



# Hidden Probabilistic One-Counter Automata

Thesis submitted in accordance with the requirements of the University of Liverpool for  
the degree of Doctor in Philosophy by

**Mehmet Kurucan**

July 2020



# Abstract

This thesis furthers the ongoing study of devising faster algorithms for the learning problem of infinite-state systems. We examine the Hidden Probabilistic One-Counter Machine (HP1CA) model, a natural extension of both Probabilistic One-Counter Automata (P1CA) and Hidden Markov Model (HMM) in this thesis. HP1CAs allow for the quadratic time complexity of learning the parameters of the model based on observational data. This gives them a big advantage over other infinite-state systems, e.g., probabilistic pushdown automata for which the same task requires cubic time.

An HP1CA is a P1CA where both the control states and the counter values are hidden, but the output sequence is not (note that there is no input sequence). Such a model has a non-negative counter whose value can change by at most one during a transition. The probability of a transition depends only on the current control state and whether the counter value is zero or not. The probability of a given output symbol depends on the current control state, but not on the value of the counter. A sequence of transitions is valid only if it starts and ends in a special initial control state and final control state, respectively, with the counter value equal zero.

After introducing the model, we adapt *Baum-Welch* algorithm, which is a special case of *Expectation-Maximization* algorithm used in unsupervised learning in HMM, to update the parameters of an HP1CA. Furthermore, we also adopted two dynamic programming algorithms which are called *Forward* algorithm and *Backward* algorithm, respectively to our new model. Our adapted algorithms have worst-case quadratic time complexity as compared to linear time for HMM. At the same time, its complexity is linear if the maximum value of the counter is assumed to be bounded by a constant (as it is commonly in practice).

After that, we look at other possible HP1CA models that differ depending on the semantic of what a valid transition sequence is. This can depend on the the initial and final terminal state condition as follows. First, an HP1CA can either start in (1) a fixed control state with counter zero or (2) have multiple initial control states, and some fixed probability distribution specifies the likelihood of starting in a given control state (with a counter value zero). Second, an HP1CA may be required to end with with (A) a unique

final control state with counter value zero, (B) a unique final control state with arbitrary counter value, (C) multiple final control states with zero counter value. This gives rise to six possible semantics, and only (1-A) semantic studied by us earlier. We adapt the learning and dynamic programming algorithms to the remaining five models.

In the end, we demonstrate that our HP1CA model lead to better accuracy and lower fit error of observation sequences of as compared to HMMs through an implementation.



# Acknowledgements

First and foremost I would like to express my deep gratitude to my supervisor, Dominik Wojtczak, who guided me throughout my doctoral years with his vast knowledge, and unearthly patience while giving me the freedom to work in my own way. He provided an invaluable source of knowledge to me when I was lost in the darkest and bring me to the brightest days.

I would like to thank my all friends in Liverpool especially Mete Ozbaltan who gave great support to the implementation of all models that are used in this thesis. I am lucky having found so many friends in Computer Science who also supported and encouraged me over the years. I would like to thank all those valuable people.

Furthermore, I would like to thank my family for their unconditional love and support. Especially my mother, Zarife Kurucan, and my father, Aziz Kurucan, were ready to do the work for me if they could when I got stuck. Without their prayers, I would not be the person I am today.

Last, but not least, I would like to thank my girlfriend Melike Nur Sonal, who will be my wife in the future, for her love and constant support, and for keeping me well over the past few months.



# Contents

|                                                                                                      |            |
|------------------------------------------------------------------------------------------------------|------------|
| <b>Abstract</b>                                                                                      | <b>i</b>   |
| <b>Acknowledgements</b>                                                                              | <b>iii</b> |
| <b>Contents</b>                                                                                      | <b>vii</b> |
| <b>List of Figures</b>                                                                               | <b>x</b>   |
| <b>1 Introduction</b>                                                                                | <b>1</b>   |
| 1.1 Research Hypothesis and contribution . . . . .                                                   | 4          |
| 1.2 Related Work . . . . .                                                                           | 4          |
| 1.2.1 Hidden Markov Models . . . . .                                                                 | 5          |
| 1.2.2 Pushdown Systems . . . . .                                                                     | 5          |
| 1.2.3 Counter Systems . . . . .                                                                      | 5          |
| 1.3 Outline of the thesis . . . . .                                                                  | 5          |
| <b>2 Preliminaries</b>                                                                               | <b>8</b>   |
| 2.1 General Notations . . . . .                                                                      | 8          |
| 2.2 Conditional Independence . . . . .                                                               | 9          |
| 2.3 Markov Property . . . . .                                                                        | 12         |
| 2.4 Automata Formalisms . . . . .                                                                    | 15         |
| 2.5 Conclusion . . . . .                                                                             | 17         |
| <b>3 Probabilistic State Space and Infinite State Models</b>                                         | <b>18</b>  |
| 3.1 State Space Models . . . . .                                                                     | 19         |
| 3.1.1 Definition of Probabilistic Finite State Automata (PFSM) and Equiv-<br>alence to HMM . . . . . | 20         |
| 3.1.2 Definition of HMM . . . . .                                                                    | 25         |
| 3.1.3 Fundamental Algorithms of HMM . . . . .                                                        | 29         |
| 3.2 Infinite State Models . . . . .                                                                  | 39         |
| 3.2.1 Stack Based Systems . . . . .                                                                  | 41         |

|          |                                                                                                                   |            |
|----------|-------------------------------------------------------------------------------------------------------------------|------------|
| 3.2.2    | Counter Systems . . . . .                                                                                         | 45         |
| 3.2.3    | Definition of P1CA . . . . .                                                                                      | 49         |
| 3.3      | Conclusion . . . . .                                                                                              | 53         |
| <b>4</b> | <b>Model-1: One initial terminal and one final terminal in a HP1CA</b>                                            | <b>56</b>  |
| 4.1      | Formal Definition of HP1CA . . . . .                                                                              | 57         |
| 4.1.1    | Configuration of HP1CA . . . . .                                                                                  | 58         |
| 4.1.2    | Transition relation of HP1CA . . . . .                                                                            | 60         |
| 4.2      | Adaptation of Algorithms . . . . .                                                                                | 64         |
| 4.2.1    | Adapted Forward Algorithm . . . . .                                                                               | 65         |
| 4.2.2    | Adapted Backward Algorithm . . . . .                                                                              | 79         |
| 4.2.3    | Adapted Baum-Welch Algorithm . . . . .                                                                            | 88         |
| 4.3      | Conclusion . . . . .                                                                                              | 103        |
| <b>5</b> | <b>Characteristic Feature HP1CA Models</b>                                                                        | <b>107</b> |
| 5.1      | Model-2: Multiple initial terminals and unique final terminal . . . . .                                           | 108        |
| 5.1.1    | Applied an initial vector to adapted forward algorithm . . . . .                                                  | 109        |
| 5.1.2    | Applied initial vector to adapted backward algorithm . . . . .                                                    | 114        |
| 5.1.3    | Applied initial vector to adapted Baum-Welch algorithm . . . . .                                                  | 118        |
| 5.2      | Model-3: Multiple final terminals and unique initial terminal . . . . .                                           | 123        |
| 5.2.1    | Applying adapted <i>forward</i> algorithm to Model-3 . . . . .                                                    | 124        |
| 5.2.2    | Applying adapted <i>backward</i> algorithm to Model-3 . . . . .                                                   | 127        |
| 5.2.3    | Applying adapted <i>Baum-Welch</i> algorithm to Model-3 . . . . .                                                 | 130        |
| 5.3      | Model-4: Initial vector and multiple final terminals . . . . .                                                    | 132        |
| 5.3.1    | Applying adapted <i>forward</i> algorithm to Model-4 . . . . .                                                    | 133        |
| 5.3.2    | Applying adapted <i>backward</i> algorithm to Model-4 . . . . .                                                   | 137        |
| 5.3.3    | Applying adapted <i>Baum-Welch</i> algorithm to Model-4 . . . . .                                                 | 140        |
| 5.4      | Model-5: One initial terminal with zero counter and one particular final<br>state with non-zero counter . . . . . | 141        |
| 5.4.1    | Applying adapted <i>forward</i> algorithm to Model-5 . . . . .                                                    | 143        |
| 5.4.2    | Applying adapted <i>backward</i> algorithm to Model-5 . . . . .                                                   | 145        |
| 5.4.3    | Applying adapted <i>Baum-Welch</i> algorithm to Model-5 . . . . .                                                 | 148        |
| 5.5      | Model-6: Initial vector and one particular final state with non-zero counter                                      | 149        |
| 5.5.1    | Applying adapted <i>forward</i> algorithm to Model-6 . . . . .                                                    | 150        |
| 5.5.2    | Applying adapted <i>backward</i> algorithm to Model-6 . . . . .                                                   | 153        |
| 5.5.3    | Applying adapted <i>Baum-Welch</i> algorithm to Model-5 . . . . .                                                 | 155        |
| 5.6      | Conclusion . . . . .                                                                                              | 157        |
| <b>6</b> | <b>Experiments</b>                                                                                                | <b>159</b> |
| 6.1      | Implementation and results of Model-1 . . . . .                                                                   | 159        |
| 6.2      | Implementation and results of other HP1CA models . . . . .                                                        | 165        |

|                     |            |
|---------------------|------------|
| <b>7 Conclusion</b> | <b>169</b> |
| <b>References</b>   | <b>172</b> |



# List of Figures

|      |                                                                                                                                                                                                                                                                                                                                            |    |
|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 2.1  | A DAG diagram that represent the joint distribution $Pr(S_1, S_2, S_3)$ of three variables of the network where it is convenient to be formed by application of the product rule as $Pr(S_1, S_2, S_3) = Pr(S_3 S_1, S_2)Pr(S_2 S_1)Pr(S_1)$ . Here the link goes from $S_1$ to $S_2$ then $S_1$ is the parent of the node $S_2$ . . . . . | 10 |
| 2.2  | Three illustrates of the concepts of the <i>d-separation</i> . . . . .                                                                                                                                                                                                                                                                     | 11 |
| 2.3  | An example Markov chain process. Each edge depicts the probability of taking that transition from its origin state. There are three states in the set of state-space where $S = \{Sunny, Rainy, Cloudy\}$ . . . . .                                                                                                                        | 14 |
| 2.4  | An example for deterministic and non-deterministic finite automata . . . . .                                                                                                                                                                                                                                                               | 15 |
| 2.5  | Input tape with reading head . . . . .                                                                                                                                                                                                                                                                                                     | 16 |
| 3.1  | An example of state-space based on Markovian property . . . . .                                                                                                                                                                                                                                                                            | 19 |
| 3.2  | An example for the graphical representation of <i>PFSM</i> . . . . .                                                                                                                                                                                                                                                                       | 21 |
| 3.3  | An example for HMM structure . . . . .                                                                                                                                                                                                                                                                                                     | 26 |
| 3.4  | The observed output sequence that has been produced by the most likely sequence of hidden states . . . . .                                                                                                                                                                                                                                 | 28 |
| 3.5  | The forward trellis for computing the total probability of the observation by summing all the previous forward values multiplied by transition probabilities and emission probability. . . . .                                                                                                                                             | 30 |
| 3.6  | The induction step of the <i>backward</i> procedure that proceeds by summing all the future $\beta_{t+1}(S_i)$ values with weighted state transition probability and future observation probability. . . . .                                                                                                                               | 32 |
| 3.7  | The Viterbi computation and the backtrace keeps the most probable path that leds to the state. . . . .                                                                                                                                                                                                                                     | 34 |
| 3.8  | The calculation of the joint probability of being in state $S_i$ at time $t$ and state $S_j$ at time $t + 1$ . . . . .                                                                                                                                                                                                                     | 36 |
| 3.9  | The calculation of the probability of being in state $S_j$ at time $t$ . . . . .                                                                                                                                                                                                                                                           | 37 |
| 3.10 | The structure of the stack-based machine . . . . .                                                                                                                                                                                                                                                                                         | 42 |
| 3.11 | Transition diagram of a <i>PDA</i> . . . . .                                                                                                                                                                                                                                                                                               | 43 |
| 3.12 | The example of the word <i>aabb</i> is accepted by a <i>PDA</i> . . . . .                                                                                                                                                                                                                                                                  | 45 |

|      |                                                                                                                                                                                 |     |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 3.13 | The structure of the stack-based machine . . . . .                                                                                                                              | 46  |
| 3.14 | Transition diagram of a <i>OCA</i> . . . . .                                                                                                                                    | 47  |
| 3.15 | Transition diagram and set of transitions of a <i>P1CA</i> . . . . .                                                                                                            | 51  |
| 4.1  | A simple example of <i>Hp1CA</i> . . . . .                                                                                                                                      | 59  |
| 4.2  | The output sequence that has been produced by the most likely sequence of hidden states and counter pair . . . . .                                                              | 61  |
| 4.3  | The parameters of a <i>HP1CA</i> for a simple example . . . . .                                                                                                                 | 67  |
| 4.4  | The trellis of the preliminary forward flow . . . . .                                                                                                                           | 68  |
| 4.5  | Normalisation calculation of adapted forward algorithm . . . . .                                                                                                                | 73  |
| 4.6  | The trellis diagram of preliminary calculation of adapted backward algorithm . . . . .                                                                                          | 81  |
| 4.7  | Normalisation calculation of adapted backward algorithm . . . . .                                                                                                               | 84  |
| 4.8  | Gamma calculation of being in state-counter pair . . . . .                                                                                                                      | 93  |
| 4.9  | $\xi$ calculation for all hidden states given the observation sequence $(a, b)$ . . . . .                                                                                       | 96  |
| 5.1  | Two possible initial point for each designated paths for starting the distribution to reach the final state $(S_T, 0)$ . . . . .                                                | 109 |
| 5.2  | Two initial states which $(q_1, 0)$ and $(q_2, 0)$ yield multiple starting points for computation of state distribution. . . . .                                                | 110 |
| 5.3  | Two initial states which $(q_1, 0)$ and $(q_2, 0)$ yield multiple starting points for computation of state distribution. . . . .                                                | 113 |
| 5.4  | The calculation of preliminary backward flow at $t = 0$ . . . . .                                                                                                               | 116 |
| 5.5  | Two possible initial point for each designated paths for starting the distribution to reach the final state . . . . .                                                           | 123 |
| 5.6  | Two final states which $(q_1, 0)$ and $(q_2, 0)$ yield multiple ending points for the computation of state distribution. . . . .                                                | 124 |
| 5.7  | Two final states which $(q_1, 0)$ and $(q_2, 0)$ yield multiple starting points for computation of forward flow normalization. . . . .                                          | 125 |
| 5.8  | The calculation of preliminary backward flow at $t = T - 1$ . . . . .                                                                                                           | 128 |
| 5.9  | Model-4 provides multiple initial points and multiple terminate points for all designated paths that are valid throughout the length of the given observation sequence. . . . . | 133 |
| 5.10 | Multiple terminals at both side of initial and final that yield multiple starting and terminating points for computation of the normalization of forward flow. . . . .          | 135 |
| 5.11 | Multiple final terminals provides multiple starting points for computation of the both preliminary and normalization of <i>backward</i> flow. . . . .                           | 138 |
| 5.12 | The counter value of the final terminal $q_F$ might be non-zero instead zero value as in the previous models. . . . .                                                           | 142 |
| 5.13 | Three different type of Model-5 which pretending to generate the $T$ length output sequence . . . . .                                                                           | 142 |

|      |                                                                                                                                                                                                           |     |
|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 5.14 | Multiple counter values that yields multiple final points where each one of them possibly being a final terminal to the model for computation of the normalization of <i>forward</i> calculation. . . . . | 144 |
| 5.15 | The value of the normalization of <i>backward</i> flow is one for if the particular hidden variable is equal to final terminal with non-zero counter value. . . .                                         | 147 |
| 5.16 | There are multiple initial terminals and one unique final terminal with non-zero counter value. . . . .                                                                                                   | 150 |
| 5.17 | Multiple initial terminal and multiple counter values that yield multiple starting and terminated points to the model for computation of the normalization of <i>forward</i> calculation. . . . .         | 151 |
| 5.18 | Applying initial vector where the probability distribution of multiple initial terminals are hold in, to Model-6. . . . .                                                                                 | 154 |
| 6.1  | The number of iterations for Model-1 . . . . .                                                                                                                                                            | 160 |
| 6.2  | The time scales of both models of <i>HP1CA</i> and <i>HMM</i> over the same datasets                                                                                                                      | 161 |
| 6.3  | The time scales of both models of <i>HP1CA</i> and <i>HMM</i> over the same datasets                                                                                                                      | 163 |
| 6.4  | The time scales of both models of <i>HP1CA</i> and <i>HMM</i> over the same datasets                                                                                                                      | 164 |
| 6.5  | The computation times of five model types of <i>HP1CA</i> and <i>HMM</i> . . . . .                                                                                                                        | 166 |
| 6.6  | The accuracy of five model types of <i>HP1CA</i> and <i>HMM</i> . . . . .                                                                                                                                 | 166 |
| 6.7  | The accuracy of five model types of <i>HP1CA</i> and <i>HMM</i> . . . . .                                                                                                                                 | 167 |

# Chapter 1

## Introduction

Systems that exhibit both probabilistic and controlled behavior are common in many domains. Models of such systems help to understand them before they are built. It is useful to make a distinction between finite and infinite-state models. In the finite state models, the behavior of objects is examined by processing through the state sequence defined by transitional logic and condition. For instance, Finite State Automata (or Machine) (FSA) (or Probabilistic Finite State Automata (PFSA), which is a version of FSA where a fixed probability is associated with each transition) is simply finite-state model where it can change from one state to another in response a transition. In the infinite state models, the behavior of FSA is extended by adding an unbounded memory (or stack) for exhibiting more complex behavior. Pushdown Automata (PDA) is a well-known stack-based system that is an instance of an infinite-state model.

Automata theory is the basis for the theory of formal languages. The word "Automata" (or it is also called *machine*) might be defined as a device that accepts the formal languages. They are important tools to understand languages better. A language can be seen as a system that expresses the behavior of natural ideas, which consists of a set of symbols and rules to process them. Formalizing the concept of a language, these variables must be used. A common feature is that a language is built up by a collection of sentences; a sentence consists of a sequence of words and a word is constructed via a combination of syllables. Therefore, a word is a sequential data that is generated by symbols of its underlying alphabet.

The natural concepts of the formal languages are the rewriting rules of grammars. A grammar is a powerful tool that consists of a set of rules for describing and analyzing languages. It generates a wide variety of complex languages from all the valid strings of a language. Regular grammar (RG) and context-free grammar (CFG) are well-known formal grammars that provide various formation rules to be used in the formal language fields. The basis of a recognizer that determines for any given string whether it is grammatical (i.e. belongs to the language) is described as automata, as separate formalism, in the formal language theory.

The languages that are generated by a regular grammar can be accepted by an FSA in linear time. FSA is a very simple model of a computer. The construction of this machine includes two main parts which are input tape from where it reads a string of symbols and finite control states that is been switched from state to state based on what symbol is read by the machine at each time step. The acceptance of the given string is only determined by the termination of the machine in a particular state of the set of finite states (so-called final state). If the machine ends up in any other state, then the string is not accepted. Two kinds of FSA are used in processing of regular languages. One of them is called Deterministic Finite State Automata (DFSA) where this machine follows at most one unique trace for any given input tape. The other is called Non-deterministic Finite State Automata (NFSA) which has many alternative traces for a given input. Another extension of the concept of a FSA is called Probabilistic Finite State Automata (PFSA) that in way lies between DFSA and NFSA [113]. The transitions of a PFSA is associated with probabilities. This concept endow the machine with a random process behavior which generates a random path for any given string. In practice, the generalization of the most common type of PFSA is called a Hidden Markov Model (HMM) in which the outputs, as well as the transitions, are governed by a random process.

