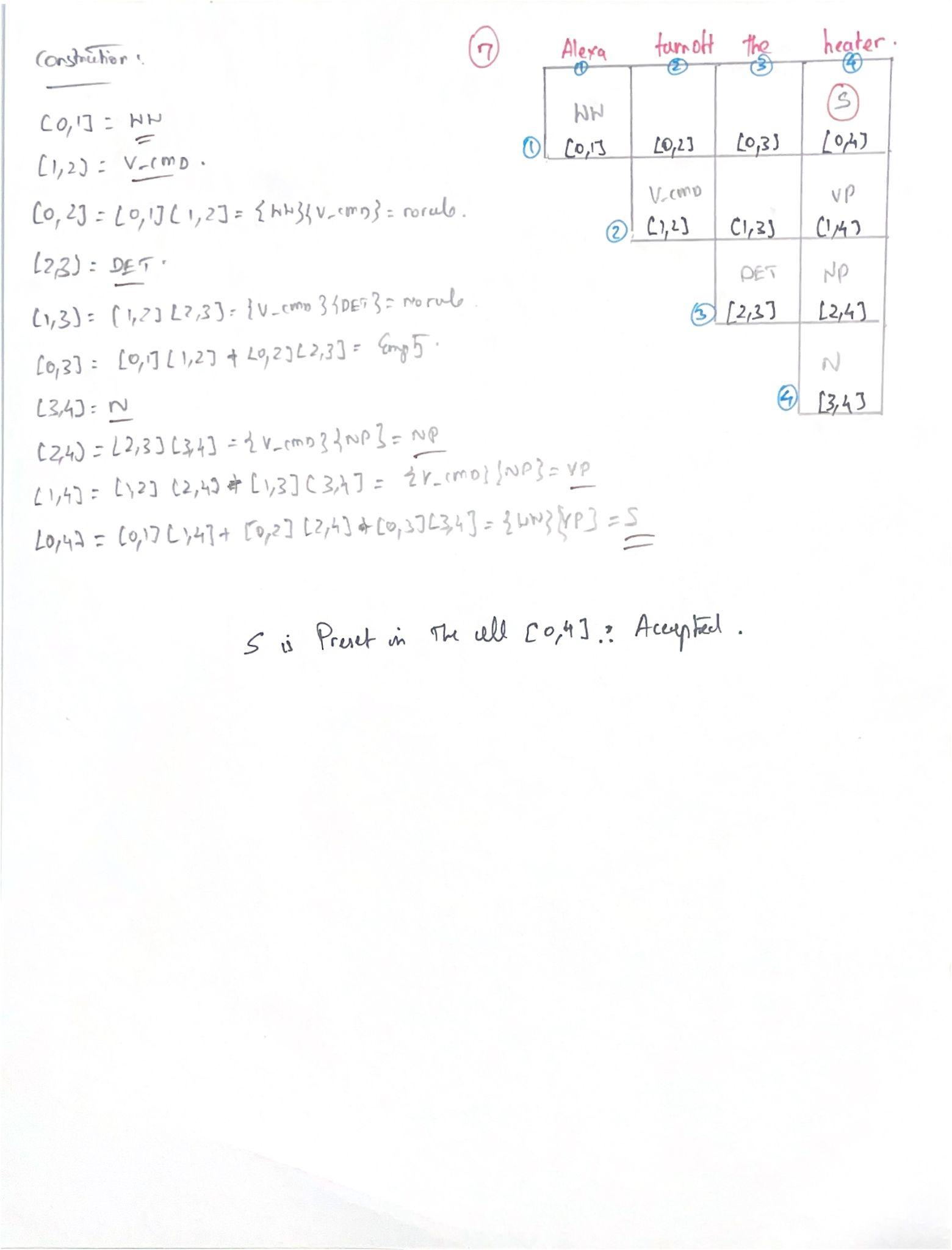


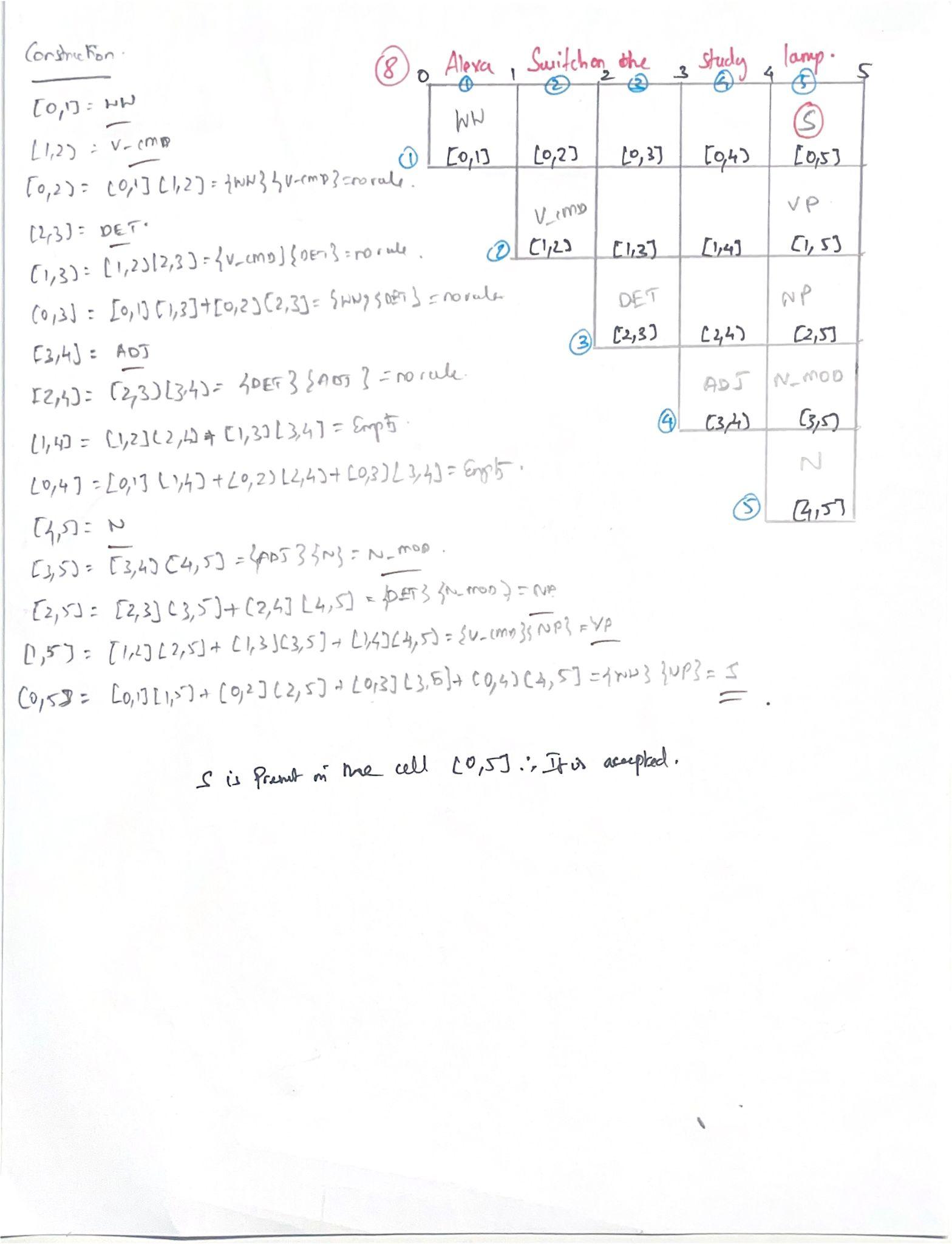


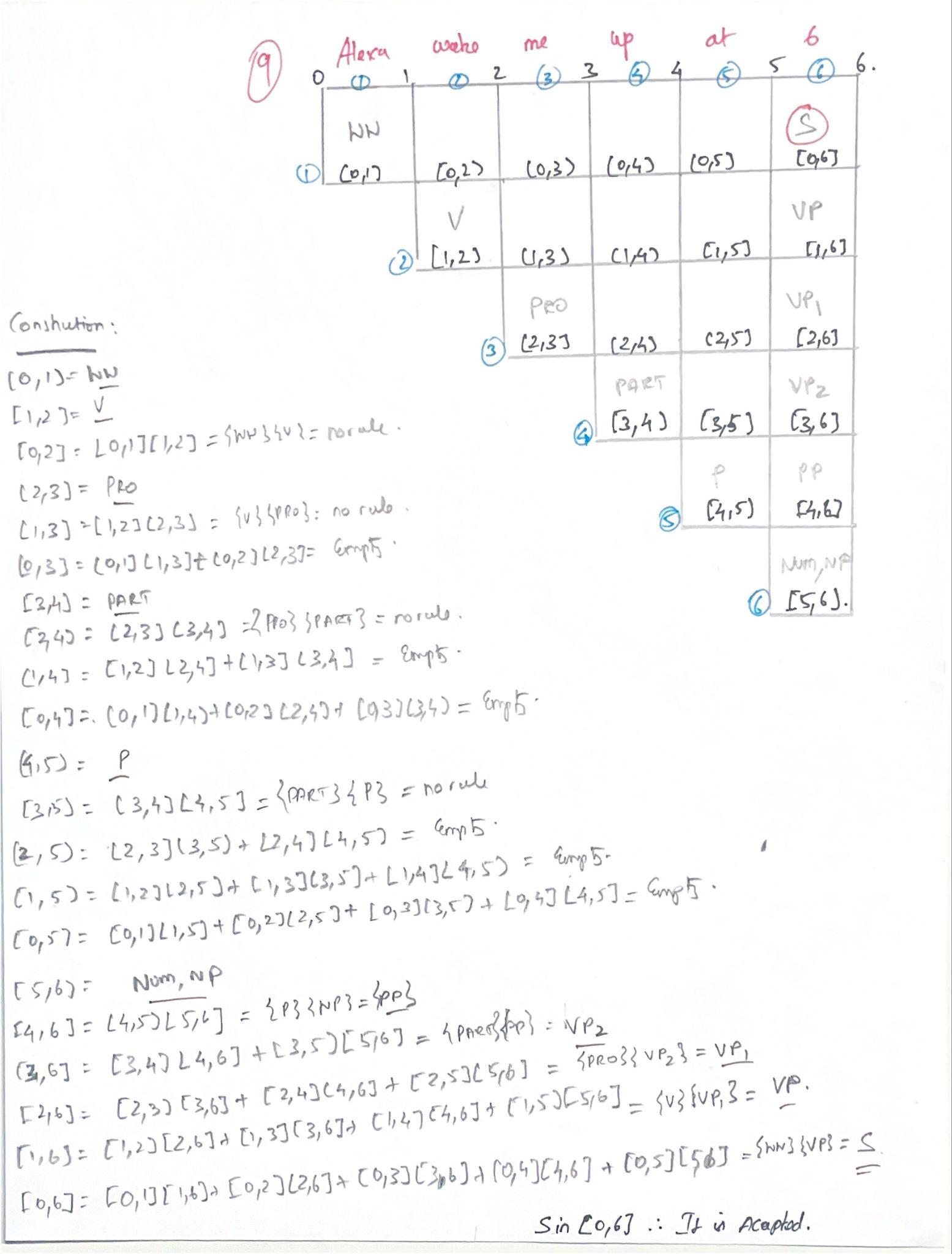


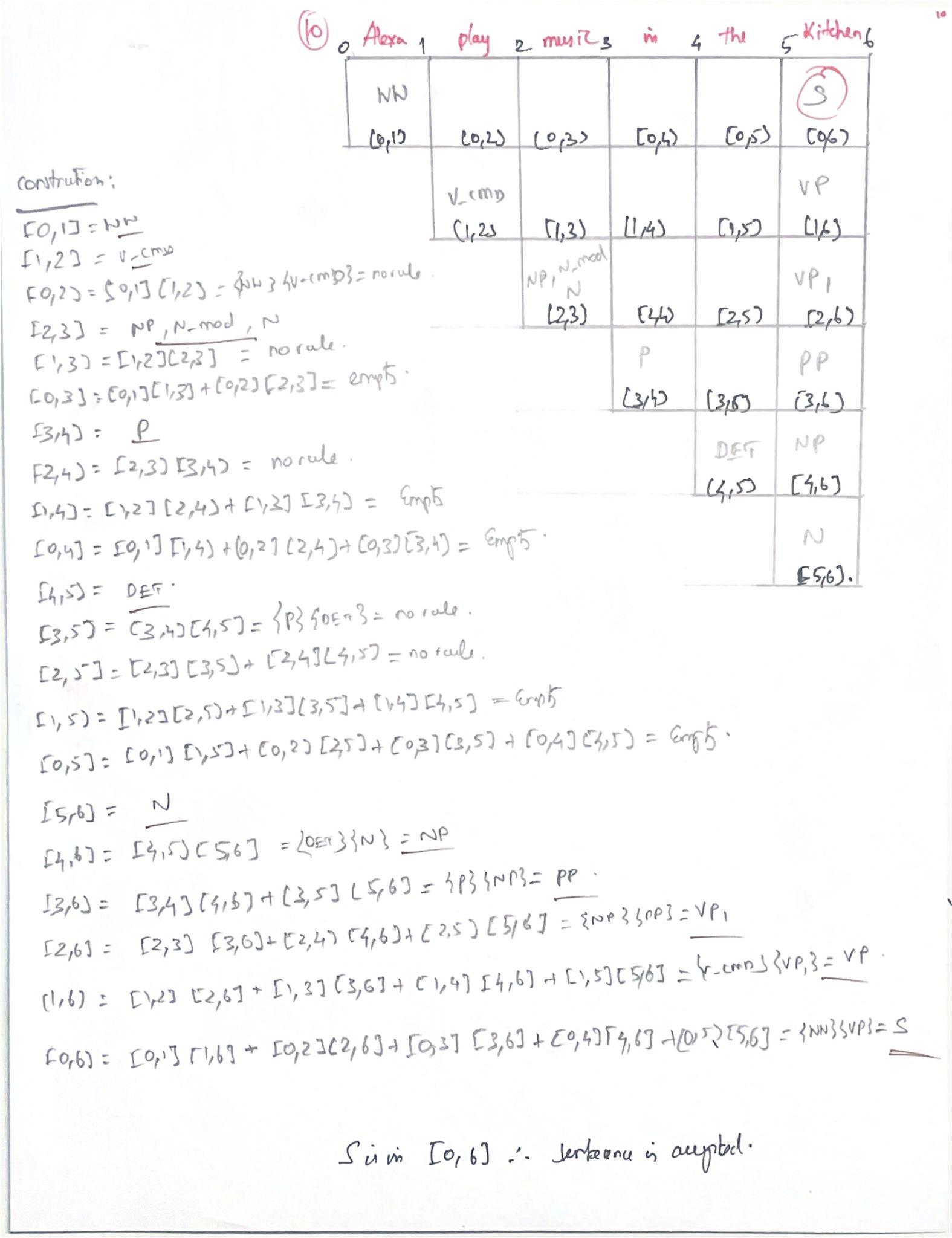


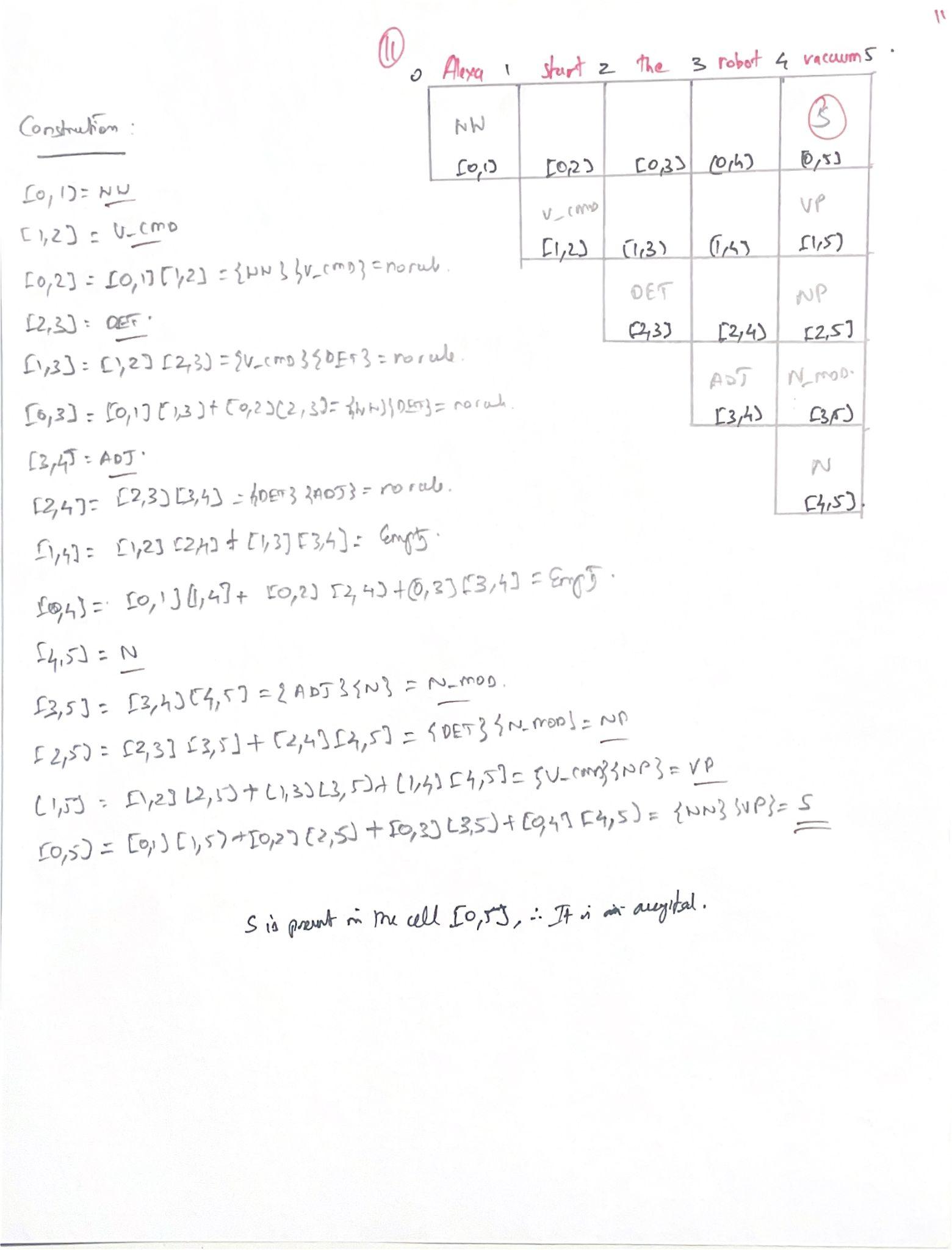


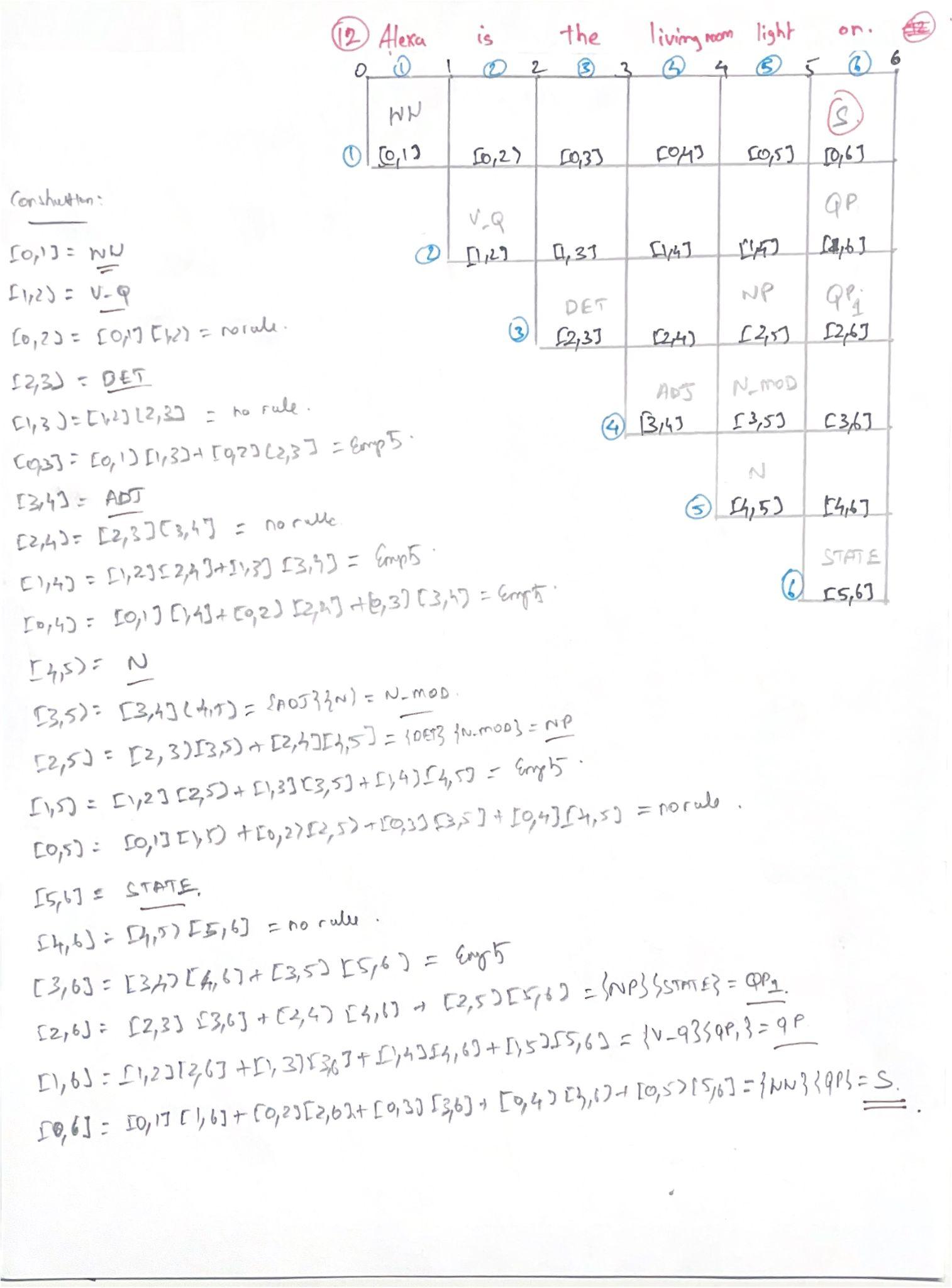


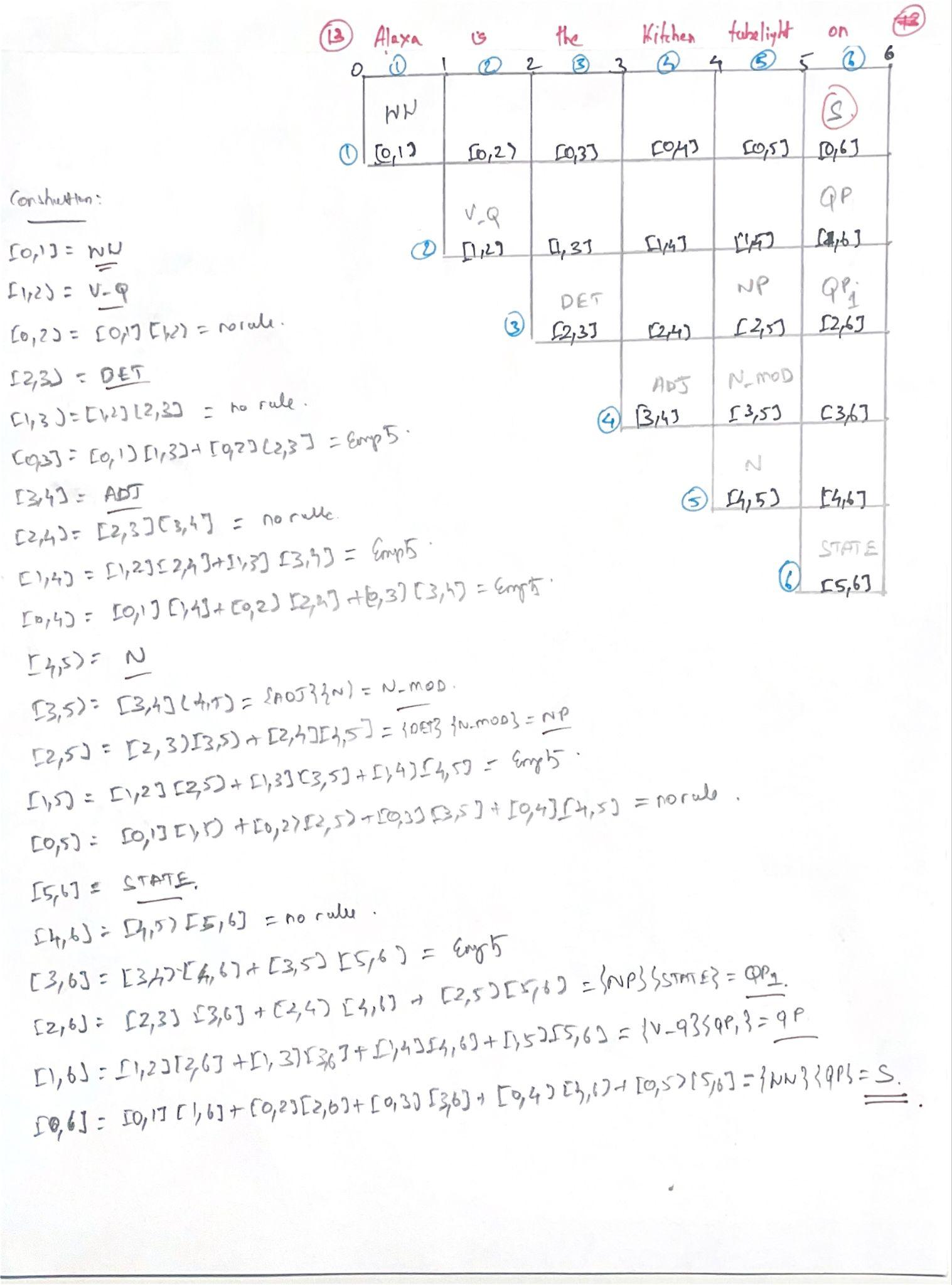


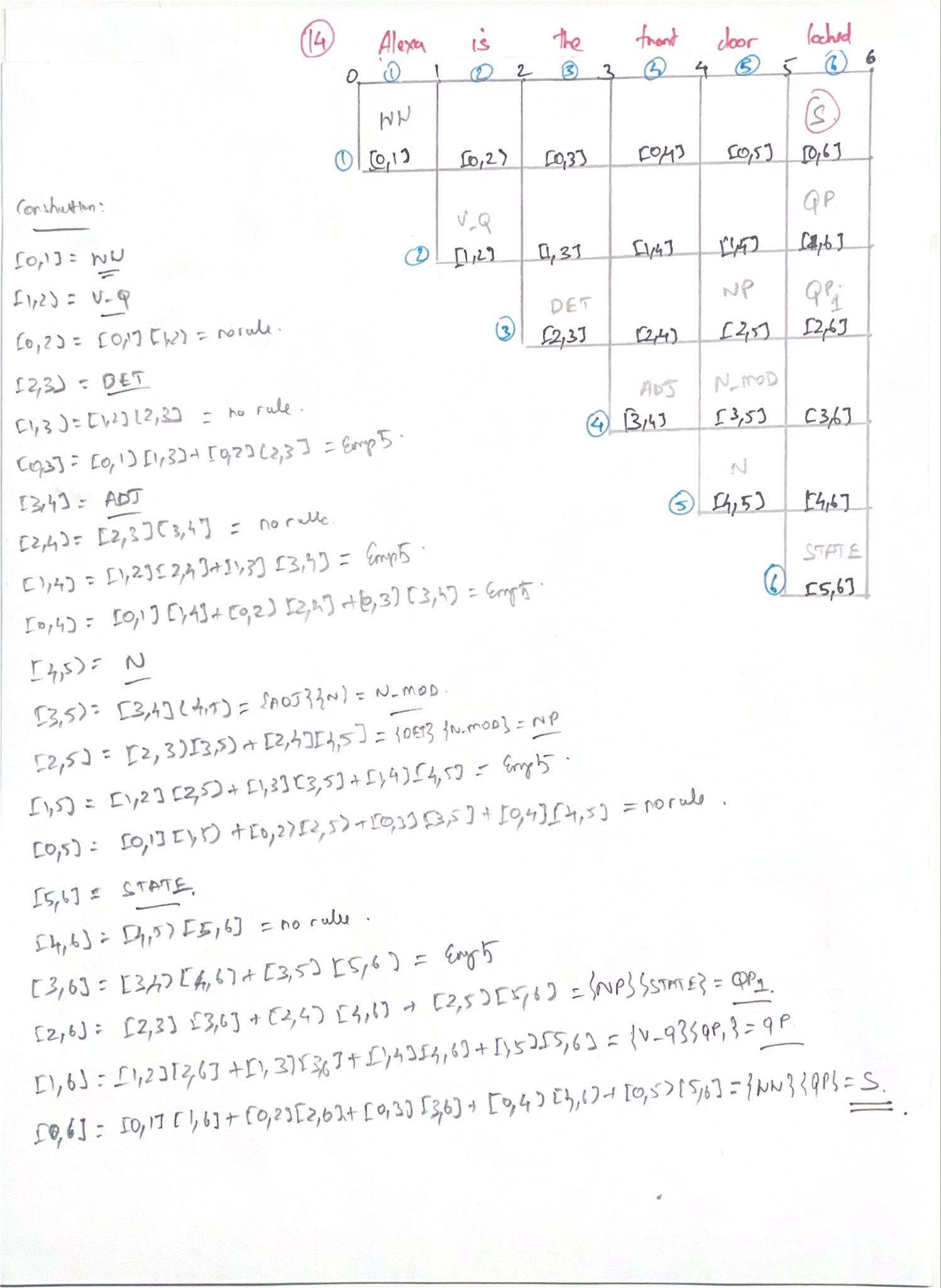


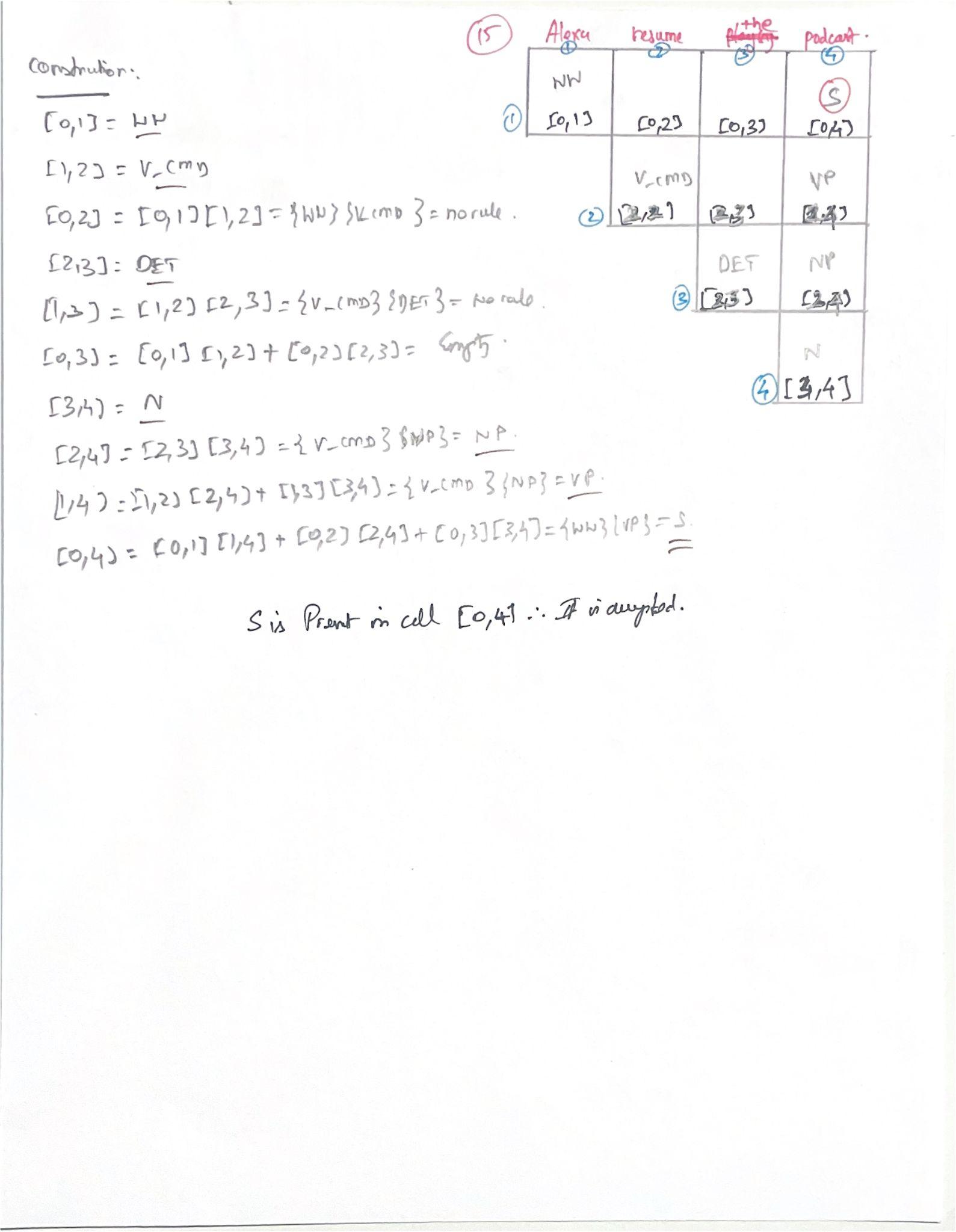