String patterns that are generated by using a set of recursive product rules can be done by Context-Free Grammar (CFG). This is another kind of grammar that is used in formal languages. Any of the production rules in the grammar can be applied regardless of context. That means the language is generated by repeated the rule applications and their sentences are not influenced by the context. The language that is generated by CFG can be characterized as the language accepted by a class of machines called Pushdown Automata (PDA). A pushdown automaton is an extension of the FSA, which consists of

finite automaton and input tape, by allowing it to use a restricted stack. The input tape can be read one symbol at a time by a head that can move on the input tape from left to right only. At any instant, in time the input head is placed on a particular cell of the input tape and reads the symbol written on that cell. This machine has memory management capability that stores an unbounded amount of information on the stack. But access to the information on the memory is limited. Probabilistic variants of this model, which is called Probabilistic Pushdown Automata (PPDA), is obtained by associating probabilities to transition rules. The total probability of all rules applicable to a given configuration is one. The formalism of this stochastic machine defines the same language class that is generated via Stochastic Context-Free Grammar (SCFG) [1]. The distribution of SCFG on the string is a natural extension of CFG to the stochastic setting. Therefore, it can be said that an SCFG is a CFG where the rules (productions) are associated with probabilities.

A probabilistic one counter automaton (P1CA) is just a PPDA with only one stack symbol and a special bottom symbol. This counter machine is typically defined as having a finite number of control states and an additional non-negative counter which can be incremented or decremented during transitions. The transition functions that abstract the behavior of the machine can be enabled or disabled depending on whether the counter is equal to 0 or not. This can easily be seen to be equivalent to a PPDA with one stack symbol. The stack acts precisely as a non-negative counter, and the counter is equal to 0 precisely when the bottom symbol is the only symbol remaining on the stack. A P1CA can recognize the stochastic one counter languages (OCL) which are a subclass of stochastic context-free languages (SCFL) which are generated by SCFG, and those languages are a superset of regular languages which are generated by regular grammars [93].

A SCFL and its subclasses can be recognized in cubic time by an algorithm such as CYK algorithm [24]. In other words, there exists an algorithm that can check if a given input string is a member of the language that runs in  $\mathcal{O}(n^3)$  time, where  $n$  is the length of the string. However, this running time is too high for many practical applications, e.g., natural language processing. Having models for which faster parsing algorithms exist is of crucial importance. On the other hand, HMM is a finite-state model for which such algorithms run in linear time in the size of the input. Thus, we propose HP1CA, HMM-like, as an alternative model which is an infinite-state model, can accept non-regular languages, and whose worst-case computational complexity of the analysis is quadratic in the length of

the input, but in many cases can be even linear.

## 1.1 Research Hypothesis and contribution

The hypothesis of this thesis is that there exist useful models that have lower computational complexity than SCFG. The algorithms for SCFG, which are called Inside-Outside algorithms, are used to compute the probabilities that a given production will be used in a random derivation of a sequence. This is used as an estimation part of the Expectation-Maximization (EM) algorithm to learn the maximum likelihood probabilities for an SCFG [35]. That algorithm is analogous to the Forward-Backward algorithm of HMM and can be used to iteratively re-estimate rule probabilities to maximize the probability by training on the corpus of sentences.

However, the Inside-Outside algorithm have cubic computational complexity to estimate the parameters of the stack-based models which recognize SCFL and its subclasses. To reduce this time complexity, we instead adapt the fundamental learning algorithms of HMM to the stochastic one counter machines setting. The equivalent of HMM in that setting is our new model called hidden probabilistic one-counter automata (HP1CA) which is a P1CA but the control states and counter values are invisible.

The estimation algorithms of HMM are successfully adapted to our new model. The adopted version of algorithms of the new model have quadratic time complexity. Since the new model is P1CA, it also recognizes OCL which is the subclass of SCFL. Thus, the computational complexity of algorithms that are used for estimating the parameters is drastically reduced from cubic to quadratic.

## 1.2 Related Work

In this section, vast literature that is related to the results in this thesis is mentioned. Formal definitions, notions, and results about the main contributed models that constitute the structure of this thesis are discussed in Chapter 3.

### 1.2.1 Hidden Markov Models

HMM is one of the main models that are widely mentioned in this thesis. It is simply defined as that it is an invisible Markov Chain (which is not observed), and hidden states emit random observation symbols, which are visible. HMM was studied by Lawrence Rabiner in the late 1980s [89] on the speech recognition field. Also, it was studied on the pattern recognition ([34],[112], [111]) and challenging against the learning problem of grammars ([110]) show the link that is between automata theory and HMM.

### 1.2.2 Pushdown Systems

The theory of pushdown systems is applied to the verification of recursive programs. The reachability problem of these systems can be solved by analysing the set of configurations with automata ([16],[38]). PDA is a well-known model that can represent the pushdown system and consist of finite control states and finite stack alphabet, where a stack stores the context ([21]). Probabilistic Pushdown Systems (or PPDA) were introduced in [43] with the equivalent recursive model Recursive Markov Chains, which is a natural abstract model for probabilistic procedural programs. PPDA is also used for model checking [39].

### 1.2.3 Counter Systems

The deterministic counter machine is a fundamental model of infinite-state systems where it is a pushdown automaton with only two stack symbols that they are one unique stack symbol and the bottom symbol that anchor at the bottom of the stack and it is never removed ([108]). This is a pushdown machine but the restricted stack works as a counter where pushing the symbol onto the stack simulates incrementing the counter, and popping a symbols simulates decrementing the counter. The special stack symbol allows for testing whether the value of the counter is zero. Probabilistic one-counter automata (POCA) has applications in Markov decision processes ([42]) and stochastic games ([17]).

## 1.3 Outline of the thesis

The main goal of this thesis is to study the complexity of learning problems for the estimation of parameters of probabilistic one counter automata and show that they can be solved in quadratic time.

In Chapter 2 some necessary preliminaries to understand the rest of the thesis are defined. General notation and abbreviations are illustrated in Section 2.1 in Chapter 2. The framework of conditional independence over graphical models, which is the main technique that we use to prove the correctness of the adopted algorithms in Chapters 4 and 5, is presented in Section 2.2. In Section 2.3, the definition of Markov property that essentially underlies the processing of Markov models is studied. This framework allows us to calculate the state transition probability for the adapted algorithms. The last section of Chapter 2 recalls the general definition of some existing automata models.

Chapter 3 consists of two main sections that introduce space state models and infinite-state models, respectively. These two main sections examine the fundamental structures of the contribution models, which are HMM and P1CA, and their related models. It starts by stating the consequences of the equivalence between the state-space model (i.e. HMM) and probabilistic automata (i.e. PFSA) in Section 3.1.1 of Chapter 3. The important structural properties of HMM and its fundamental algorithms whose adaptation is the main inspiration to the main study of this thesis are shown in Section 3.1.2. In order to understand the second contribution model of this thesis, it is necessary to understand the structure of the stack-based models. In Section 3.2.1, fundamental constructions and the operation of PDA, which constitutes the basis of stack-based systems, is defined. In the end of that chapter, the second contribution model which is a kind of infinite-state model P1CA which is the counter version of PPDA is studied.

In Chapter 4, devises a new stochastic counter machine model where the transitions between states cannot be observed due to these variables being hidden. It starts by illustrating the formal definition and the configuration of this new model. Furthermore, the transition relation of the model emphasizes how the counter variable is used during transitions. Moreover, how a particular counter sequence is considered over the calculation of the joint probability distribution of a given observation sequence is shown in Section 4.1.2. Finally, the definition and proofs of the adapted version of the fundamental HMM algorithms which each of them contributes the whole learning algorithm that is used to estimate the parameters of the model are presented.

In Chapter 5, five other possible HP1CA models which are obtained by simply chang-

ing the definition of the initial and final state of the model are described. The chapter starts with a HP1CA model that make use of a distribution on the initial control state, which is also a typical feature of an HMM. The third model consists of multiple final terminals that all have zero counter values. A path that comes from one unique starting terminal can be terminated at one of the final states. Another model is a combination of the second and third type models of HP1CA where it has both multiple initial and final states, but the counter values in both of them are equal to zero. The last two types of models have a unique final terminal, however, the counter value of this terminal can be non-zero. Furthermore, the distinguishing feature of these two models is that one of them has a single initial terminal with zero counter value, and the other has multiple initial terminals (i.e. initial vector) that have zero counter values.

In Chapter 6, the experimental performance of all HP1CA models is presented over two sections. In the first section, we compare the experimental results of the model that is defined in Chapter 4 against HMM. The dataset that is used in the training and testing part is created by Model-1 type HP1CA. Moreover, the datasets that are used in the second part are produced by the models corresponding in the comparison with HMM. The main reason for this is that the other HP1CA models have different initial and final terminals. In the second section, all remaining five models use the same number of hidden variables and the same length of observation sequences while the experiment present the result of comparing those models with HMM.

## Chapter 2

# Preliminaries

In this chapter, we recap the general notation and concepts that are used in further chapters. The general notation and the definition of commonly used probability rules are presented in the first section. In the next section then, conditional independence over graphical models is studied. Markov property which allows us to calculate the state transition probability is introduced in Section 2.3. The last section recalls the general definition of some existing automata models.

### 2.1 General Notations

By  $\mathbb{R}$  we denote set of *reals*, by  $\mathbb{Z}$  the set of *integers*, and by  $\mathbb{N} \stackrel{\text{def}}{=} \{n \geq 0 : n \in \mathbb{Z}\}$  the set of *natural* numbers. We use uppercase letters  $O, S, C$  to denote the sets that are used to indicate the variables of observation, state and counter respectively and we use the corresponding lowercase letter of the set (i.e.  $o, s, c$ ) by adding a subscript index to refer to the elements of the set. An index is generally a number, like in  $o_1$ .

$T$  refers to the length of the observation sequence. The maximum length of each set of variables is equal to the designated time interval, for instance, observation sequence  $O = (o_1, o_2, \dots, o_T)$  consists of  $T$  outputs and at each individual time only one specific output symbol must be produced (i.e.  $o_1 = a, o_2 = d$ , etc.). The other set of variables,  $S$ , and  $C$  have the same notation. We also use an abbreviation to denotes the partial observation sequence as  $o_{1:T}$  which contains output symbols from time  $t = 1$  to  $t = T$ . A counter value is a natural number  $c_t : \{0, 1, \dots\} \rightarrow \mathbb{N}$  where  $c_t \in C$ .

**Definition 2.1.1.** A *probability measure*  $Pr$  is defined on probability space  $(\Omega, \mathcal{F}, Pr)$  where it is on a set  $\Omega$  as a function by following properties [92]:

- $Pr(S) \in [0, 1]$
- $Pr(\Omega) = 1$
- $Pr(\cup_{i=1}^{\infty} S_i) = \sum_{i=1}^{\infty} Pr(S_i)$

The set  $\mathcal{F}$  is called  $\sigma$ -field (i.e.  $\sigma$ -algebra) that contains both  $\Omega$  and  $\emptyset$  and a probability measure  $Pr$  maps  $\mathcal{F} \rightarrow [0, 1]$ .

In probability theory there are two elementary rules, which are also used when proving the adapted version of the fundamental algorithms of HMM, where they are known as *sum rule* and the *product rule*. We will use the simple definition of these two rules due to be trivial explain more details here. For more details [55].

**Definition 2.1.2.** A *sum rule* is applied when two events (e.g.  $S_i$  and  $S_j$ ) that could occur in different ways but they could not be performed simultaneously.

$$Pr(S_i) = \sum_{S_j} Pr(S_i, S_j) \quad (2.1)$$

**Definition 2.1.3.** A *product rule* is used when a process is separated into two successive events where  $S_i$  denotes the first event and  $S_j$  represents the second event (i.e they are independent)

$$Pr(S_i, S_j) = Pr(S_j|S_i)Pr(S_i) \quad (2.2)$$

## 2.2 Conditional Independence

The main aspect of this thesis is adopting three parameter learning algorithms for *HMM* to our model. Both models *HMM* and *HP1CA* process long random sequences. A graphical model with a diagrammatic representation allow for easy analysis [14].

**Definition 2.2.1.** Two frequent graphical model forms *directed acyclic graphs* (DAG)(e.g. *Bayesian network*) and *undirected graphs* (e.g. *Markov random fields*) are commonly used

to exhibit the acyclic conditional dependence structure between random variables [57]. The models in this thesis is based on DAG and a DAG contains *nodes* (it can also be called *vertices*) which are identified with random variables and *arrows* (or it is called *edges*) where provides the link which express probabilistic relationship between the nodes [57, 109, 88]. According to the definition of [57] a DAG is represented by  $G(V, E)$  where  $V$  are nodes (i.e. variables) and  $E$  are the edges of the graph. Each random variables of the graph are indexed by the nodes where  $\{S_v : v \in V\}$ .

**Definition 2.2.2.** A Bayesian Networks (BN) is a directed acyclic graph that represents the joint probability distributions over a set of random variables which are nodes (i.e. vertices)  $V$  and edges  $E$  [33, 65]. Each edge corresponds to a conditional dependency, and each node corresponds to a random variable. Let  $S = \{S_1, \dots, S_N\}$  be  $N$  variables and the probability distribution of the variables in the network is denoted as  $Pr(S_i|a_{S_i})$  where  $a_{S_i}$  is the parent of the variable  $S_i$  [77]. The edge is represented by an arrow from  $a_{S_i}$  to  $S_i$ . When there is no parents of  $S_i$ , then it is a prior probability distribution and can be shown as  $Pr(S_i)$ .

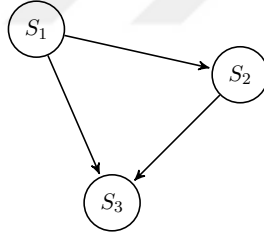


Figure 2.1: A DAG diagram that represent the joint distribution  $Pr(S_1, S_2, S_3)$  of three variables of the network where it is convenient to be formed by application of the product rule as  $Pr(S_1, S_2, S_3) = Pr(S_3|S_1, S_2)Pr(S_2|S_1)Pr(S_1)$ . Here the link goes from  $S_1$  to  $S_2$  then  $S_1$  is the parent of the node  $S_2$ .

**Definition 2.2.3.** A *conditional independence* of two random variables could be simply explained as one of the variables does not amend the probability distribution of the other variable [114, 7, 103, 99]. Let  $G$  denotes the given DAG and the joint distribution over the set of  $S = \{S_1, \dots, S_N\}$  recognize the recursive product rule [82]:

$$Pr(S_1, \dots, S_N) = \prod_i Pr(S_i|a_{S_i}) \quad (2.3)$$

where  $a_{S_i}$  is the parent of  $S_i$  in  $G$ . For instance, the DAG in Figure 2.2a determines the recursive decomposition

$$Pr(S_1, S_2, S_3) = Pr(S_3|S_1)Pr(S_2|S_1)Pr(S_1) \quad (2.4)$$

According to [82], Equation 2.3 indicates that each variable of the  $S_i$  is conditionally independent of all its other variables by given its parent set  $a_{S_i}$  and such a set of independences is also called *Markovian* where it indicates that each state becomes independent of the past. The shorthand notation  $\perp\!\!\!\perp$  is used for conditional independence [31] in which

$$S_3 \perp\!\!\!\perp S_2 | S_1 \quad (2.5)$$

denotes that  $S_3$  is conditionally independent of  $S_2$  given  $S_1$ . In addition, the decomposition over the set of variables in Equation 2.3 indicates repeated many independencies that are tested by using the sum and product rules of the probability. This kind of approach can be read directly from the DAG by using the graphical feature that is called *d-separation* [81].

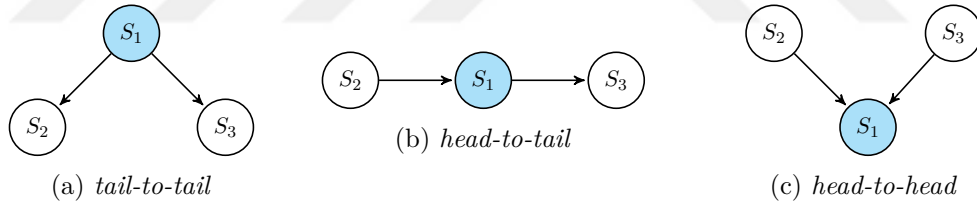


Figure 2.2: Three illustrates of the concepts of the *d-separation*.

*d-separation* is an approach that is used as an elegant way to read the conditional independence properties of the joint distribution directly from the DAG in Bayesian Network. There are three simple examples that would be considered with respect to connectivity concepts which are *tail-to-tail*, *head-to-tail* and *head-to-head*. For more details please see [14].

- **tail-to-tail:** There is a *divergent* variable where it is a parent of its both neighbors. Let assume that we have three random variables which are indicated as  $S_1$ ,  $S_2$ , and  $S_3$ . Now suppose we have the conditioned node  $S_1$  as illustrated in Figure 2.2a that blocks the path from  $S_2$  to  $S_3$ . The node  $S_1$ , which is connected to the tails of two

arrows, causes  $S_2$  and  $S_3$  to become conditionally independent.

$$S_2 \perp\!\!\!\perp S_3 | S_1 \quad (2.6)$$

- **head-to-tail:** There is a *sequential* variable where it is a child of one of its neighbors and a parent of the other. Now the node  $S_1$  is *head-to-tail* variable with respect to connected two arrow which one is head and the other is tail. The node blocks the path from  $S_2$  to  $S_3$  as in Figure 2.2b and so, we can obtain the following conditional independence

$$S_2 \perp\!\!\!\perp S_3 | S_1 \quad (2.7)$$

- **head-to-head:** The final concept is a *convergent* variable where it is a child of both its neighbors shown in Figure 2.2c. The node  $S_1$  is now *head-to-head* variable with respect to the heads of two arrows which block the path when it is unobserved. It means that  $S_2$  and  $S_3$  are independent with no variables observed.

$$S_2 \perp\!\!\!\perp S_3 | \emptyset \quad (2.8)$$

Thus the nodes  $S_2$  and  $S_3$  are not independent with conditioning on  $S_1$

$$S_2 \not\perp\!\!\!\perp S_3 | S_1 \quad (2.9)$$

In this concept of *d-separation*, we need also to consider descendants of  $S_1$ . It means that if there is a single path in which more than one variable on the path between  $S_1$  and its descendants and if it is observed either the node or its descendants the path becomes unblocked.

## 2.3 Markov Property

The Markov property is generally explained as *memoryless* property of a stochastic process where the conditional probability distribution of the future and the past states of the process are independent when the current state is known (i.e. conditioning on the current state) [86].

**Definition 2.3.1.** Suppose that  $X = (X_1, \dots, X_T)$  is a random variable sequence that have values from the set of state space  $S = \{s_1, \dots, s_N\}$ . Two Markov properties that

provide the tractable aspect about time series are [71]:

- **Limited Horizon:** The probability of being in the future state at time  $t+1$  depends only on the current state at time  $t$ .

$$Pr(X_{t+1} = s_k | X_1, \dots, X_t) = Pr(X_{t+1} = s_k | X_t) \quad (2.10)$$

- **Time Stationary:** The conditional probability distribution over next state conditioning on current state does not change over time.

$$Pr(X_{t+1} = s_k | X_t) = Pr(X_2 = s_k | X_1) \quad t \in 1 \dots T \quad (2.11)$$

In brief, the Markov property of a random process states that its future evolution depends only on its present state rather than its past [54].