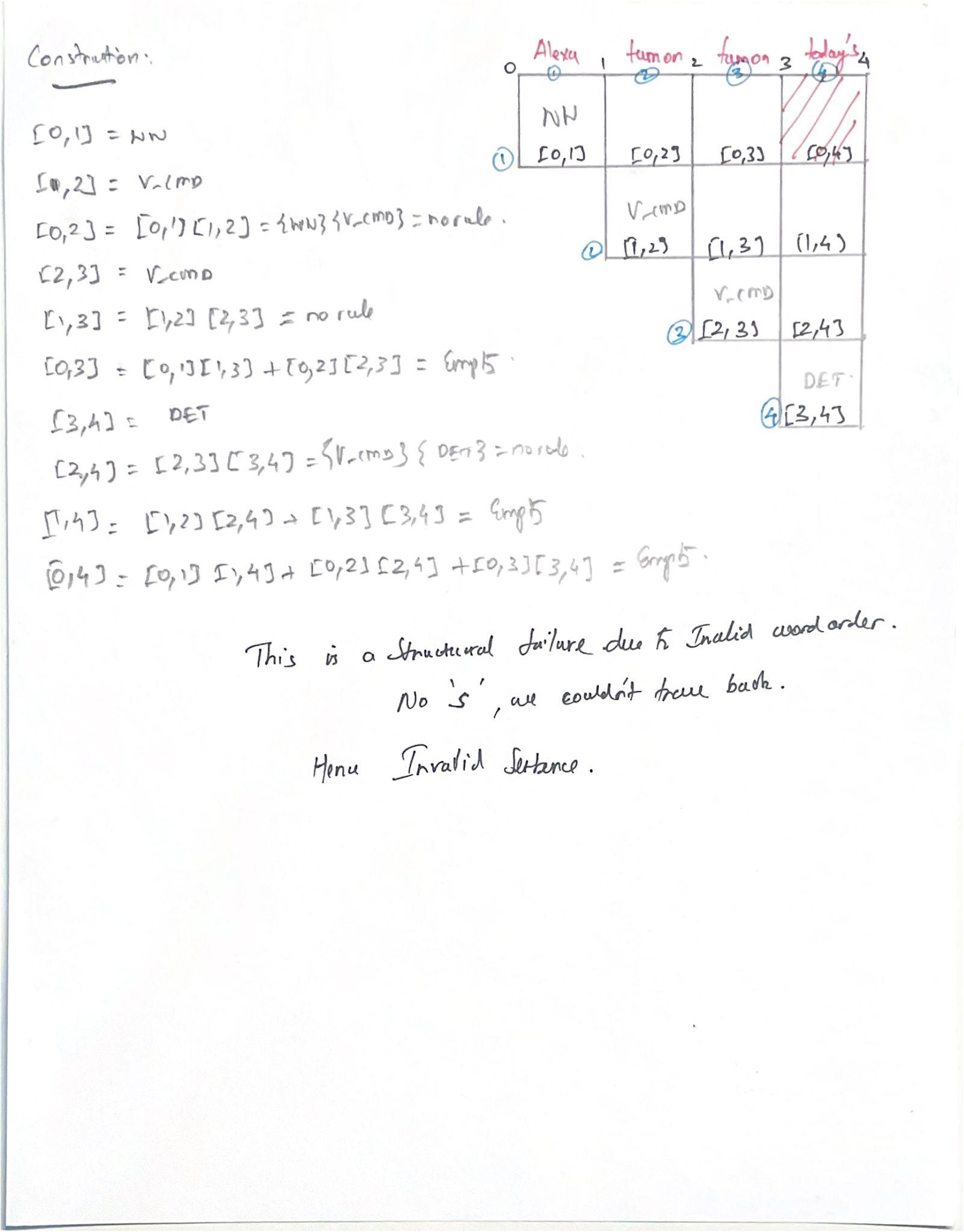
### Invalid Sentence

Demonstrating that the parser accepts valid sentences is only half the story. Proving its precision by showing what it