The assumptions (i.e. Markov properties) described above are used in Markov chain which is a stochastic process that evolves over on sequence of random variables. Markov chain is defined as it is a Markov process in either discrete or continuous time [62]. The discrete time Markov chain is the basis of the Hidden Markov Model but the states are not hidden. The formal definition is [60] :

**Definition 2.3.2.** Let  $X = (X_1, \dots, X_T)$  is a random variable sequence that takes the value from the set of state-space  $S = \{s_1, \dots, s_N\}$ . According to the probability space definition 2.1.1, every random variable  $X$  with a value from the set of  $S$  is a function  $X : \Omega \rightarrow S$ . A time-homogeneous Markov chain contains initial vector  $\pi$  and transition matrix  $A$  where

- $Pr(X_0 = i) = \pi(i)$  initial distribution where  $i \in S$
- $Pr(X_{t+1} = i | X_t = j) = A(i, j)$  state transition matrix where  $i, j \in S$

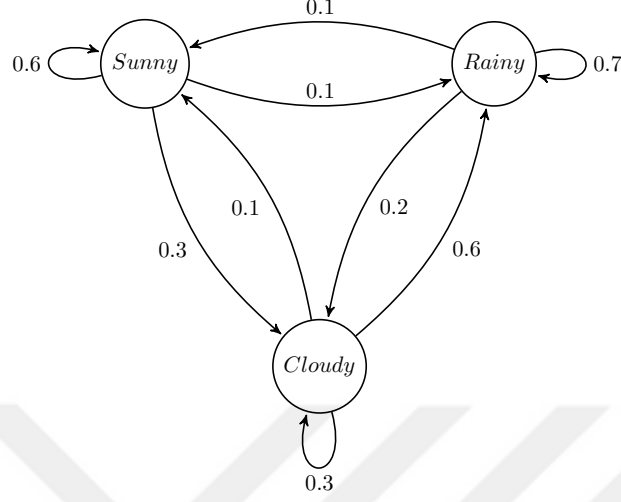


Figure 2.3: An example Markov chain process. Each edge depicts the probability of taking that transition from its origin state. There are three states in the set of state-space where  $S = \{Sunny, Rainy, Cloudy\}$

The initial distribution of the weather example of the Markov chain in Figure 2.3 is  $\pi = [0.5, 0.4, 0.1]$  which means that the probability of starting in state *Rainy* is 0.4, the probability of starting in state *Sunny* is 0.5, etc. The transition from *Sunny* to *Rainy* is denoted by  $A(S_{Sunny}, S_{Rainy})$  and the probability of this transition is

$$Pr(X_{t+1} = S_{Rainy} | X_t = S_{Sunny}) = 0.1 \quad (2.12)$$

The other transition probability from *Rainy* to *Rainy* (i.e. hold in the same state on next step) is 0.7, etc. The transition matrix  $A$  and the initial distribution vector  $\pi$  must satisfy the probability assumption which is that the sum of each row of the matrix equals to one [79] where

$$\sum_{j \in S} A(i, j) = 1 \quad \forall i \in S \quad (2.13)$$

and initial distribution

$$\sum_{i \in S} \pi(i) = 1 \quad \forall i \in S \quad (2.14)$$

## 2.4 Automata Formalisms

In this section, we begin to step into the field of the automata which is generally described as a specific process where it is carrying out the production via automatic processes [47]. The two models Hidden Markov Model (HMM) and Probabilistic One-Counter Automata (P1CA) which are important to this thesis are rooted in the automata theory.

A Markov Chain (MC) can be represented by Finite State Automata (FSA) but the transitions are probabilistic rather than deterministic [60]. On the one side, HMM is a generalisation of the MC and on the other a P1CA is a Pushdown Automata (PDA) with one stack symbol. The both models have a relation to the theoretical automata study. More details about these models can found in Sections 3.1 and 3.2, respectively.

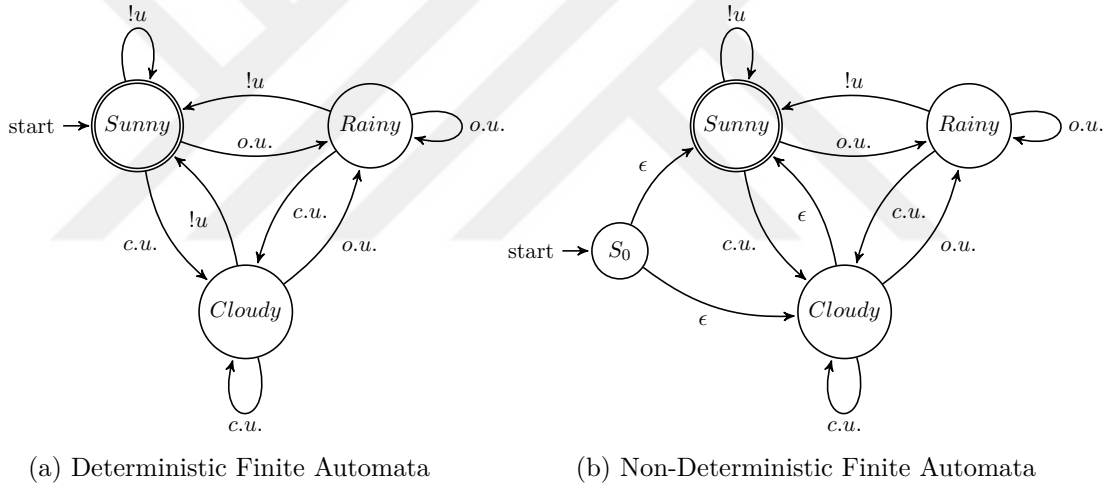


Figure 2.4: An example for deterministic and non-deterministic finite automata

For instance, let us assume that we have a Finite State Automaton as shown in Figure 2.4a. A state is depicted by circles and the transition is illustrated with arrows. The starting state is denoted as an arrow which comes from labeled *start* and it is pointed to the node with its head (e.g. the starting state is *Sunny* in Figure 2.4a). The example represents the using of an *umbrella* (i.e. *u* represents the abbreviation of the umbrella) with respect to weather condition of a day. The umbrella is not used while the weather is *Sunny* (i.e. being in state *Sunny*). The input when an umbrella is not used is labeled as *!u* on the edge. If the umbrella is open then it is shown as *o.u* and the state changes to *Rainy*. Lastly,

if the umbrella is closed that is denoted as *c.u.* then the current state is updated to *Cloudy*.

In this example, the input words that are labeled over the arrows indicate the transition function value instead of the real numbers that are used in Figure 2.3. The transitions function of *FSA* is generally shown by the elements of a non-empty finite set of *alphabet* [84]. The general notation of the alphabet is shown with  $\Sigma = \{!u, c.u., o.u.\}$ . And each element of the alphabet is called a *letter*. A *word* is a set of the finite sequence of letters that could be generated by using the letters from  $\Sigma$ . For example, *!u!uc.u.o.u.!u* is a word over the alphabet  $\Sigma$  where it is hold in input *tape* divided into *cells* that is read by the control state as shown in Figure 2.5.

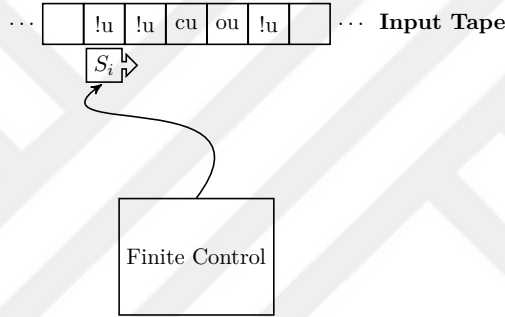


Figure 2.5: Input tape with reading head

Initially, the head is positioned on the first cell of the input tape to start the reading of given word. The control state is in the starting state (e.g. according to Figure 2.5, it is  $S_i = \text{Sunny}$ ). Through each step, the head reads the cell first then moves to one cell to the right. The reading action updates the condition of the current state according to the transition function. This action process either stays in the current state or moves to the next state. Accepting a word could only be done if the current state reaches the final state after the reading of input tape is completely done.

The behavior of *FSA* can be split into two types in which every transition is uniquely determined or whenever two or more transitions may be applicable at any time.

**Definition 2.4.1.** A *Deterministic Finite Automata (DFA)* is tuple  $(Q, \Sigma, \delta, S_0, S_T)$  [52, 51, 69, 37] where

- $Q$  is the non-empty set of the states

- $\Sigma$  is an alphabet
- $\delta : Q \times \Sigma \rightarrow Q$  a transition function
- $S_0 \in Q$  is the initial state
- $S_T \in Q$  is the set of final (i.e. accepting) states

**Definition 2.4.2.** A *Non-Deterministic Finite Automata (NFA)* is tuple  $(Q, \Sigma, \delta, Q_0, S_T)$  where

- $Q, \Sigma$  and  $S_T$  are as for DFA
- $Q_0$  is a non-empty set of initial states
- $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$  a transition relation where  $\mathcal{P}(Q)$  denotes the power set of  $Q$

## 2.5 Conclusion

In this section we briefly described the notations which are used in the other chapters. DAG model is a useful tool to provide a significantly advantage of easy understanding of the calculation for the model. *MC* and *FSA* are the graphical models and *MC* is a *FSA* however transition matrix  $A$  is used for probabilistic distribution instead of transition function  $\delta$  that is made it to become more related to *probabilistic finite state automata (PFSA)* [107]. In the next section we will examine the two main models which one of them is the *MC* but the states are hidden and the emission matrix  $B$  is used and the other model *PICA* which is a subclass of Probabilistic Pushdown Automata (*PPDA*) that is a stack based model of *PDA* with assigned probability value to transition function. Only one stack symbol is used in this model.

## Chapter 3

# Probabilistic State Space and Infinite State Models

In this chapter, we define two models that allow us to specify the main model of this thesis. These two models are Hidden Markov Model (*HMM*) and Probabilistic One-Counter Automata (*PICA*). Those both models are based on Finite State Automata (*FSA*). For instance, *HMM* is generally expressed as the expansion of Markov Chain (*MC*) which can be identified with Probabilistic Finite State Machine (*PFSM*). Both models are the DAG model where the probability distribution is used for updating the current state. In sub-section 3.1.1, we introduce the *PFSM* and its equivalence with *HMM*. In the next sub-section, we introduce the general definition of *HMM* and its three fundamental learning algorithms.

The second model Probabilistic One-Counter Automata (*PICA*) is a stack-based automata which is a subclass of the *Probabilistic Push-down Automata* (*PPDA*) with only one stack symbol. *PPDA* is a Pushdown Automata (*PDA*) where a probability distribution is assigned to the transition function that can look at the top of the stack and push a new symbol on top or pop the top symbol. In this section, we will first introduce the stack base systems where the stack size is unbounded. In sub-section 3.2.2, we will show how a stack may be used as a counter by using only one stack symbol. In the last sub-section, *PICA* model will be defined and how it processes a given input.

### 3.1 State Space Models

The term *state-space* was firstly used by Kalman [61] for the study of estimation of the states and the parameters from the data. This probabilistic graphical model is used for dynamic systems, where they exhibit a changing proceeded in a process, in time series analysis. Some fields that applied this model are engineering, statistics, computer science and economics to solve a wide range of dynamical systems problems.

A state space model consists of hidden sequence of variables which are also called states and an observed sequence of variables that is called observations. This graphical model, *state-space*, describes the relations between states and observations [64] as shown in Figure 3.1. These are the dependencies of the observations on the hidden variables in this state-space model.

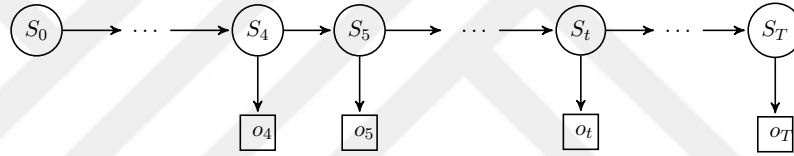


Figure 3.1: An example of state-space based on Markovian property

The Markov process lays at the core of the state space model by governing how the probability distribution on the transition to the the current state at time  $t$  from the the state at previous time  $t - 1$  [72, 29]. *HMM* is a well-known state-space model that is used in many research areas [27]. This model uses the Markov assumptions as a key to evaluate the state transition density. That means that if the current state  $S_5$  is known then the previous state  $S_4$  does not add any information about the later states  $S_6, \dots, S_T$ . The other property that a state-space model has is observation density which describes the relationship between the sequence of the latent variable (i.e., states  $S_t$ ) and the output variable (i.e., observations  $o_t$ ) at time  $t$ .

To summarize the state space model, we can write the probability distribution as follows [63]:

- **State Equation:**

$$Pr(S_{t+1}|S_t) = f(S_{t+1}|S_t) \quad (3.1)$$

• **Observation Equation:**

$$Pr(o_t|S_t) = g(o_t|S_t) \quad (3.2)$$

In Figure 3.1,  $S_0$  we present the initial state of the model. The distribution of the initial state is given as

$$Pr(S_0) = p(S_0) \quad (3.3)$$

In conclusion, an HMM is a state-space model which contains hidden variable sequence and a time series output data which exhibits a memoryless dynamical behavior. It means that there is a relationship between the sequence of observations and the latent states at the same time  $t$ . As it is known that our other main basis model *PICA* is a *PPDA* with has only one stack symbol and it is a another automata variant that constitutes the basis of automata theory. In order to show that *HMM* are also equivalent to this kind of automata, in next section we will introduce *PFSM* and its relation to *HMM* by showing how the main algorithms of *HMM* can be used for parsing in *PFSM*.

### 3.1.1 Definition of Probabilistic Finite State Automata (PFSM) and Equivalence to HMM

The essential definition of a *PFSM* is a generalization of non-deterministic *Finite State Automata (NFSA)*, with empty string (i.e.,  $\epsilon$ ), in which every state and every transition is associated with a probability value [80]. This framework has been used in various research areas for behavior modeling of dynamic systems in speech recognition [73], a tool based on *PFSM* for statistical pattern recognition [20, 56, 6], computational linguistics [18], symbolic time series analysis for signal processing [91, 90].

Basically, the only difference of a *PFSM* from *FSM* is a probabilistic choice that is used for the next state of the target transition. Under this aspect, a *PFSM* might have a formal definition which is a similar but extended concept of an *FSM*. There is a variety of formal definitions regarding *PFSM* in the literature but all have same meaning [102, 25, 8]:

**Definition 3.1.1.** A *PFSM* is a tuple  $\mathcal{A} = (Q, \Sigma, \delta, I, P, F)$  where

- $Q$  is a finite set of *states*
- $\Sigma$  is the *alphabet*

- $\delta \subseteq Q \times \Sigma \times Q$  is a set of *transitions*
- $I : Q \rightarrow \mathbb{R}^+$  *initial* state probabilities
- $P : \delta \rightarrow \mathbb{R}^+$  *transition* probabilities
- $F : Q \rightarrow \mathbb{R}^+$  *final* state probabilities

Each probability value is a non-negative real number and the sum of probabilities for a given state and symbol must equal to 1.  $I, P$  and  $F$  are functions such that

$$\sum_{q \in Q} I(q) = 1 \quad (3.4)$$

and

$$F(q) + \sum_{q' \in Q, a \in \Sigma} P(q, a, q') = 1 \quad \forall q \in Q \quad (3.5)$$

The probability where a particular path is taken depends on the probability distribution for that transition. The state-symbol pair represents the available paths and defined in terms of their particular transition probabilities. An example *PFSM* is presented in Figure 3.2. It starts by determining the initial state by using initial distribution. Then, it either stays in the current state or chooses randomly a path that depend on the transition probabilities. A symbol is labeling each transition with the corresponding probability value.

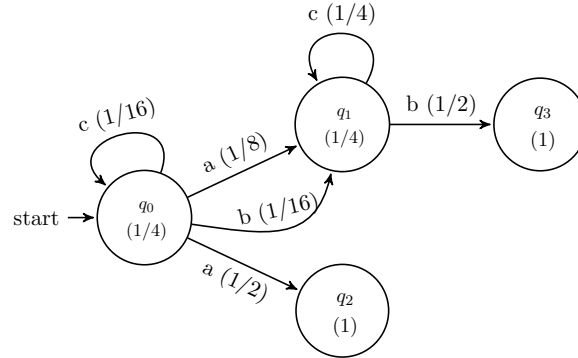


Figure 3.2: An example for the graphical representation of *PFSM*

Given a *PFSM*  $\mathcal{A}$ , it generates a string by following two processes as

- **Initialisation:** A state is determined as an initial state with respect to initial distribution. Let  $q_0 \in Q$  be the starting state as shown in 3.2.

- **Iteration:** Let  $q$  represents the current state. Staying in the same state happens with probability  $F(q)$ . Moving to the next state happens by executing a transition with probability  $P(q, a, q')$  where  $a \in \Sigma$  and  $q' \in Q$ . After a valid move, an output symbol  $a$  is generated and the current state is changed to  $q'$ .

The algebraic structure of  $FSM$  contains a finite set of states and a finite set of letters (alphabet). Each state is connected by transitions where each transition is labeled with a symbol in the alphabet. After reading an input symbol from the tape, the automaton moves from one state to another according to the transition function. Accepting the given input word is only valid if the automaton reaches the final state while the end of the tape is reached. However, the basic structure of  $PFSM$  is  $FSM$  but it allows for probabilistic transitions. Thus, a given string (i.e. word) can be generated by  $PFSM$  through multiple valid paths instead of reading input tape to decide whether it is accepting or rejecting.

Let  $\Delta(o)$  represents the set of all valid paths for the string  $o$  in  $PFSM \mathcal{A}$ . The probability of generating  $o$  with the model  $\mathcal{A}$  is

$$Pr(o) = \sum_{path \in \Delta(o)} Pr(path) \quad (3.6)$$

For instance, let a  $PFSM$  as shown in Figure 3.2 is given and assume that the given string is  $cacb$ . There is only one valid path for this string

$$\Delta(o) = \{(q_0, c, q_0, a, q_1, c, q_1, b, q_3)\}$$

The probability of this string is:

$$\begin{aligned} Pr(cacb) &= I(q_0) \cdot P(q_0, c, q_0) \cdot P(q_0, a, q_1) \cdot P(q_1, c, q_1) \cdot P(q_1, b, q_3) \cdot F(q_3) \\ &= 1.0 \cdot 0.0625 \cdot 0.125 \cdot 0.25 \cdot 0.5 \cdot 1.0 \\ &= 0.0009765625 \end{aligned}$$

The given example in Figure 3.2 is ambiguous. For example, there are two different valid paths which are  $\Delta(a) = \{(q_0, a, q_1), (q_0, a, q_2)\}$  for the string  $a$ . The probability of the

string  $a$  is:

$$\begin{aligned}
 Pr(cacb) &= I(q_0) \cdot P(q_0, a, q_1) \cdot F(q_1) + I(q_0) \cdot P(q_0, a, q_2) \cdot F(q_2) \\
 &= 1.0 \cdot 0.125 \cdot 0.25 + 1.0 \cdot 0.5 \cdot 1.0 \\
 &= 0.53125
 \end{aligned}$$

According to [110], an automaton model could be learned from observations of the system. *PFSM* is such a model where different states can generate different symbols with different probabilities. On the other hand, Hidden Markov Model (HMM) also has similar structure learning from the data gathered from observations. However, the symbols are emitted at the states instead of at the transitions which are only used to move. Thus, there are various studied works to show the link between *PFSM* and *HMM*. Such as, using transition emission distribution instead of emission matrix which is equivalent to *HMM* [34], comparing the methods for grammatical inference [110], parsing a string by using *HMM* algorithms [112, 111]. The work [112] is more relevant to the subject of this thesis by representing how dynamic programming algorithm, which is the similar idea that is proposed for *HMM*, is used to compute the parsing efficiently in *PFSM*.

### Parsing with *PFSM*

Let assume that we have the paths  $\Delta(o)$  to compute the probability of  $o$  efficiently. *Forward algorithm*, as shown  $\alpha_t(q) \forall q \in Q$  where  $0 \leq t \leq T$  (i.e.  $T = |o|$  represents the length of the given string), computes the probability of generating the prefix of the string (i.e.  $o_1 \dots o_t$ ) and reaching state  $q$ :

- *Base Step*:

$$\alpha_0(q) = I(q) \tag{3.7}$$

- *Induction Step*:

$$\alpha_t(q) = \sum_{q' \in Q} \alpha_{t-1}(q') \cdot P(q', o_t, q), \quad 1 \leq t \leq T \tag{3.8}$$

The total probability of the given string  $o$  is:

$$Pr(o) = \sum_{q \in Q} \alpha_T(q) \cdot F(q) \tag{3.9}$$

The other algorithm for computing the given string is *Backward algorithm*, denoted as  $\beta_t(q)$ , that computes the probability of generating the suffix  $o_{t+1} \dots o_T$  from the state  $q$ :

- *Base Step*:

$$\beta_T(q) = F(q) \quad (3.10)$$

- *Induction Step*:

$$\beta_t(q) = \sum_{q' \in Q} \beta_{t+1}(q') \cdot P(q, o_t, q'), \quad 0 \leq t \leq T-1 \quad (3.11)$$

The total probability of the given string  $o$  is:

$$Pr(o) = \sum_{q \in Q} I(q) \cdot \beta_0(q) \quad (3.12)$$

According to Equation 3.6, a *PFSM* generates a string by summing the probability of all executed paths. However, to maximize the probability of generating the string is provided by choosing the maximum probability value corresponding to the optimal path. The computation of the probability of this maximization aspect can be efficiently performed by using *Viterbi algorithm*. It is represented  $v_t(q)$  where  $\forall q \in Q, 0 \leq t \leq T$ , as it calculates the maximum path of generating the prefix  $o_1 \dots o_t$  and reaching state  $q$ :

- *Base Step*:

$$v_0(q) = I(q) \quad (3.13)$$

- *Induction Step*:

$$v_t(q) = \max_{q' \in Q} v_{t-1}(q') \cdot P(q', o_t, q), \quad 1 \leq t \leq T \quad (3.14)$$

The maximum probability of generating the string  $o$  is:

$$Pr(o) = \max_{q \in Q} v_T(q) \cdot F(q) \quad (3.15)$$

These algorithms have the same computational complexity  $\mathcal{O}(T \cdot N)$  where  $T$  is the length of the string and  $N$  is the number of the transitions in *PFSM*  $\mathcal{A}$ . According to [111], *Viterbi re-estimation algorithm* produces same solution as *Baum-Welch algorithm* which is

used to estimate the parameters of the model  $\mathcal{A}$ .

In conclusion, we needed to find a link between this kind of automata and *HMM*. In the automata theory area, the *probabilistic automata*, which uses corresponding algorithms of *HMM* for parsing a string, is the closest automata to *HMM*. This aspect illustrates that different automata can use these fundamental algorithms by adapting them to their own framework principles. Therefore, we adopted those algorithms for our model *HPICA* which is a *PICA* but the control states and counter value are hidden and they emit an output symbol for each discrete time  $t$  as *HMM* does. It is a generalization of a counter machine where it is a subclass of *PPDA* which is stack base automata that is generalized from FSM. In the next section, we will show with details these two models which are important to this thesis.

### 3.1.2 Definition of HMM

The *HMM* can be viewed as a state-space model of Figure 3.1 in which has unobservable variables (i.e. hidden states) and observable variables which are generated by latent variables by applying stochastic process at a discrete-time. It comprises a Markov Chain, which is a stochastic model that governs the probability of sequences of random variables (i.e states), but the states are not observed directly. This model is widely used in different fields as a statistical tool for modelling discrete time-series. It was developed by Baum and Petrie [11] to estimate the parameter of the model. The other fields that are interested in the theory and application of *HMM*:

- **Various kinds of recognition:** speech recognition [4, 89, 26], handwriting recognition [87], [53], [117], face recognition [75, 28, 94], gesture recognition [115, 78, 74]
- **Bioinformatics:** Biological sequence analysis [96], [101], [95], gene prediction [83],[70]
- **Environment:** earthquakes [13, 116, 23], weather [59, 58, 30]
- **Finance:** stock market analysis [49, 76]

Let us consider the following HMM example. The weather is a stochastic process and let assume that we have three states which represent the weather conditions as show 3.3a. If the current state is *Sunny*, there are different chances that the next state will be either *Sunny* or one of the other conditions. In Markov Chain, the states of the model are directly

observed, however in *HMM* only the sequence of observable variables are observed, and probabilistically produced by hidden variables of the model. Let assume that we use seaweed to make an inference the condition of the weather due to there is no chance directly knowing the weather. We have only knowledge that the seaweed might be dry, damp, soggy depending on the weather. Each day, the condition of the seaweed is recorded to make an observation sequence for adequately modeling this scenario by applied *HMM*. The weather is a hidden variable and the condition of seaweed is an observable variable in this model.

In Figure 3.3b, an observation symbol  $o_t$  is emitted by each latent state corresponding to the probability distribution denoted as  $B(S_t, o_t)$ . The set of observation symbol is  $O = \{dry, damp, soggy\}$ , and at each individual time step  $t$  one of the observable symbols  $o_t \in O$  is produced by a latent variable. As it was mentioned before an unobservable Markov chain produces stochasticity output symbols. The transition distribution between hidden states is denoted as  $A(S_{t-1}, S_t)$  where  $S_{t-1}$  represents the previous hidden state, and  $S_t$  is current hidden state in the model.  $\pi(S_0)$  is the initial vector that stores the initial distribution probability of the model. In this example the  $\pi(S_1)$  is the probability of state  $S_1$  being the start state of the state sequence.

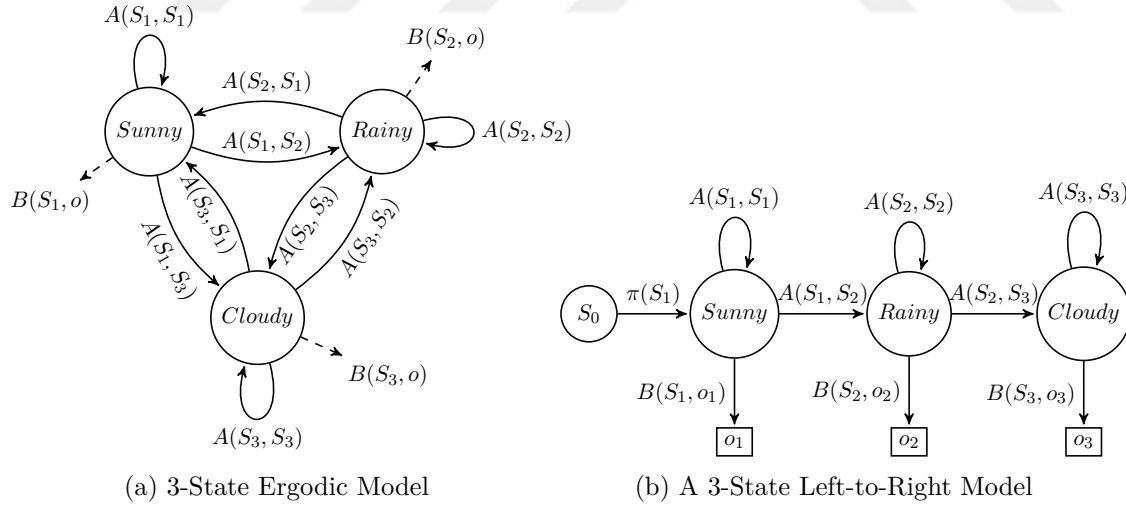


Figure 3.3: An example for HMM structure

An *HMM* can be defined as a tuple  $\theta = (S, O, \pi, A, B)$  where

- $S$  is the set of hidden states in the model  $S = \{Sunny, Rainy, Cloudy\}$

- $V$  is the set of observation symbols in the model  $V = \{dry, damp, soggy\}$
- $\pi$  is initial state vector the probability of the initial state  $S_0$  being in  $q$  at time  $t = 0$ :

$$\begin{aligned}\pi(q) &= Pr(S_0 = q), \quad q \in S \\ \sum_{q \in S} \pi(q) &= 1\end{aligned}\tag{3.16}$$

- $A$  is state transition probability distribution where  $A(S_{t-1}, S_t)$  stores the probabilities of state  $S_t$  following state  $S_{t-1}$ :

$$\begin{aligned}A(S_{t-1}, S_t) &= Pr(S_t = q | S_{t-1} = q'), \quad q, q' \in S \\ \sum_{q \in S} A(q', q) &= 1 \quad \forall q' \in S\end{aligned}\tag{3.17}$$

- $B$  is emission symbol probability distribution where  $B(S_t, o)$  stores the probability of observation symbol  $o$  being produced by the hidden state  $S_t$ , discrete time  $t$ :

$$\begin{aligned}B(S_t, o) &= Pr(o_t = o | S_t = q), \quad o \in V \text{ and } q \in S \\ \sum_{o \in O} B(q, o) &= 1 \quad \forall q \in S\end{aligned}\tag{3.18}$$

The distribution of each variable in the sequence of observation might be different for each  $t$  due to *HMM*'s stochastic behavior. In Figure 3.4, the random variables  $\{S_1, S_2, S_3, S_4, S_5\}$  are the hidden states and they are related to the other hidden variables based on a conditional probability distribution. Due to *HMM* being stochastic and Markovian model, it is clear that Markov properties can be applied to these models. Thus, the hidden variable  $S_t$  at time  $t$ , depends only on the value of hidden variable  $S_{t-1}$ . The hidden value at time  $t-2$  or before that have no influence. For instance, let assume that we have an output sequence (*accbda*) (as in Figure 3.4) which is generated by the state sequence ( $S_1 S_5 S_2 S_4 S_3 S_2$ ). First, we choose the start state based on the initial state distribution, in this instance  $S_1$  has been chosen, and then produce a corresponding observation symbol from the set of observation symbols  $\{a, b, c, d\}$  according to the emission probability. The hidden state  $S_1$  emits the symbol  $a$  at time  $t = 1$ . Evaluating state condition will be chosen conditionally depending on the current state as  $Pr(S_5 | S_1)$  which is the first order of Markov property. Thus, the hidden state at time  $t = 2$  is  $S_5$  and the observation symbol drawn is  $c$ . The next latent

variable will be chosen through the same process and so on.

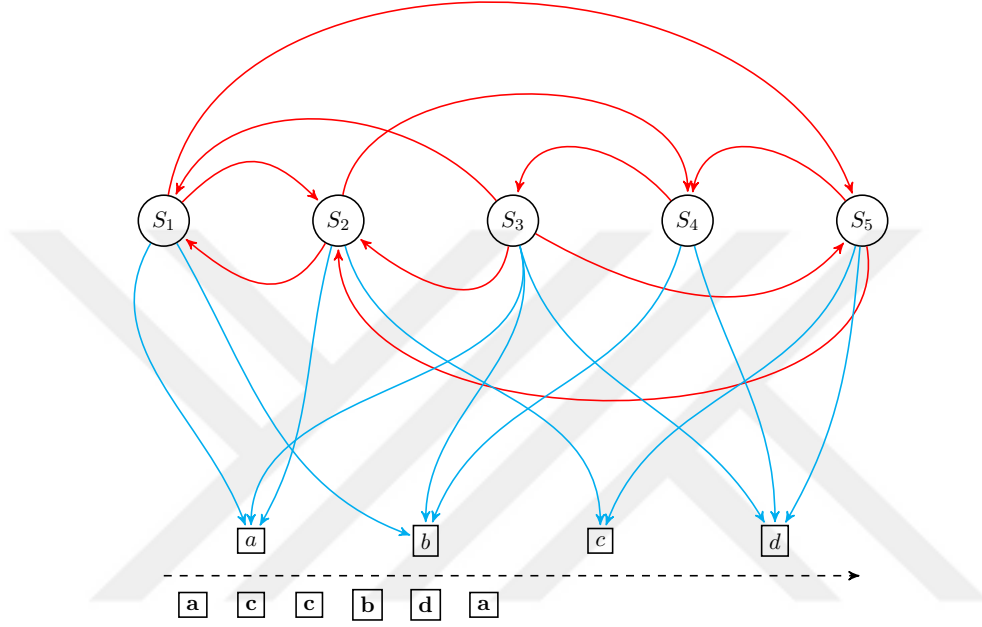


Figure 3.4: The observed output sequence that has been produced by the most likely sequence of hidden states

The computation of the joint probability of being in a particular sequence of latent state  $Q$  and corresponding the generating a particular observation event sequence  $O$  is given by

$$\begin{aligned} Pr(O, Q) &= Pr(O|Q) \times Pr(Q) \\ &= \prod_{t=1}^T Pr(o_t|S_t) \times Pr(S_0) \prod_{t=1}^T Pr(S_t|S_{t-1}) \end{aligned} \quad (3.19)$$

The computation of the probability of a particular observation sequence is calculated by summing over all possible state sequence

$$Pr(O) = \sum_Q Pr(O, Q) = \sum_Q Pr(O|Q)Pr(Q) \quad (3.20)$$

For the particular example that is shown in Figure 3.4 we would sum over

$$Pr(accbda) = Pr(accbda, S_1 S_1 S_1 S_1 S_1) + Pr(accbda, S_1 S_1 S_1 S_1 S_2) + \dots \quad (3.21)$$

The calculation of the probability of the observation sequence requires about  $TN^{T+1}$  additions and multiplications where  $T$  is the length of the sequence of the observation and  $N$  is the number of latent variables. However, this calculation would be computationally infeasible even for small values of  $N$  and  $T$ . The more efficient way to solve this calculation instead of using such an exponential algorithm is *Forward* and *Backward* algorithms which are a kind of *dynamic programming* algorithm that reduce the calculation complexity to solve these problems. In the next section, three main problems for *HMMs* and *dynamic programming* algorithms solving them will be presented. These algorithms are the basis for the parameters estimation algorithms that are the main contribution of this thesis.

### 3.1.3 Fundamental Algorithms of HMM

An influential tutorial [89] introduced the idea that *HMMs* are characterized by three fundamental problems

- **Likelihood Problem:** According to this problem we have a given sequence of observation  $O$  and we would like to determine the probability of the output sequence that generated a given *HMM*  $\theta = (\pi, A, B)$ .
- **Decoding Problem:** Again we are given an observation sequence  $O$  and the model  $\theta = (\pi, A, B)$ . We now would like to determine the most likely sequence of hidden states that generated these observations.
- **Learning Problem:** We are given observation sequences and the set of hidden states of *HMM*. We would like to determine the optimal set of model parameters to maximize the likelihood of generating these observation sequences.

#### Likelihood Computation: Forward Algorithm and Backward Algorithm

As it was defined that the computation of the probability of the sequence of observation that is generated by a *HMM* is simply calculated by summing the probabilities from all of the latent state sequence multiplied by the probability distribution of corresponding

observable symbol in each hidden state. However, this aspect provides an infeasible computation as it has exponential computational complexity. The *forward* algorithm provides a recursive calculation that explicitly reduces the calculation complexity of the first problem for *HMMs*.

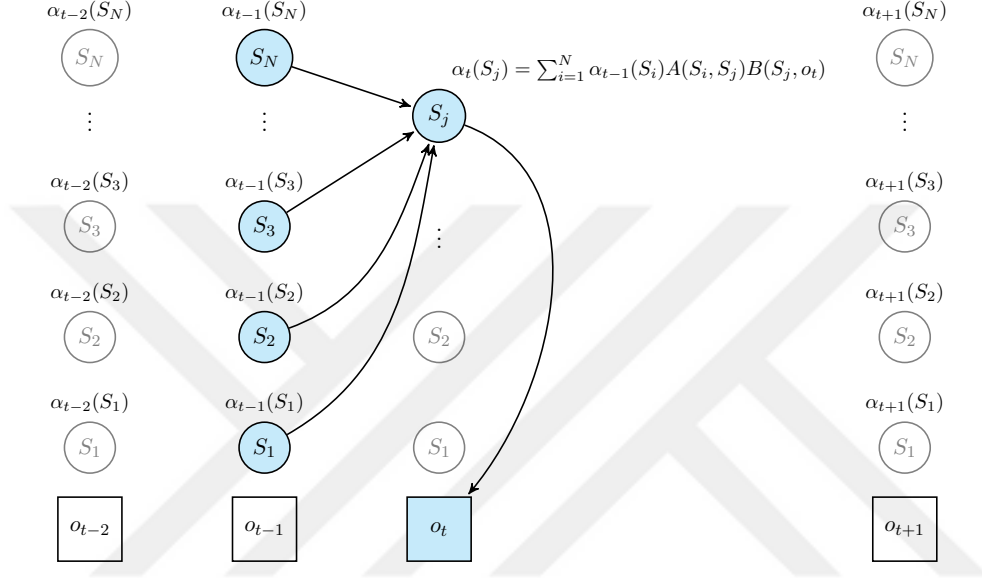


Figure 3.5: The forward trellis for computing the total probability of the observation by summing all the previous forward values multiplied by transition probabilities and emission probability.

The *forward* algorithm computes the likelihood of the observation by summing over the probabilities of all possible paths that led to the sequence of observations. Figure 3.5 shows the trellis structure of *forward* that calculates the partial probabilities  $\alpha_t(S_j)$  at each state. Each partial probability represents the probability of being in hidden state  $S_j$  after seeing the observation  $o_t$  up to time  $t$ . The formal expression is:

$$\alpha_t(S_j) = Pr(o_1 o_2 \dots o_t, S_t = S_j) \quad (3.22)$$

This joint probability calculation has recursive derivation relations between previous partial probabilities by summing over all possible paths that lead to the current node

$$\alpha_t(S_j) = \sum_{i=1}^N \alpha_{t-1}(S_i) A(S_i, S_j) B(S_j, o_t) \quad (3.23)$$

where  $\alpha_{t-1}(S_i)$  is the forward probability of the previous path up to time  $t - 1$ ,  $A(S_i, S_j)$  is the transition probability from previous state  $S_i$  to current state  $S_j$ , and  $B(S_j, o_t)$  is the probability of emission of the appropriate symbol  $o_t$  in hidden state  $S_j$ . In order to start the recursion calculation of this algorithm, we need to use initial distribution probability at the base step:

- **Base Step:**

$$\alpha_1(S_j) = \pi(S_j)B(S_j, o_1) \quad 1 \leq j \leq N \quad (3.24)$$

- **Recursion Step:**

$$\alpha_t(S_j) = \sum_{i=1}^N \alpha_{t-1}(S_i)A(S_i, S_j)B(S_j, o_t) \quad 1 \leq j \leq N, 1 < t \leq T \quad (3.25)$$

Each step of the recursion involves multiplying by a  $N \times N$  matrix. The total cost of the computation of *forward* algorithm for the whole chain  $T$  is  $\mathcal{O}(N^2T)$ . There is another kind of dynamic programming algorithm that is called *backward* algorithm. The structure of this procedure works as *forward* algorithm but rather the processing is started from the last of the sequence and it flows until the initial step.