correctly rejects is **equally important**. This validation confirms that the system is not just matching keywords; it is rigorously enforcing a deep grammatical structure. It proves that **not everything that starts with "Alexa" works.**

To showcase this, we will now analyze sentences that fail for reasons more complex than simply having a word that is **not part of the lexicon**. The following examples highlight **different types of rejection** from structural impossibilities to contextual violations ,each demonstrating a specific strength of the parser in identifying and discarding ungrammatical commands.

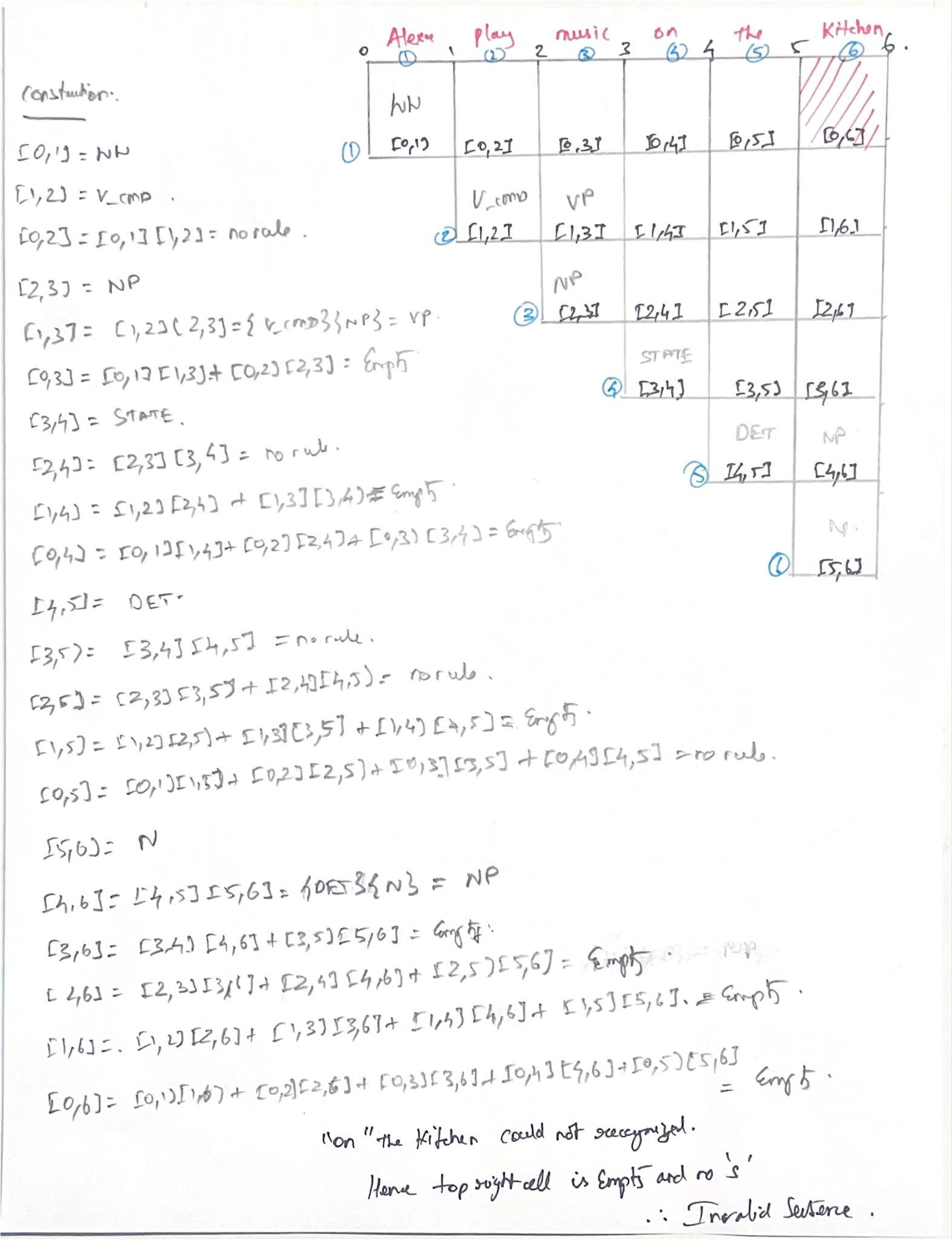
INVALID SENTENCE 1

Alexa turn on turn on today's (**Structural Failure-**Invalid sequence of phrases **)**