### Backward Calculation

This algorithm is the time-reversed version of the *forward* algorithm. It can be applied when we need to find the probability of observing all future events  $o_{t+1:T}$  while being in state  $S_t$  at time step  $t$ . While, the *forward* calculation is a joint probability, the *backward* variable is a conditional probability:

$$\beta_t(S_j) = \Pr(o_{t+1}o_{t+2} \dots o_T | S_t = S_j) \quad (3.26)$$

The *backward* flow can be calculated inductively in a similar manner as *forward* calculation whereas working from right to left through the trellis as shown in Figure 3.6. This algorithm consists of the following steps:

- **Base Step:**

$$\beta_1(S_j) = 1 \quad 1 \leq j \leq N \quad (3.27)$$

• **Recursion Step:**

$$\beta_t(S_j) = \sum_{i=1}^N \beta_{t+1}(S_i) A(S_j, S_i) B(S_i, o_{t+1}) \quad 1 \leq j \leq N, 1 \leq t \leq T \quad (3.28)$$

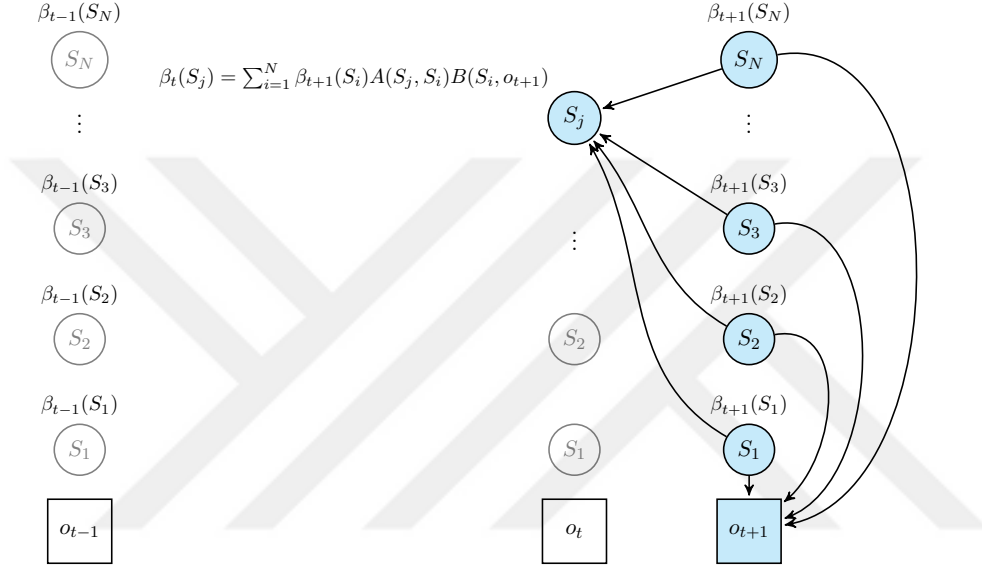


Figure 3.6: The induction step of the *backward* procedure that proceeds by summing all the future  $\beta_{t+1}(S_i)$  values with weighted state transition probability and future observation probability.

More generally, the combination of *forward* and *backward* variables can be used to calculate the probability of an observation sequence as follow.

$$\begin{aligned} Pr(O, S_t = S_j) &= Pr(o_1 o_2 \dots o_T, S_t = S_j) \\ &= Pr(o_1 \dots o_t, S_t = S_j, o_{t+1} \dots o_T) \\ &= Pr(o_1 \dots o_t, S_t = S_j) \times Pr(o_{t+1} \dots o_T | o_1 \dots o_t, S_t = S_j) \\ &= Pr(o_1 \dots o_t, S_t = S_j) Pr(o_{t+1} \dots o_T | S_t = S_j) \\ &= \alpha_t(S_j) \beta_t(S_j) \end{aligned} \quad (3.29)$$

In the third step of the Equation 3.29 *d-separation* is applied where the part of the observation sequence  $o_1 \dots o_t$  is conditionally independent with the future observation sequence

$o_{t+1} \dots o_T$  conditioned on the hidden state  $S_t = S_j$ . Therefore, the total probability of the observation sequence is:

$$Pr(O) = \sum_{i=1}^N \alpha_t(S_i) \beta_t(S_i) \quad 1 \leq t \leq T + 1 \quad (3.30)$$

The complexity of this algorithm has the same cost as *forward* algorithm, that is  $\mathcal{O}(N^2T)$ .

### Decoding Problem: Finding the best state sequence

The second problem is regarding determining the best sequence of hidden variables that are the underlying source of some observed data. In other words, finding the best hidden state sequence that explains the given observation sequence. The most common decoding algorithm used in *HMMs* is *Viterbi* algorithm which is also a kind of dynamic algorithm. Note that, a contributed algorithm is being used for the comparison of the performance of the models between *HMM* and *HP1CA* in the implementation section. Formally, the algorithm calculates

$$v_t(S_j) = \max_{S_1 \dots S_{t-1}} Pr(S_1 \dots S_{t-1}, o_1 \dots o_t, S_t = S_j) \quad (3.31)$$

The value of *Viterbi* is represented as  $v_t(S_j)$  where each cell of trellis represents the probability that the model is in state  $S_j$  after seeing the observation at time step  $t$  and the best probable state sequence  $S_1 \dots S_{t-1}$  has been already passed through to reach the hidden state  $S_j$ . This algorithm is identically the same as *forward* algorithm, but it takes the maximum value over the previous path probabilities instead of summing over all probabilities of the previous path.

$$v_t(S_j) = \max_{i=1}^N v_{t-1}(S_i) A(S_i, S_j) B(S_j, o_t) \quad (3.32)$$

The *Viterbi* algorithm has one extra component that is called *backtrace* and the corresponding variable is represented as  $bt_t(S_j)$ . This variable records the most probable path reaching the current node.

The formal definition of the recursion of this dynamic programming algorithm is as follows:

- **Base Step:**

$$\begin{aligned} v_1(S_j) &= \pi(S_j)B(S_j, o_1) & 1 \leq j \leq N \\ bt_1(S_j) &= 0 & 1 \leq j \leq N \end{aligned} \quad (3.33)$$

- **Recursion Step:**

$$\begin{aligned} v_t(S_j) &= \max_{i=1}^N v_{t-1}(S_i)A(S_i, S_j)B(S_j, o_t) & 1 \leq j \leq N, 1 < t \leq T \\ bt_t(S_j) &= \operatorname{argmax}_{i=1}^N v_{t-1}(S_i)A(S_i, S_j)B(S_j, o_t) & 1 \leq j \leq N, 1 < t \leq T \end{aligned} \quad (3.34)$$

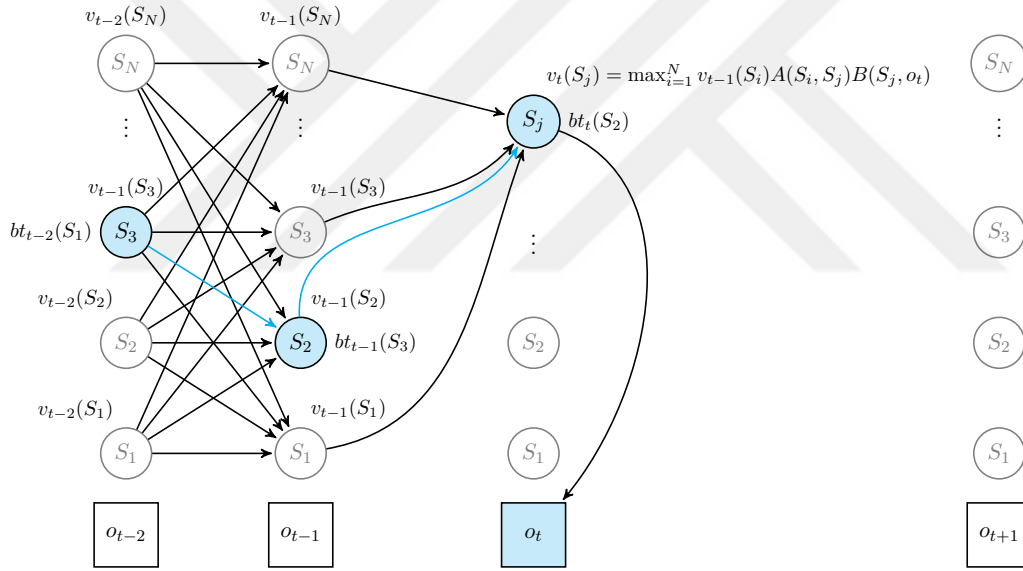


Figure 3.7: The Viterbi computation and the backtrace keeps the most probable path that leads to the state.

### Learning Problem: Parameter estimation

If a data set  $O = (o_1, o_2, \dots, o_T)$  is observed then we would like to determine the parameters of the model  $\theta = (\pi, A, B)$  which best explain the given observation sequence. The *Maximum Likelihood Estimation* function is used by marginalizing over the state variables

to the joint distribution 3.19 to find the values that maximize  $Pr(O)$  [15].

$$\theta' = \underset{\theta}{\operatorname{argmax}} Pr(O|\theta) \quad (3.35)$$

where  $\theta'$  is the new model that is found from all possible  $\theta$  to maximize the probability of the given observation  $O$  by choosing the locally maximal likelihood  $Pr(O|\theta)$ . The efficient framework to estimate the parameters of *HMM* is a kind of iterative hill-climbing algorithm which is called *Baum-Welch* algorithm [9]. It is a special case of the *Expectation Maximization*(EM) algorithm which has essentially the same solution. There are some previous work that have been studied to illustrate the advantage of *Baum-Welch* framework in *HMMs* [9, 10, 12]

- Increasing the likelihood in every iteration  $Pr(O|\theta') \geq Pr(O|\theta)$
- It converges to a local optima
- It has linear convergence

We start with no information about the model but we have a given observation sequence that we can train the model by using both transition and emission probabilities. The algorithm computes the initial estimate in which state transition and output symbol were used the most, then those estimates are used to derive a better estimate, and so on, iteratively increasing the probability of estimate until the model gives a higher probability for the given observation sequence. It is also called a convergence.

As mentioned before, *Baum-Welch* algorithm is a special case of EM that consists of two steps which are *E* step (i.e. the abbreviation of Expectation) and *M* step (i.e. Maximisation). In *E* step, we need to count a variant of used transitions and observation symbols in the model. The expectation of state transition may be expressed by applying a simple maximum likelihood estimation as:

$$\hat{A}(S_i, S_j) = \frac{\text{Number of transitions from } S_i \text{ to } S_j}{\text{Total number of transitions from } S_i} \quad (3.36)$$

The computation of the numerator is the probability of moving from state  $S_i$  to  $S_j$  while being in a particular point in the observation sequence at time  $t$ . Formally:

**Definition 3.1.2.**  $\xi$  defines the probability of being in state  $S_i$  at time  $t$  and state  $j$  at time  $t + 1$ , given the observation sequence  $O$

$$\begin{aligned}\xi_t(S_i, S_j) &= Pr(S_t = S_i, S_{t+1} = S_j | O) \\ &= \frac{Pr(S_t = S_i, S_{t+1} = S_j, O)}{Pr(O)}\end{aligned}\quad (3.37)$$

The chain rule of probability theory is applied to obtain the second line of the above equation. The numerator of Equation 3.37 is shown in Figure 3.8 where the forward calculation (i.e.  $\alpha_t(S_i)$ ) has held the probability of the observation sequence  $o_{1:t}$  and being in the state  $S_i$ . The rest of the observation sequence, i.e.  $o_{t+2:T}$ , is calculated according to backward flow (i.e.  $\beta_{t+1}(S_j)$ ) with condition on the hidden state  $S_j$ . There is only one observation symbol remaining. It is emitted by state  $j$  depending on the emission distribution  $B(S_j, o_{t+1})$ , and the last is an existing distribution of state transition (i.e.  $A(S_i, S_j)$ ) between the states  $S_i$  and  $S_j$ . There are four arguments that multiplied together compute the numerator of Equation 3.37.

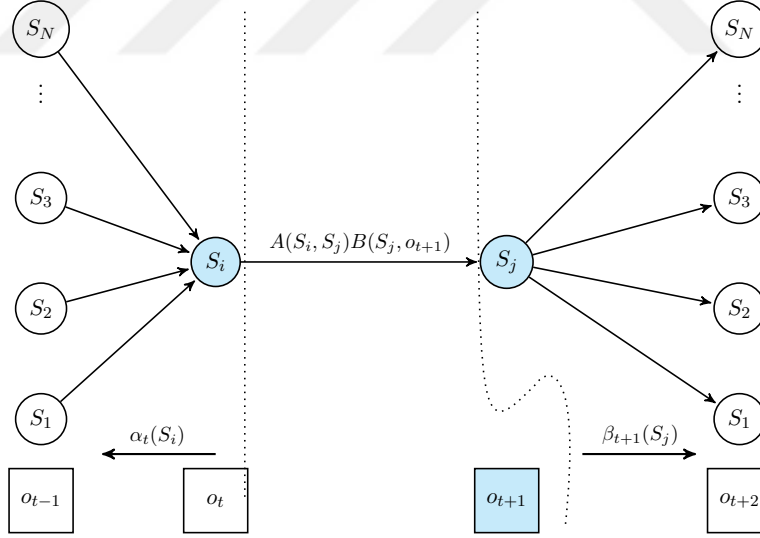


Figure 3.8: The calculation of the joint probability of being in state  $S_i$  at time  $t$  and state  $S_j$  at time  $t + 1$

The denominator of the Equation 3.37 equals to Equation 3.30 that is the probability of

whole observation sequence. Thus, the final equation form of  $\xi_t(S_i, S_j)$  is

$$\xi_t(S_i, S_j) = \frac{\alpha_t(S_i)A(S_i, S_j)B(S_j, o_{t+1})\beta_{t+1}(S_j)}{\sum_{i=1}^N \alpha_t(S_i)\beta_t(S_i)} \quad (3.38)$$

The other estimation formula we need is which provides the expected number of observation symbols that are emitted by corresponding hidden state. In  $E$  step, the expected number of the emitted observation symbol in the model could be computed as

$$\hat{B}(S_j, o_k) = \frac{\text{Number of times in state } S_j \text{ and emitting symbol } o_k}{\text{Total number of times in state } S_j} \quad (3.39)$$

where  $\hat{B}(S_j, o_k)$  is the estimated of the emission matrix by computing the probability of a given output symbol  $o_k$  (i.e.  $o_k \in V$ ) while the current hidden state is  $S_j$ . Thus, we need a new definition that illustrates the probability of being in a particular hidden state.

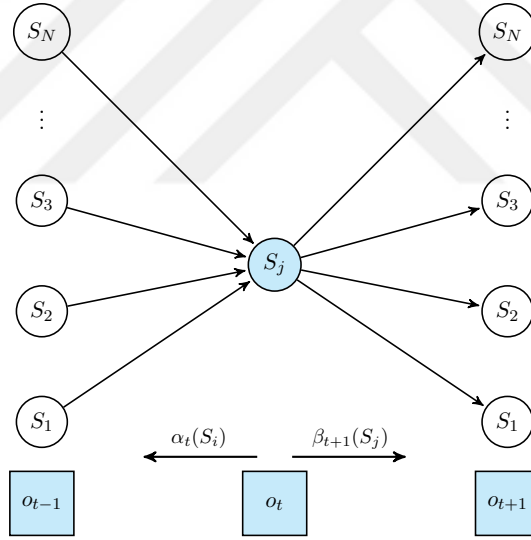


Figure 3.9: The calculation of the probability of being in state  $S_j$  at time  $t$

**Definition 3.1.3.**  $\gamma_t(S_j)$  computes the probability of being in hidden state  $S_j$  at time step  $t$ , given observation sequence  $O$

$$\begin{aligned} \gamma_t(S_j) &= Pr(S_t = S_j | O) \\ &= \frac{Pr(S_t = S_j, O)}{Pr(O)} \end{aligned} \quad (3.40)$$

As shown in Figure 3.9, the probability of the part of the whole observation sequence in which  $o_{1:t}$  and being in hidden state  $S_j$  is the calculation of the forward flow  $\alpha_t(S_j)$ . The probability of the observation sequence  $o_{t+1:T}$  conditioned on hidden state  $S_j$  is computed by the backward algorithm  $\beta_t(S_j)$ . The multiplication of the forward and backward flows constitute the numerator of equation 3.40.

$$\begin{aligned}\gamma_t(S_j) &= \frac{\alpha_t(S_j)\beta_t(S_j)}{Pr(O)} \\ &= \frac{\alpha_t(S_j)\beta_t(S_j)}{\sum_{i=1}^N \alpha_t(S_i)\beta_t(S_i)}\end{aligned}\tag{3.41}$$

We are now ready to update the parameters of the model. In  $M$  step, the estimation of the state transition matrix needs two arguments which are the sum over all the number of the state transition from  $S_i$  to  $S_j$ , and the total number of transitions comes out from state  $S_i$ . The equation 3.38 assists to obtain the updated formula for transition matrix as

$$\hat{A}(S_i, S_j) = \frac{\sum_{t=1}^{T-1} \xi_t(S_i, S_j)}{\sum_{t=1}^{T-1} \sum_{k=1}^N \xi_t(S_i, S_k)}\tag{3.42}$$

In  $M$  step, the numerator of the updated formula of the emission matrix is the sum of all time steps  $t$  of  $\gamma_t(S_j)$  in which the observation symbol  $o_t = o_k$ . The denominator of the formula is sum of all  $\gamma_t(S_j)$  for all time steps  $t$ .

$$\hat{B}(S_j, o_k) = \frac{\sum_{t=1}^T \mathbb{I}(o_t) \gamma_t(S_j)}{\sum_{t=1}^T \gamma_t(S_j)}\tag{3.43}$$

The notation  $\mathbb{I}(o_t)$  represents that the corresponding observation symbol is  $o_k$  at time  $t$  where

$$\mathbb{I}(o_t) = \begin{cases} 1 : & o_t = o_k \\ 0 : & \text{otherwise} \end{cases}$$

The updated formula of the initial distribution could be obtained by the calculation of the expected frequency in state  $S_i$  at the particular time step  $t = 1$

$$\pi(S_i) = \gamma_1(S_i)\tag{3.44}$$

In conclusion, *HMM* is an important stochastic process model that is used in various

research areas that use inference. A *Markov* chain exists behind the probabilistic process of these models; however, the states of the model are invisible and only the output distribution allow us to infer the model parameters. In other words, each hidden state emits an output symbol from the set of observation symbols which leads to generating an output data set about from the model. The probability distribution of the output symbol is stored in an emission matrix which is denoted as  $B$ . After the observation symbol is emitted, the evolution of state proceeds depending on the probability distribution of the state transition which is denoted as  $A$ . The third parameter of the model is initial state distribution which is represented by  $\pi$ . The starting state of the model is determined by the initial vector. Therefore, the probabilistic model *HMM* has three sets of probability parameters  $\theta = (\pi, A, B)$  that govern the dynamics of its *Markov* process, and that produce an observation sequence. The interest in *HMM* is that given a set of observed data that was generated by a *HMM* and then being able to compute the probabilities of the observation sequence, we are able to find the best feasible path that generates the observation sequence, and estimate the parameters of the model by doing unsupervised learning. Four fundamental algorithms that deal with those problems are *forward* and *backward* algorithms, *Viterbi* algorithm, and the last one *Baum-Welch* used for parameter estimation. The adapted version of those algorithms forms the main contribution of this thesis which is presented in Chapter 4. We adapted them to be used for *probabilistic one-counter* automata which are defined in the next section of this chapter.

## 3.2 Infinite State Models

Infinite state models (i.e. infinite-state automata) are more powerful than finite state machines and can make use of a potential unbounded memory [85, 105]. The infinite-state model is stochastic models that are finitely presentable that have probabilistic recursive behavior [106]. Thus, the structure of these systems has finite control unit along with unbounded feature such as a *pushdown* store.