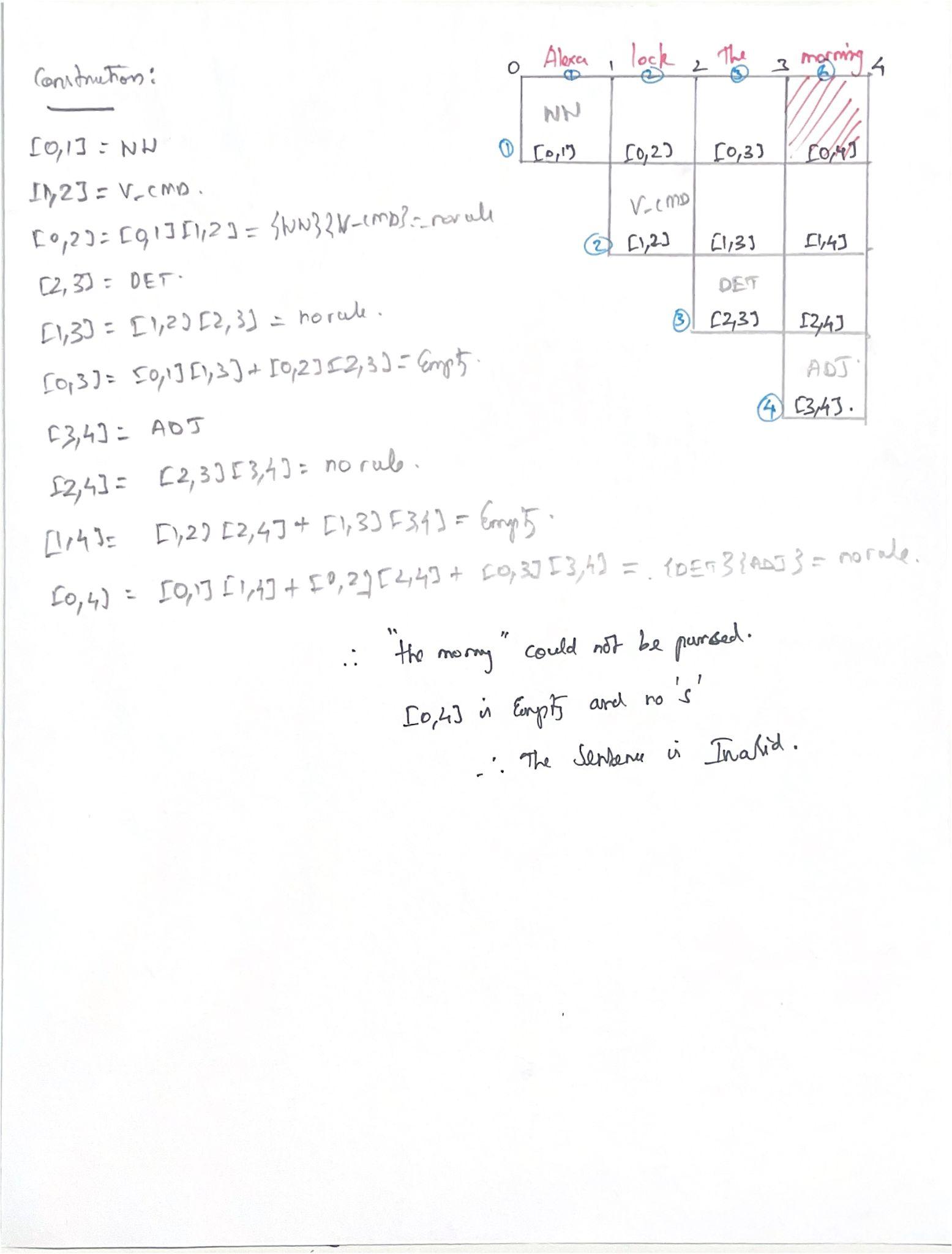
INVALID SENTENCE 2

Alexa play music on the kitchen (**Contextual Failure - “on”** wrong grammatical**)**



INVALID SENTENCE 3

Alexa lock the morning (**Malformed Phrase Failure- Bad phrase)**



## Conclusion

This project successfully demonstrated the complete pipeline for parsing a subset of natural language. By defining a specific domain, researching a realistic corpus, and applying formal language theory, a robust parser was developed. The process of creating a Context-Free Grammar, normalizing it to CNF, and validating it with the CKY algorithm proved to be a highly effective method for enforcing syntactic rules. The parser's ability to accept all 23 valid sentences while rejecting a variety of structurally and contextually flawed sentences confirms the correctness and precision of the grammar design.

## Resources

* [Talk to me: Exploring user interactions with the Amazon Alexa - Irene Lopatovska, Katrina Rink, Ian Knight, Kieran Raines, Kevin Cosenza, Harriet Williams, Perachya Sorsche, David Hirsch, Qi Li, Adrianna Martinez, 2019](https://journals.sagepub.com/doi/10.1177/0961000618759414)
* [User Interactions with “Alexa” in Public Academic Space](https://irenelopatovska.wordpress.com/wp-content/uploads/2018/08/asist18-alexa-in-public-02.pdf)
* [Best Practices for Sample Utterances and Custom Slot Type Values | Alexa Skills Kit](https://developer.amazon.com/en-US/docs/alexa/custom-skills/best-practices-for-sample-utterances-and-custom-slot-type-values.html)
* [“Alexa, You're Really Stupid”: A Longitudinal Field Study on Communication Breakdowns Between Family Members and a Voice Assistant](https://www.frontiersin.org/journals/computer-science/articles/10.3389/fcomp.2022.791704/full)
* [Decoding Alexa’s Personal Information Collection—A Guide](https://www.privacy.com/blog/alexa-personal-information)
* [All your Alexa recordings will go to the cloud soon, as Amazon sunsets Echo privacy | ZDNET](https://www.zdnet.com/home-and-office/smart-home/all-your-alexa-recordings-will-go-to-the-cloud-soon-as-amazon-sunsets-echo-privacy/)