Infinite-state machines are naturally applied in various of *formal verification* of systems. According to [36] there are different source of infinity such as

- **Data manipulation:** Processing data tree in which position carries a data value from some infinite domain [32, 98].

- **Control structure:** Verification problem for the systems which has unbounded stack [97, 16].
- **Asynchronous communication:** Asynchronous communication systems are automata that are adjusted with unbounded *FIFO* queues (*FIFO-first-in,first-out*) [2, 5].
- **Parametrization:** The problem of distributed systems which have an unbounded number of processes [48]
- **Real-time constraints:** Real-valued times are used to measure delays between events [3].

In particular, pushdown automata and one-counter automata are such infinite-state automata models which share a closely related underlying state transition system with the corresponding stochastic processes. Pushdown automata (i.e. *pushdown* store) is a storage-access model in which only the last item of the input can be placed on the stack which is accessed by the control unit of the model. According to the nature of the stack-based system, pushdown automata are also called *LIFO* (i.e. *LIFO-last-in,first-out*) storage-access systems due to the last item that pushed into the stack is the first item to pop out from the stack [44]. The other example of the infinite-state model is one-counter automata that are also storage-access restriction and it is the subclass of pushdown automata. The model is regarded as a pushdown automata with only a single stack letter. The value of the counter is a non-negative integer and after each step the counter value can be increased by one or decrease by one, or stay the same.

In the next sub-section, the definition of the stack-based systems (i.e. pushdown automata) that is based on *FSM* with extra memory called stack will be introduced. In section 3.2.2, the counter systems will be defined. A pushdown machine can be viewed simply as a counter machine if it has a singleton stack alphabet. And the last sub-section of this infinite models section is probabilistic one-counter machines. These systems have naturally the same structure of counter systems as well as the probability assigned to the transition function.

### 3.2.1 Stack Based Systems

In computer science, a stack-based system provides an unbounded memory where information can be added or removed according to the access manner of *last-in,first-out* (*LIFO*). The access method *LIFO* is also known as *pushdown* method that is the way of saving information of the machine which operates the transition function as a recursive evaluation of the state transition of the model while performing the stack operation. Thus, a kind of this machine always requires a stack. Pushdown automata (*PDA*) is the well-known stack-based machine and it provides also basis for *PICA* which is the second main fundamental model of this thesis.

The basic definition of *PDA* that is an *NFSA* with  $\epsilon$ -transition where it has been augmented with *LIFO* stack memory that provides an infinite amount of information as shown in Figure 3.10. The basic construction of a *PDA* contains three components which are a data input tape, a control unit and a stack. The input tape contains a given word sequence which must contains the symbols in the alphabet of the model. The tape is read in only one direction from *left-to-right* as shown in Figure 3.10. According to the transition function, the control state  $S_i$  is the member of the set of finite control states of the machine is evaluated after reading the corresponding symbol in the input tape and the machine is also able to operate to add a new symbol to the stack, pop a symbol from the stack, do both, or do no operations to the stack.

**Definition 3.2.1.** A *PDA* is specified as a 7-tuple  $\mathcal{P} = (Q, \Sigma, \Gamma, \delta, q_0, \perp, F)$  [50, 100, 104] where

- $Q$  is a finite set of control *states*
- $\Sigma$  is a finite set of *input* alphabet
- $\Gamma$  is a finite *stack* alphabet
- $\delta$  is the *transition* function where  $\delta \subseteq (Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma) \times (Q \times \Gamma^*)$ . Here,  $\Gamma^*$  represents that the finite-length strings over alphabet  $\Gamma$ .
- $q_0 \in Q$  is the *start* state of the *PDA*
- $\perp$  is the *stationary* symbol of the stack where it represents that the stack is empty when it is the only symbol in the stack.

- $F \subseteq Q$  is the set of final (i.e. accepting) states

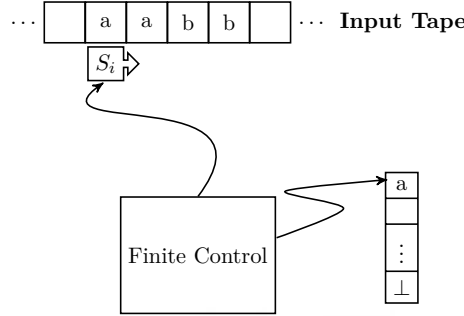


Figure 3.10: The structure of the stack-based machine

### Configuration of a *PDA*

The configuration of a *PDA* is characterized by a triple

$$(q, w, \gamma) \subseteq Q \times \Sigma^* \times \Gamma^*$$

where

- $q$  is the current control state
- $w$  is the remaining part of the input word
- $\gamma$  is the content of the stack

### Transitions of a *PDA*

The transition of a *PDA* consists of three operations. The first, machine reads one symbol at a time from the input tape. The consumed symbol is used in the transition.  $\epsilon$  is the empty string and it might be read from the input word. If it is read then the automaton does not consume any input symbol. The second, the condition of the new state might change or might be the same state as before. And the third operation is the stack operation with respect to the stack string that is labeled over the transition. There are four different stack operations that they are pop a symbol from the stack, no change to the stack (i.e. the symbol that appears at the top of the stack is the same as the previous stack symbol),

change the top of the symbol of the stack without adding or removing the symbol. And the final operation is to replace the top symbol with two or more symbols by changing the top symbol of the stack and then pushing new symbols onto the stack.

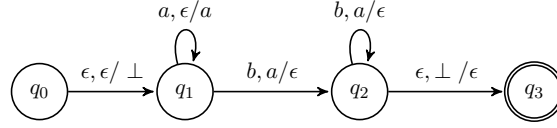


Figure 3.11: Transition diagram of a *PDA*

The transition function of the model is represented as

$$(q, \alpha w, X\beta) \rightarrow (p, w, \alpha\beta)$$

where reading the input symbol  $\alpha$ , which might be also an empty (i.e.  $\epsilon$ ) string, from the input tape and replacing the current stack symbol  $X$  by  $\alpha$ , and the state goes to  $p$  from  $q$ . Note that the remaining word  $w$  and the stack symbol  $\beta$  which is below the top of the stack influence the action of the model later [50].

The graphical notation of a corresponding transition of a *PDA* is labeled as  $a, X/Y$  in Figure 3.11 over the arc between the control states. The arc label tells that the input symbol that is consumed is  $a$  and the old top symbol of the stack  $X$  is popped out from the stack and the new stack symbol  $Y$  is pushed onto the stack.

### Acceptance of a *PDA*

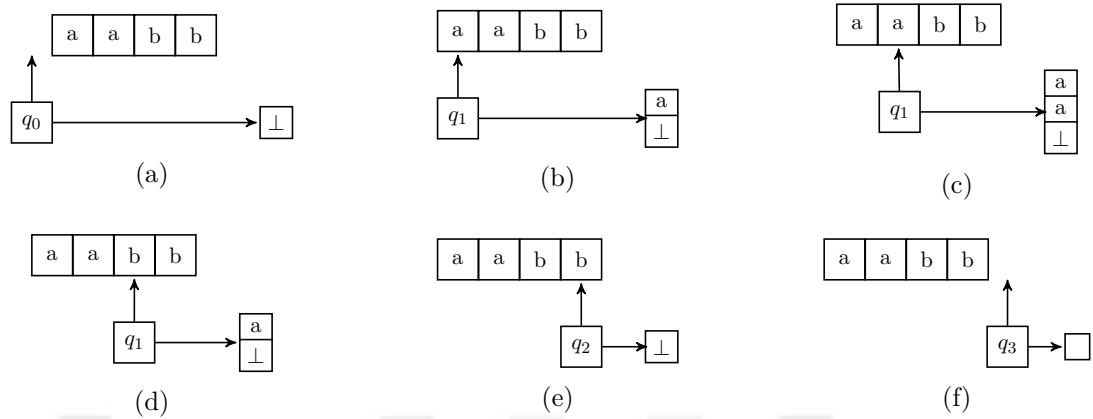
In case of a *PDA* there are two different modes of acceptance which are

- *acceptance by final state*: The machine has reached to the designated final state when the head reads the final input symbol.
- *acceptance by empty stack*: The stack of the machine is empty when the input word is finished.

According to [50], these two modes of acceptance are equivalent. Thus a *PDA* accepts a language  $L$  if and only if reaches to the final state with empty stack when the input is completely read. This acceptance method will be used for the definition of *P1CA* and some model of *HP1CA* where the model reaches the final state with counter value zero.

**Example 3.2.1.** Now, assuming that there is a given input word  $aabb$  that was generated by using the symbols in the alphabet of a *PDA* and assuming that  $x$  represents the stack alphabet of the same machine then we might represent the graphical notation of this machine as in Figure 3.11. By the following steps, the *PDA*  $\mathcal{P} = (Q = \{q_0, q_1, q_2, q_3\}, \Sigma = \{a, b\}, \Gamma = \{a, b, \perp\}, \delta, q_0, \perp, F = \{q_3\})$  accepts the given word  $aabb$ :

- At the initial step, the current control state is  $q_0$  and no consumed any input symbol, just only the initial stack symbol  $\perp$  is pushed onto the stack as shown in Figure 3.12a.
- According to the transition function, the control state moves from  $q_0$  to  $q_1$ .
- The current control state is now  $q_1$  and the first input symbol  $a$  is read by this control state. According to the transition function which is labeled over the loop arc above state  $q_1$ , without popping any symbol from the stack, a new stack symbol,  $a$ , is pushed onto the stack as shown in Figure 3.12b.
- The same process is also done while reading the second symbol of the input due to the second symbol of the input word being  $a$ . Again, the stack symbol  $a$  is pushed onto the stack as it is shown in Figure 3.12c.
- Now, the head reads the symbol  $b$  and the current state is still  $q_1$ . At this stage, the top of stack symbol is removed from the stack as shown in Figure 3.12d and the state moves from  $q_1$  to  $q_2$ .
- The last input symbol  $b$  is read by the control state  $q_2$ . According to the transition function, if the machine is in the state  $q_2$  and the symbol read is  $b$  then the corresponding top stack symbol is removed from the stack. Hereby, there is only one stack symbol, which is  $\perp$ , and it remains on the stack as shown in Figure 3.12e.
- In the end, the machine does not consume any input symbol from the tape while being in-state  $q_2$ . According to the transition function, the initial stack symbol  $\perp$  is popped and the state moves from  $q_2$  to  $q_3$ . Thus, the machine reaches the accepting state with empty stack as shown in Figure 3.12f.

Figure 3.12: The example of the word *aabb* is accepted by a *PDA*

In brief, this pushdown automaton accepts the language which could generate the string where each string starts with some number of symbols *a*, followed by the same number of symbols *b*. This language might be denoted as

$$\mathcal{L} = \{a^n b^n | n \in \mathbb{N}\}$$

In computer science, the restriction of the stack of *PDA* is able to be simulate one counter machine which is able to count the symbols in the given string. This restriction allow only for one stack symbol to be used and one special initial stack symbol.

### 3.2.2 Counter Systems

In this sub-section, we will define a fundamental model of infinite-state systems: *counter machines* (i.e. *counter automata (OCA)*) which is a pushdown automaton with only two stack symbols which are one stack symbol and the bottom of stack symbol which is never removed. The restricted stack works as a counter where the machine uses only one stack symbol to push onto the stack, this action represents the increment by one, if the machine removes the symbol from the stack that action represents the decrementation by one, and the last action is the testing the value of a counter is zero that means the stack is empty: only then the stack contains the special bottom of the symbol symbol. For example, *w* is a given word and the *counter machine* as shown in Figure 3.13 uses a stack symbol *x* to push onto the stack when the head reads the input symbol *a* in *w*, after that, the machine

removes a  $x$  from the stack when the input symbol  $b$  is read in the input word  $w$ .

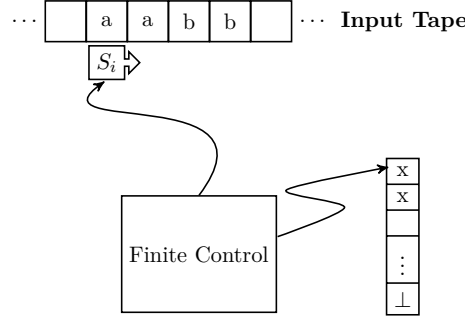


Figure 3.13: The structure of the stack-based machine

Due to their equivalent relation to the pushdown machine, we could say that a *counter* machine is a nondeterministic finite automaton with an additional memory that holds an unbounded non-negative natural number instead of a stack. At any transition, the counter can be incremented, decremented, left intact, or checked if the counter value is zero.

**Definition 3.2.2.** A *counter* machine is a 5-tuple  $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$  [22, 45, 108] where

- $Q$  is a finite set of control *states*
- $\Sigma$  is the finite set of *input* alphabet
- $\delta$  is the *transition* function as  $\delta \subseteq (Q \times (\Sigma \cup \{\epsilon\}) \times \{0, +\}) \times (Q \times \{-1, 0, 1\})$  where  $\{0, +\}$  represents the current counter value is zero or bigger than zero, respectively.  $\{-1, 0, 1\}$  denotes the counter process where the counter can be decremented, not changed, or incremented at any transition. Note that if the current counter is zero, then the operation of the counter prevents the decrementation process due to the counter value could not being a non-negative number.
- $q_0 \in Q$  is the starting state
- $F \subseteq Q$  is the set of final states
- There is a particular stack symbol  $\perp$  which is anchored at the bottom of the stack. It denotes that the counter is zero when it is the only symbol in the stack.

### Configuration of an *OCA*

A configuration of an *OCA* is a triple

$$(q, w, c) \subseteq Q \times \{-1, 0, 1\}$$

where  $q \in Q$  is a control state,  $w$  is the remaining part of the input word, and  $c \in \mathbb{N}$ .

### Transition relation of an *OCA*

The transition of an *OCA* has similar operations as the transition of a *PDA* unless no stack operation is considered. The transition relation on an *OCA* consists of three operations where it is shown as

$$(q, w, c) \rightarrow (p, c + a)$$

The machine reads a symbol from the given input tape at a time and the current control state, the input symbol, and the current counter is shown as  $(q, w, c)$  at the same time step. The state  $q$  moves to the control state  $p$  with counter changes  $c + a$ . If the current counter  $c$  is zero then  $a \in \{0, 1\}$ , if the counter  $c$  is bigger than zero then  $a \in \{-1, 0, 1\}$ . Note that a transition must provide both  $c$  and  $c + a$  that are both non-negative numbers.

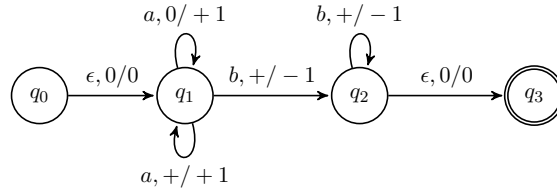


Figure 3.14: Transition diagram of a *OCA*

The transition of an *OCA* is labeled as  $a, c/c'$  over an edge in graphical notation as it is shown in Figure 3.14. In here,  $a$  is the input symbol,  $c$  is the current counter value, and  $c'$  is the updated counter value. The automaton might read an empty string  $\epsilon$  in the input word. If there is a transition in this case, the counter value will not be changed.

$$(q, \epsilon, c) \rightarrow (p, c)$$

### Acceptance of an *OCA*

The particular counter value is zero at the initial step of an *OCA* and the current state is the starting state  $q_0$ . Also, the counter value of a final state is zero at the last step. Thus, an *OCA* accepts a language if and only if reaches the final state with zero counter value when the end of the input string is reached.

**Example 3.2.2.** In this example let us use the same language as used in Example 3.2.1 in which  $\mathcal{L} = \{a^n b^n | n \in \mathbb{N}\}$ . Assume that the same word,  $aabb$ , is generated by the language  $\mathcal{L}$ . According to the transition diagram, which is shown in Figure 3.14, the given word is accepted by that counter machine in the following steps:

- At the initial step, we are in the starting state  $q_0$  and the current counter equals to zero. Without consuming an input symbol the state moves from  $q_0$  to  $q_1$  with zero counter value.
- At time step  $t = 1$ , the head reads the first symbol from the input tape. The transition  $a, 0/+1$  is enabled due to the current counter is zero. According to this transition function, we are in still  $q_1$  and the counter is incremented by one.
- At the next time step, the head again reads another  $a$  symbol from the input word. Note that the current counter value is bigger than zero, thus the transition function is now  $a, +/+1$ . That means if the counter is non-zero and the input symbol  $a$  is read, then stay in the same control state and increment the value of counter by one.
- In the third step, the input symbol  $b$  is consumed. According to the transition  $b, +/ -1$ , the state moves from  $q_1$  to  $q_2$  with decremented the counter value by one. The value of counter was two, but now it is evolved to one.
- The last input symbol  $b$  is read by the head and according to the transition  $b, +/ -1$ , we are still in the control state  $q_2$  by decrementing the counter by one. Now the counter is zero and all input string completely read.
- The last move is occurred regarding the transition  $\epsilon, 0/0$  which means that no input symbol is consumed and if counter is zero, then it is being held at zero. The state moves from  $q_2$  to accepting state  $q_3$ . Finally, the *counter* machine read the whole given input word, reached the final state with zero counter value. Thus, this word, which belongs to language  $\mathcal{L}$ , is accepted by this *OCA*.

A *OCA* has close relation with a *PDA* that both are used to model the discrete time-step of computational infinite-state systems with unbounded memory access. As is seen in Example 3.2.2, the *OCA* accepts the language which is also recognized by a *PDA*. However, either of the machines are stochastic, i.e., there is no assigned probabilistic value to the transition function of those automata. Because of this, these models are not directly relevant to our work which deals mainly with stochastic models. However, there are various works that extend these models to the probabilistic setting. One of them is called *probabilistic pushdown machine (PPDA)* and the other is called *probabilistic one-counter automata (P1CA)*. In the next sub-section, we define the second of these two models.

### 3.2.3 Definition of P1CA

In previous sub-sections, we defined non-probabilistic infinite-state models, one of them is *PDA* which has been studied for the model checking [38], and the other is *OCA*, which amount to *PDA* with only one particular stack symbol. A lot of research paper studied their relationship to other infinite-state models [66, 67]. In prior work, these non-stochastic infinite-state models have been extended with probabilistic behavior. A *probabilistic pushdown automata (PPDA)* is a probabilistic recursive procedure of a *PDA* extended by adding probabilistic transitions [39, 40]. A *P1CA* is a subclass of *PPDA* with singleton stack alphabet [19, 68].

In general, the formal definition of a *P1CA* is very close with respect to the definition of *OCA*. However, two transition functions, which are denoted as  $\delta^0$  and  $\delta^+$ , is used instead of a particular transition function  $\delta$  in *OCA*. The transition function of *P1CA* has a probabilistic value that is assigned to each transition. The value of the current counter determines which transition function can be enabled. If the counter is non-zero, then the transition relation  $\delta^+$  is enabled, and the transition function  $\delta^0$  is enabled when the counter is zero.

**Definition 3.2.3.** A *P1CA* is formally defined in the following form. A *P1CA* is 6-tuple  $\mathcal{C} = (Q, \Sigma, \delta^0, \delta^+, q_0, F)$  [46] where

- $Q$  is a finite set of control *states*
- $\Sigma$  is a finite set of *input* alphabet

- $\delta^0$  is the transition relation as  $\delta^0 \subseteq (Q \times \Sigma \times \mathbb{R}_{\geq 0}) \times (Q \times \{0, 1\})$  that is enabled when the counter value is zero.
- $\delta^+$  is the transition relation as  $\delta^+ \subseteq (Q \times \Sigma \times \mathbb{R}_{\geq 0}) \times (Q \times \{-1, 0, 1\})$  that is enabled when the counter value is non-zero.
- $q_0 \in Q$  is the starting state
- $F \subseteq Q$  is the set of final states
- There is a special bottom of stack symbol  $\perp$ . The stack acts as a counter which is non-negative, and if the top symbol of stack is  $\perp$ , then the counter is equal to zero.

### Configuration of a *PICA*

A configuration of a *PICA* has been represented in [41] as

$$\begin{aligned}\delta^0 &\subseteq Q \times \mathbb{R}_{>0} \times Q \times \{0, 1\}, \quad \text{if } c = 0 \\ \delta^+ &\subseteq Q \times \mathbb{R}_{>0} \times Q \times \{-1, 0, 1\}, \quad \text{if } c > 0\end{aligned}$$

where  $\delta^0$  is enabled when the counter value is zero, and  $\delta^+$  is enabled when the counter is non-zero.  $\mathcal{P}_{u,v}^{(c)}$  is the probability of such a transition  $(u, \mathcal{P}_{u,v}^{(c)}, v, c) \in \delta^0$ , and likewise  $\mathcal{Q}_{u,v}^{(c)}$  denotes the unique probability of transition  $(u, \mathcal{Q}_{u,v}^{(c)}, v, c) \in \delta^+$

### Transition relation of a *PICA*

Two kind of transition functions are used in a *PICA* where the counter value determines which transition function is enabled. The form of the particular transition is shown in the section of configuration of a *PICA*. If such a transition exists, then the probability of this transition is bigger than zero and it is a real number. However, if there is not such a transition then the probability of that transition is equal to zero.

The sum of the probability distribution of the transitions which come from the control state  $u$  must be one,

$$\sum_{c=0}^1 \sum_v \mathcal{P}_{u,v}^{(c)} = 1$$

, and

$$\sum_{c=-1}^1 \sum_v \mathcal{Q}_{u,v}^{(c)} = 1$$

Let  $Q = \{q_0, q_1, q_2, q_3\}$  be the set of control state of a *PICA* where the initial state is  $q_0$  and the target state is  $q_3$  as shown in Figure 3.15. There is a set of transitions which consists of the positive ones (i.e.  $\delta^+$ ) of the form  $(s, w, p, c, t)$  where  $s$  is the current state,  $w$  is the letter that is read from the input tape,  $p$  is the probability of the current transition,  $c$  is counter changing factor that is one of from  $\{-1, 0, 1\}$ , and  $t$  is the target control state.

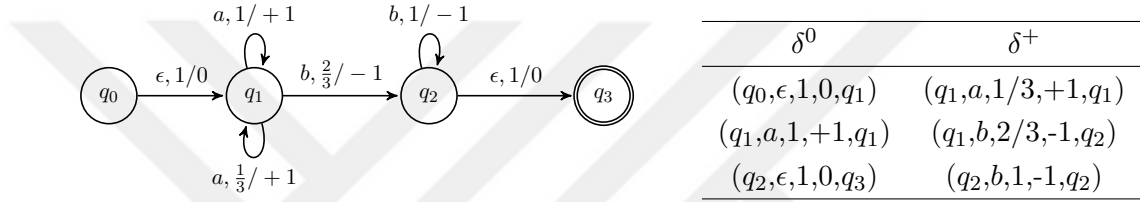


Figure 3.15: Transition diagram and set of transitions of a *PICA*

The value of  $c$  is restricted as  $\{0, 1\}$  while the counter value equals to zero. The graphical notation of each existing transition is shown as  $(w, p/c)$  over an edge as it is shown in Figure 3.15 where  $w$  is the input word,  $p$  is the probability of the current transition, and  $c$  is the value that provides the amount of the counter changing. For instance, there is a path that has a  $1/2$  probability value from state  $q_1$  which has zero counter value to the same state  $q_1$  while reading the input word  $w$  as

$$(q_1, w, 0) \xrightarrow{1/2} (q_1, c)$$

The next counter value  $c$  will be restricted to  $\{0, 1\}$ . The only difference in the presentation of a particular transition of a *OCA* is that a probability value of the current transition is labeling each arrow.

### Acceptance of a *PICA*

The acceptance of a *PICA* is similar as *OCA* accepting an input word. At the initial base, the automaton starts with a zero counter value in the initial state. At the final state, the machine reaches the accepting state with has the counter value that must be equal to zero when the machine reaches the end of the input string, otherwise, the given input is

rejected, i.e., has probability zero of being accepted.

**Example 3.2.3.** Assume that the same language  $\mathcal{L} = \{a^n b^n | n \in \mathbb{N}\}$  generated the input word  $ab$ . According to the transitions that are shown in Figure 3.15, the probability of the given word is

$$\begin{aligned} Pr(ab) &= (q_0, \epsilon, 1, 0, q_1) * (q_1, a, 1, +1, q_1) * (q_1, b, 2/3, -1, q_2) * (q_2, \epsilon, 1, 0, q_3) \\ &= 1 * 1 * 2/3 * 1 \\ &= 2/3 \end{aligned}$$

At  $t = 0$ , the we are in initial state with zero counter value. There is only one path from starting state  $q_0$  to the next state  $q_1$  where the probability of the existing path is equal to one. At the next time step  $t = 1$ , the first initial word is read from the given input tape by the machine while in state  $q_1$  and the counter value is still zero. The transition function  $\delta^0$  is enabled due to the current counter value being zero. In this case, when the first letter,  $a$ , is read, one of the transition functions corresponding counter value zero or non-zero will be chosen. It also should be noted here that there may be more than one transition value in the same transition function. Under these circumstances, the transition value to be selected may either make the counter value non-zero in the next time steps or it may not reach the final state. If one of these situations occurs, the input tape is rejected, i.e., the probability of accepting the input tape is set to zero.

For example, if  $(q_1, a, 1/2, 0, q_1)$  transition were chosen for this example, the counter value would remain zero at time step  $t = 1$  and the same transition function (i.e.  $\delta^0$ ) would still be enabled at the next time step  $t = 2$ . As seen in the Figure 3.15, this transition function will be zeroed because there is no such transition value which is associated with the word  $b$  in  $\delta^0$ , and the probability value of this given input will be zero.

$$\begin{aligned} Pr(ab) &= (q_0, \epsilon, 1, 0, q_1) * (q_1, a, 1/2, 0, q_1) * \emptyset * (q_2, \epsilon, 1, 0, q_3) \\ &= 1 * 1/2 * 0 * 1 \\ &= 0 \end{aligned}$$

Suppose for the same example that  $n = 2$  is given. The word  $aabb$  will be produced by the

same language  $\mathcal{L}$ . The probability of this word is

$$\begin{aligned} Pr(aabb) &= (q_1, a, 1, +1, q_1) * (q_1, a, 1, +1, q_1) * (q_1, b, 2/3, -1, q_2) * (q_2, b, 1, -1, q_2) \\ &= 1 * 1/3 * 2/3 * 1 \\ &= 2/9 \end{aligned}$$

In brief, the difference of a *PICA* with respect to a *OCA* is only that it has a probability value that is assigned to the existing transitions. As it is presented in [41], a *PICA* can be easily defined as a subclass of a *PPDA* with a restricted stack.

### 3.3 Conclusion

In this chapter, we described two important models that form the basis of our work. These are *HMM* and *PICA*. A *HMM* is a state-space model that contains a probability distribution of two variable sequences which are hidden state and observation. The most familiar and used model of state-space models which is used in dynamic systems that exhibit sudden changes over time, such as the economy, is a *HMM*. It is generally expressed as an expansion of *Markov Chains* to DAG models.

In sub-section 3.1.1, *PFSM* is presented to show the relations between a *HMM* and automata theory. Essentially, a *PFSM* is a generalization of non-deterministic finite-state automata without considering the empty string. Thus, the only difference of a *PFSM* with respect to non-deterministic finite-state automata is a probability distribution used while moving to the next state. To illustrate the link between a *HMM* and *FSM*, we showed that it uses fundamental algorithms of *HMM* for parsing a string. This provides an idea to use similar algorithms for other automata models.

In sub-section 3.1.2, *HMM* is handled with all its features by its fundamental algorithms which are *forward*, *backward*, *Viterbi*, and *Baum-Welch*. The states of the model are directly observed in a *Markov Chain*, however, the states of *HMM* are hidden and only observation sequence is observed instead. Thus, what is the current state or what will be the next state are unknown. However, there is a probabilistic distribution relation on the observed output sequence which is produced by a hidden variable of the model at each time

step. The probability distribution is stored in a matrix which is called *emission matrix* denoted as  $B$ . According to the *Markov* property, the current state  $S_t$  depends only on the previous state (i.e.  $S_{t-1}$ ) but neither on the past states (e.g.,  $S_{t-2}$ ) nor on the future hidden states (e.g.,  $S_{t+1}$ ). The probability distribution of the transitions between hidden states are stored in *state transition* matrix which is denoted as  $A$ . Each hidden state of the model has an initial probability distribution that are stored in initial vector that is denoted as  $\pi$ . All these arguments constitute the parameters of a *HMM*. By using these parameters, the probability of a given observation sequence can be calculated in exponential time with  $TN^{T+1}$  computational complexity, which will be too slow even for small  $T$  and  $N$  values. *Forward* and *backward* algorithms, which are dynamic programming algorithms, can be used as an alternative way to solve these computational problems in polynomial time.

There are three different calculations that calculate the likelihood problem of the given observation sequence, find the state sequence that can produce the given observation sequence with the maximum probability value, and finally, update the parameters of the model. *Forward* and *backward* algorithms are used for the first problem. The *forward* algorithm computes the observation probability by summing over the probabilities of all possible hidden state paths that could generate the observation sequence. The other kind of dynamic programming algorithm, *backward* algorithm, is the time-reversed version of the *forward* algorithm. *Viterbi* algorithm which is also a dynamic algorithm is the best way to solve the decoding problem of the *HMM*. This algorithm is used to find the best-hidden state sequence that explains the given observation sequence with maximum probability. *Baum-Welch* algorithm, which is a special case of the *Expectation - Maximization* algorithm, is the efficient framework to estimate the parameters  $\pi, A, B$ . The algorithm iteratively adapts the parameters of the model to increase the likelihood of generating the set of give observations until converging at a local optima.

In sub-section 3.2, we studied another infinite-state model *pushdown* automata (PDA) that provides infinite-state memory by adding an unbounded memory to a finite state machine. It is a stack-based system where it provides an unbounded memory where adding or removing the information from the stack with *last-in, first-out* manner access method. PDA accepts the language that is generated by a *context-free* grammars. The counter machine which can be obtained by allowing for just a single stack symbol is also considered as another infinite-state model. Adding the stack symbol onto the stack represents incre-

menting the counter, removing the top symbol of the stack illustrates the decrementation. This type of the automata is called *one-counter* automata (*OCA*). These machines are the subclass of the *PDA* and the languages that are accepted by the *PDA* are also accepted by *OCA*. The important model that is the second main model to this thesis is *probabilistic one-counter* automata (*PICA*) which is the probabilistic version of *OCA* by assigning a probability to the transition function. These machines accept the same languages which are subclass of stochastic context-free languages.

In conclusion, finite-state models of such systems often use variants of probabilistic automata whose underlying semantics is defined in terms of homogeneous Markov chains, which are also called “ fully probabilistic transition systems ” in this context. There is a Markovian models and the infinite models can be combined as presented in this chapter. In the light of this, *hidden probabilistic one-counter* automata (*HPICA*) will be defined next that illustrate that all algorithms can be adopted in order to reduce the cubic computational complexity of learning problem for *probabilistic one-counter* languages to quadratic complexity.

## Chapter 4

# Model-1: One initial terminal and one final terminal in a HP1CA

In this chapter, we represent a new model called *Hidden Probabilistic 1-Counter Automata(HP1CA)* which constitutes a new methodology for stochastic counter machines. In this model, as in *HMM*, not only states are invisible but also counter variables cannot be observed and each state has a probability distribution of producing output symbols at each discrete-time step. Thus, the structure of this model depends on the combination of two stochastic model which are *HMM* and *P1CA*.

The new model exhibits a new approach to stochastic counter machine however inter-state transition is invisible due to the control states and the counter values are hidden. In this context, we can only make an inference about which state we are in and feasible counter value of that state at a particular time step. To make this inference, we must know the visible output symbol sequence, probability distribution of state transition, and emission probability that is the probability distribution of an observation symbol that is produced by a hidden state.

The transition functions  $\delta^0$  and  $\delta^+$  are used for the transition relation in a *P1CA* as it is defined in Section 3.2.3. The corresponding transition function can be enabled depending on the counter value is zero or non-zero, respectively. In the same manner, there are two state transition matrices which are  $A^0$  and  $A^+$  are used in *HP1CA* instead of the transition function  $\delta^0$  and  $\delta^+$ . Furthermore, the model uses the observation(i.e. emission) matrix  $B$

that holds the probability distribution of observation symbols which are emitted by each of the hidden control states. The model's parameters closely reassemble the ones of *HMM*. Only initial vector is not used in this type and one extra transition matrix (i.e.  $A^+$ ) is used. The adopted version of the fundamental algorithms, which are used for *HMM*, are used to find the likelihood of a given observation sequence, and they provide quadratic computation complexity for estimating the parameter (i.e. learning problem) of *HP1CA*.

This chapter is organized as follows. In Section 4.1, the formal definition of *HP1CA* is presented with all parameters that are used for the inference process of the model. In the next section, the definition of adopted version of *forward*, *backward*, and *Baum-Welch* respectively are mentioned that include all definitions, theorems and proofs. In the end, the conclusion of this chapter briefly discusses Model-1.

## 4.1 Formal Definition of HP1CA

*HP1CA* is a discrete-time stochastic model which is *P1CA* but states are emitting an output symbol as in *HMM* and counter values are hidden. A *P1CA* has just one stack letter which determines if the stack performs a counter change in response to reading an input letter. The transition functions  $\delta^0$  and  $\delta^+$  are enabled according to the value of counter equals to zero or not, respectively. However, since *HP1CA* contains the structure of *HMM*, then it uses transition matrices  $A^0$  and  $A^+$ , which are storing the transition distribution probabilities, instead of transition functions. These matrices are enabled with respect to the counter value where it is zero or non-zero, respectively. Lastly, the output symbol (i.e. observation) is associated with all hidden states. In other words, all hidden states can produce an observation symbol with respect to a probability distribution of the emission. Those probabilities are held in  $B$  which is called observation (i.e. emission) matrix. Note that this model has only one specific starting state and one specific final state, in which both have zero counter value.

### **Formal Definition of HP1CA**

A Hp1CA  $\mathcal{C}$  consists of  $(Q, \Sigma, A^0, A^+, B, q_0, q_F)$  where

- $Q$  is the set of hidden control states
- $\Sigma$  is the space of the observation symbols

- $A^0(q', \Delta, q)$  and  $A^+(q', \Delta, q)$  are probability state transition matrices where

$$A^0 : Q \times \{0, 1\} \times Q \rightarrow [0, 1]$$

and

$$A^+ : Q \times \{-1, 0, 1\} \times Q \rightarrow [0, 1]$$

- $B(q, o)$  is observation matrix where

$$B : Q \times \Sigma \rightarrow [0, 1]$$

- $q_0 \in Q$  and  $q_F \in Q$  are starting and accepting states of the model, respectively.

#### 4.1.1 Configuration of HP1CA

As it is described, the difference of *HP1CA* with respect to *P1CA* is the state transitions are not explicitly visible. The transition probability of a *HP1CA* depends on previous hidden state, counter change value, and next hidden control state. If the counter is zero, then  $A^0$  transition matrix is enabled;

$$\begin{aligned} A^0(q', \Delta, q) &= Pr(S_t = q, C_t = \Delta | S_{t-1} = q', C_{t-1} = 0), \quad q, q' \in Q \\ \sum_{q \in Q} \sum_{\Delta=0}^1 A(q', \Delta, q) &= 1 \quad \forall q' \in Q \end{aligned} \quad (4.1)$$

If the counter is non-zero, then  $A^+$  transition matrix is enabled;

$$\begin{aligned} A^+(q', \Delta, q) &= Pr(S_t = q, C_t = c + \Delta | S_{t-1} = q', C_{t-1} = c), \quad q, q' \in Q; c \in \mathbb{N}_{>0} \\ \sum_{q \in Q} \sum_{\Delta=-1}^1 A(q', \Delta, q) &= 1 \quad \forall q' \in Q \end{aligned} \quad (4.2)$$

As defined in section 3.2, the stack-based models in automaton theory read a symbol from the given input tape and proceed to the next state according to existing state transition functions. However, since this function is not used as a transition proceeding in this model, therefore, “read input symbol” term is not suitable action for this model. However, It would be more convenient to use the term “state transition matrix” as used in *HMM*.

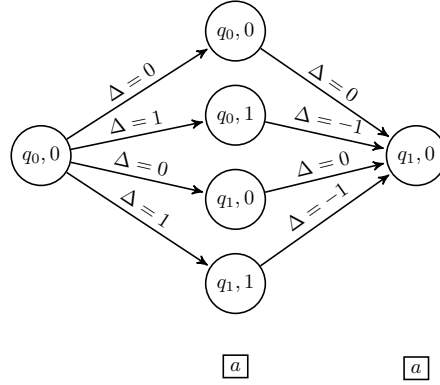


Figure 4.1: A simple example of Hp1CA

If we look closely at the parameter of the state transition matrices,  $q'$  and  $q$  represent the previous  $S_{t-1}$  and current  $S_t$  states, respectively. However, it should be noted that the counter value must be considered in *HP1CA*. So, we show both the hidden control state and the counter value as a pair  $(q, c)$ . If we return to the structure of state transition, let assume that we have a *HP1CA* as shown in Figure 4.1 to demonstrate the counter changing rate of  $\Delta$ . In the model, there are two hidden control states  $Q = \{q_0, q_1\}$  and only one observation symbol  $\Sigma = \{a\}$ . The initial state is  $q_0$  with zero counter value at time step  $t = 0$ . There are four different existing paths coming from the initial state which depend on the counter changing rate of  $\Delta$ . If the current counter is zero, then the next counter value of a particular hidden control state should be zero or one according to  $\Delta \in \{0, 1\}$  at time step  $t = 1$ . Assuming that the final state is  $q_1$  and all previous hidden state and counter pairs could reach to the final state, however,  $\Delta$  can equal different set through depending on the counter value of the previous hidden state. For instance, if we are in the particular hidden control state  $S_2 = q_1$  and the counter value is  $C_2 = 0$  (i.e. the simple notation is  $(q_1, 0)$ ) at time  $t = 2$ . There are two hidden control states, which has non-zero counter value that reach the final states, which are  $(q_0, 1)$  and  $(q_1, 1)$ . In here, the value of  $\Delta$  must equal to the set of  $\{-1, 0, 1\}$  due to the counter value being bigger than zero. If we return to the example, then the value of  $\Delta$  must be -1 for decrementing the counter value by one and to reach to  $(q_1, 0)$ .

It turns out that the structure of *HP1CA* system is close to *HMM*. We assume that in this case there is each hidden control states produce a symbol instead of reading a sym-

bol from the input tape. We shall, therefore, propose a process on output symbol to get a correspondence with *HP1CA*. As a result, we discover that these process on output symbol specification is generated by each hidden control variables of the model by considering a probabilistic distribution. We henceforth show that the emission matrix  $B$  is also used as another parameter of the model.

$$\begin{aligned} B(q, o) &= Pr(o_t = o | S_t = q), \quad q \in Q; o \in \Sigma \\ \sum_{o \in \Sigma} B(q, o) &= 1 \quad \forall q \in Q \end{aligned} \tag{4.3}$$

It is worth emphasizing that producing a symbol depends only on the hidden state, not the counter value. Hence, the emission matrix is associated with only on the hidden control states. As it is seen in Figure 4.2, computation consists of repeatedly generating any output symbol regardless of the counter value. The production of a symbol is indicated by cyan colored arrows. When it comes to determining whether or not to generate an observation sequence there are two rules: terminate in the final terminal and the counter value is zero. To formalize the intuitive behavior of the transition system of *HP1CA*, we must consider the counter value.

#### 4.1.2 Transition relation of HP1CA

The model proceeds as a stochastic process with conditional probability distribution over latent variables including counter changes for each transitions and all states are emitting an output letter  $o \in \Sigma$  at time  $t$ . The prediction of state transition is a Markovian that the current hidden control state  $S_t = q$  depends only on the previous hidden control state  $S_{t-1} = q'$ . The counter changes  $c' \xrightarrow{\Delta} c$  where  $C_{t-1} = c'$  is previous counter and  $C_t = c$  (i.e.  $c = c' + \Delta$ ) is the updated counter value after state transition is done. The counter changing rate of  $\Delta$  depends only on the previous counter to enable the changing rate sets. If the previous counter is zero, then  $\Delta \in \{0, 1\}$ , and if the previous counter is non-zero, then  $\Delta \in \{-1, 0, 1\}$ . At each existing transition, the model might emit different observation symbol according to the probability distribution of emission at each discrete time  $t$ , regardless of the counter value.

For instance, let assume that the observation sequence (*accbdadb*) is shown in Figure 4.2. The set of hidden control variables is  $Q = \{q_0, q_1, q_2, q_3\}$ , and the alphabet of the

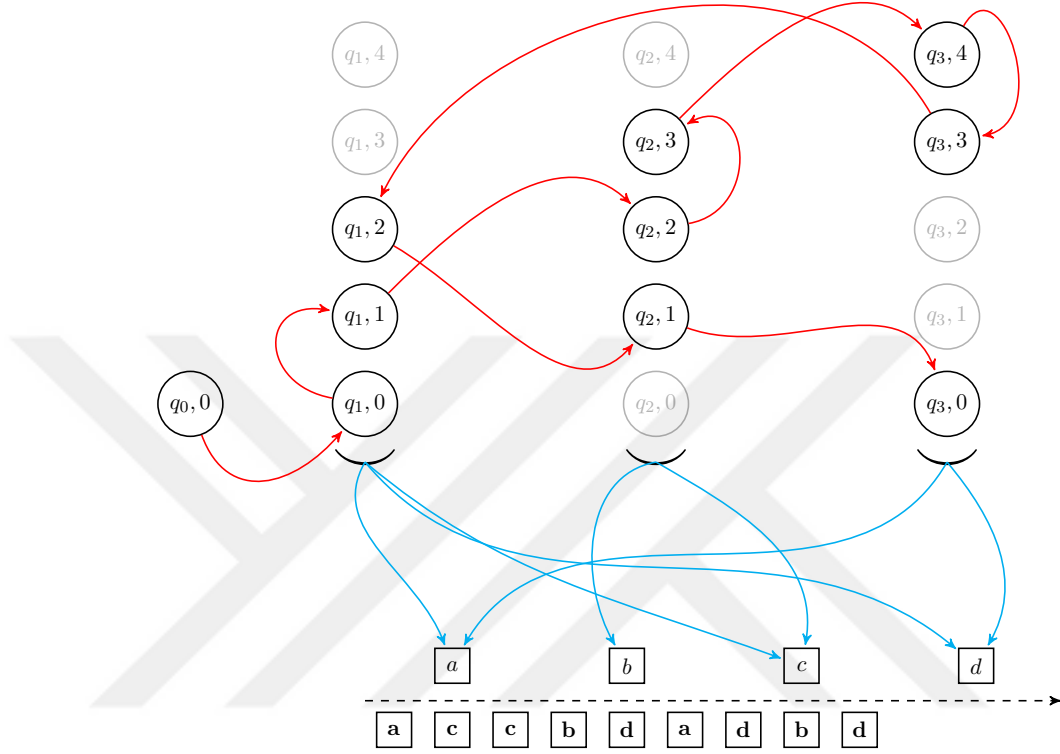


Figure 4.2: The output sequence that has been produced by the most likely sequence of hidden states and counter pair

model is  $\Sigma = \{a, b, c, d\}$ . The initial terminal is  $q_0$  and the final terminal of the model is  $q_3$  where both terminals have zero counter value. At the initial step, we are in the hidden control state  $q_0$  with zero counter value. There is only one existing transition from  $q_0$  to  $q_1$  with the same counter value (i.e. zero), and then emit a corresponding observation symbol (i.e. the symbol  $a$  is emitted) depending on the emission probability distribution. According to the first order of Markov property, the evaluation of the hidden control state and the counter value conditionally depends on the previous hidden state and counter pair as  $Pr(S_t = q_1, C_t = 0 | S_{t-1} = q_0, C_{t-1} = 0)$ . In here, the updated counter value  $C_t$  is actually equal to the sum of the previous counter value  $C_{t-1}$  and the delta value  $\Delta$  where  $C_t = C_{t-1} + \Delta$ . According to this transition, the delta value is zero. If we return to the example, the new hidden state is currently  $q_1$  with zero the counter value at time step

$t = 1$ . After the transition, this state drew the observation symbol  $a$ . In the next step (i.e.  $t = 2$ ), the hidden state does not change, but the counter value incremented one and the hidden state  $q_1$  emits the output symbol  $c$ . Now the counter value is non-zero and at the next time step  $t = 2$ , with respect to  $\Delta \in \{-1, 0, 1\}$  the counter value can be incremented or decremented or may remain the same. The same transition processing is performed until to reach the accepting state.

It is technically convenient to assume that expanding the computation of the joint probability of *HMM* as shown in Equation 3.19 by considering a particular counter sequence for *HP1CA*. The computation of the joint probability is

$$\begin{aligned} Pr(O, Q, C) &= Pr(O|Q, C) \times Pr(Q, C) \\ &= \prod_{t=1}^T Pr(o_t|S_t) \times \prod_{t=1}^T Pr(S_t, C_t|S_{t-1}, C_{t-1}) \end{aligned} \quad (4.4)$$

The first line of the equation 4.4, the sequence of counter is ignored due to the observation sequence does not depend on counter values. The first argument of the second line of the equation 4.4 is emission probability, and the second element is the state transition probability considered with the counter value. In the introduction, we have already mentioned that there are two different state transition matrices that are able to be enabled according to the corresponding counter value. If we need to plug the arguments: emission matrix and transition matrix into the equation, then we can obtain the following distribution equation

$$Pr(O, Q, C) = \prod_{t=1}^T B(S_t, o_t) A^{\delta(C_{t-1})}(S_t, C_t - C_{t-1}, S_{t-1}) \quad (4.5)$$

where

$$\delta(C_{t-1}) = \begin{cases} 0, & C_{t-1} = 0 \\ +, & C_{t-1} > 0 \end{cases}$$

In *P1CA*, the machine outputs the probability value of the input word if it is accepted. This idea leads to a transformation that the computation of the probability of the particular observation sequence  $O$  is calculated by summing over all state and counter sequences for

also *HP1CA*.

$$\begin{aligned} Pr(O) &= \sum_Q \sum_C Pr(O, Q, C) \\ &= \sum_Q \sum_C Pr(O|Q)Pr(Q, C) \end{aligned} \quad (4.6)$$

For the simple example that is shown in Figure 4.1 we would sum over

$$\begin{aligned} Pr(aa) &= Pr(aa, (q_0, 0)(q_1, 0)) + Pr(aa, (q_0, 1)(q_1, 0)) \\ &\quad + Pr(aa, (q_1, 0)(q_1, 0)) + Pr(aa, (q_1, 1)(q_1, 0)) \end{aligned} \quad (4.7)$$

Let assume that the length of the observation sequence is  $T$ , and the maximum number of the counter value is  $M$ . Furthermore, let assume that the number of the hidden control state is  $N$ . The calculation of the probability of the observation sequence requires  $TN^{T-1}M$  multiplications and additions. Note that the last observation symbol must be emitted by the final state of the model. Thus, the rest of the observation sequence could require  $N^{T-1}$  hidden control state multiplications. However, this computation is infeasible, and we need more efficient calculation to solve this problem. As it is handled in Section 3.1.2, there are two *dynamic programming* algorithms used to reduce the computational requirements for this kind of problem. The adapted version of these algorithms is the more efficient way for the calculation of the probability of the observation sequence in a *HP1CA*.

On the other hand, when considering a probabilistic counter machine as a language acceptor, it is a known fact that *One-Counter Languages (OCLs)* are the subclass of the *Context-Free Languages (CFLs)*. Ambiguity could be a nuisance for delicate stochastic grammar applications in the selection of probable derivation from between a large number of parse trees. The selection of parsing is naturally based on a suitable probability score that corresponds to the multiplication of all production rules in the parse tree. Obtaining a better probability score the probabilities of the production rules could re-estimate by using the *Inside-Outside* algorithm. However, this algorithm has a cubic time  $O(N^3)$  complexity in the size of the input and they could be costly for big input data. On the other hand, the adapted version of the *forward-backward* algorithm provides  $O(N^2)$  computational complexity in the size of the input for regular grammars.

This idea leads us to adopt the *Forward-backward* algorithm to handle feasible computational savings more than the *inside-outside* algorithm. In this context, we present a different approach by adopting the learning algorithm of *HMM* to estimate the parameters of *HP1CA*. First, individually *forward* and *backward* algorithms are adapted corresponding each possible state-counter pair throughout the given observation sequence. Then, they are used in an iterative process of *Baum-Welch* algorithm where first to find out the expected numbers of *E-step* and then to maximize the parameters in *M-step*. The time complexity of both algorithms is quadratic in the size of input.

## 4.2 Adaptation of Algorithms

In this section, the adapted version of *forward*, *backward* and *Baum-Welch* algorithms will be represented how they are plugged into the model *HP1CA* by considering the counter value. The original version of those algorithms is used for likelihood and learning problems on *HMM*, respectively, without considering the counter value. The algorithms practice the aspects of dynamic programming to make an inference by computing the given model parameters and observation sequence.

There are two designated terminals from the finite set of hidden states of this model. These terminals are initial terminal which is represented by  $S_0$  with zero counter value at time step  $t = 0$ , and the other is the final terminal which is denoted by  $S_T$  with zero counter value at time step  $t = T$ . In the light of this information, two dynamic programming algorithms, *forward* and *backward*, will be processed into two phases which are called: *preliminary* and *normalisation*. The first phase of the computation of both algorithms will be performed in a normal period as on *HMM* but considering counter values and the terminals. In the second phase, considering the specified terminals, unnecessary paths are discarded from the trellis diagram and the probabilistic values of the paths connecting both terminals to each other are normalized.

The adopted version of *Baum-Welch* algorithm is used to update the parameters of the *HP1CA* but of course the counter value is considered. This algorithm consists of two different parts represented by  $\xi$  and  $\gamma$  likewise in *HMM*. These arguments, respectively, hold the probability of the expected transition probability from each hidden state-counter pairs and the probability of being in a particular hidden state-counter pairs according to the

given observation sequence.

In subsection 4.2.1, the adapted version of the *Forward* algorithm will be introduced with the definition of both two phases and their inductive proofs. In the next sub-section, the second main algorithm that is adopted *Backward* algorithm will be defined. In the last sub-section, examining the adapted version of the estimation algorithm *Baum-Welch* will be discussed with explaining the expectation step and maximization step where the Lagrange multiplier is used to obtain the update formulas for the model.

### 4.2.1 Adapted Forward Algorithm

The Forward algorithm is used as a kind of dynamic programming algorithm on *HMM* to compute the probability of given observation sequence  $(o_{1:T})$  by summing all possible hidden state paths which could generate the given observation sequence. As the countered process of *HP1CA*, we shall consider the counter value while transforming this algorithm. There are two phases for calculation which are *The First Phase*: preliminary *forward* probabilities that is denoted by  $\hat{\alpha}$  and *The Second Phase*: normalisation of *forward* calculation that is represented by  $\alpha$ .

As mentioned before, there are two specific terminals which are  $S_0$  and  $S_T$ . The branching starts in the initial terminal and it ends in the final state when the last output symbol is emitted. With respect to this knowledge, a simple directed acyclic graph is obtained at the end of the time step  $T$ . Therefore, some particular paths are not able to bound to this branching and they lose their significance. At this point, we need to calculate the stationary paths by using the normalization phase.

#### ***The First Phase: Preliminary Probability of Adopted Forward Algorithm***

In this phase, we calculate the joint probability distribution of given observation sequence  $o_{1:t}$  and hidden state-counter pair  $S_t = (q, c)$  conditionally on the initial state  $S_0$  where

$$\hat{\alpha}_t(q, c) = Pr(o_{1:t}, S_t | S_0) \quad (4.8)$$

The algorithm processes the computing of summing of all paths that reach to the state  $S_t$  at each discrete time  $(1, 2, \dots, t)$  by considering the counter value. The calculation starts by initializing step at time  $t = 0$ . At this step, we know that where we are; at the starting state  $S_0$ . The initializing probability of preliminary calculation of adapted forward algorithm is;

$$\hat{\alpha}_0(q, c) = \begin{cases} 1 : & S_0 = q \text{ and } c = 0 \\ 0 : & \text{otherwise} \end{cases}$$

Dynamic programming algorithms are recursive. Therefore, in each time step, the algorithm computes and stores the likelihood of the observation by summing over the probabilities of all possible paths that are from each hidden control state-counter pairs to the next hidden control state-counter pairs. This calculation is performed with two steps which are

**Base Step:**

$$\hat{\alpha}_1(q, c) = A^0(S_0, \Delta, S_1)B(S_1, o_1) \quad (4.9)$$

**Recursion Step:**

if  $c - \Delta = 0$  :

$$\hat{\alpha}_t(q, c) = \sum_{q' \in Q} \sum_{\Delta=0}^1 \hat{\alpha}_{t-1}(q', c - \Delta) A^0(q', \Delta, q) B(q, o_t) \quad (4.10)$$

if  $c - \Delta > 0$  :

$$\hat{\alpha}_t(q, c) = \sum_{q' \in Q} \sum_{\Delta=-1}^1 \hat{\alpha}_{t-1}(q', c - \Delta) A^+(q', \Delta, q) B(q, o_t) \quad (4.11)$$

At the base step, the previous counter is zero therefore  $A^0$  transition matrix is enabled. There is no initial state probability vector  $\pi$  which is used in *HMM*, instead the transitions that are between from designated starting state to the next hidden states are used. For instance let a *HP1CA* has two hidden control states  $Q = \{q_1, q_2\}$  and only two observation alphabet  $\Sigma = \{a, b\}$  as shown in Figure 4.3. The initial state  $S_0 = (q_1, 0)$  is in the position of the previous state at time step  $t = 1$ . The changes counter factor  $\Delta \in \{0, 1\}$  due to the previous counter value is zero. Thus, there are four possible paths coming from initial state to

$(q_1, 0)$ ,  $(q_1, 1)$ ,  $(q_2, 0)$ , and  $(q_2, 1)$ , respectively at the time step  $t = 1$  as shown in Figure 4.4.

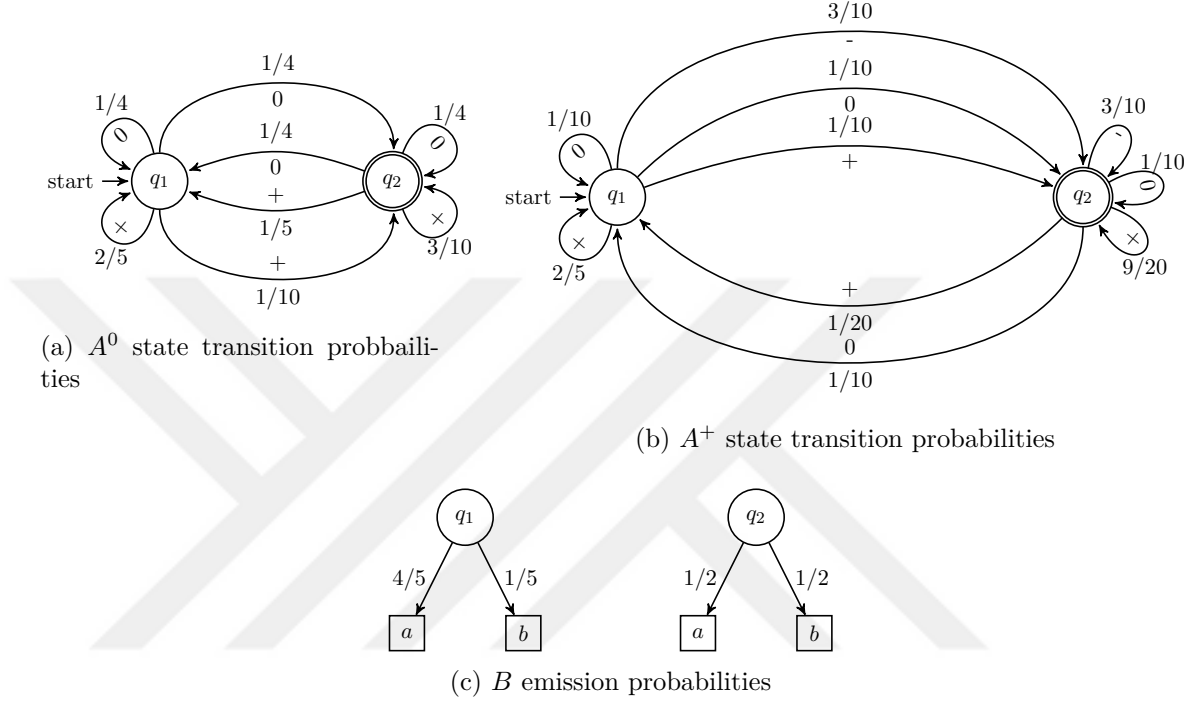


Figure 4.3: The parameters of a *HP1CA* for a simple example

The parameters of the model  $A^0$ ,  $A^+$ , and  $B$  are as shown in Figure 4.3a, 4.3b, and 4.3c. The probability of the preliminary forward flow at the base step is calculated by multiplying the state-counter transition probability and emission probability due to the previous preliminary forward value is equal to one with respect to there is only one single initial terminal as

$$\begin{aligned}
 \hat{\alpha}_1(q_1, 0) &= A^0(S_0, 0, S_1) * B(S_1, a) \\
 &= A^0(q_1, 0, q_1) * B(q_1, a) \\
 &= \frac{1}{4} * \frac{4}{5} \\
 &= 0.2
 \end{aligned}$$

In the recursion step, the summing of the multiplication of the previous preliminary forward probability, state-counter transition probability and emission probability over all the previous hidden control state as

$$\begin{aligned}
 \hat{\alpha}_2(q_1, 0) &= \left[ \hat{\alpha}_1(q_1, 0) * A^0(q_1, 0, q_1) + \hat{\alpha}_1(q_1, 1) * A^+(q_1, -1, q_1) \right. \\
 &\quad \left. + \hat{\alpha}_1(q_2, 0) * A^0(q_2, 0, q_1) + \hat{\alpha}_1(q_2, 1) * A^+(q_2, -1, q_1) \right] * B(q_1, b) \\
 &= [0.2 * 0.25 + 0.32 * 0 + 0.125 * 0.25 + 0.05 * 0] * 0.2 \\
 &= 0.01625
 \end{aligned}$$

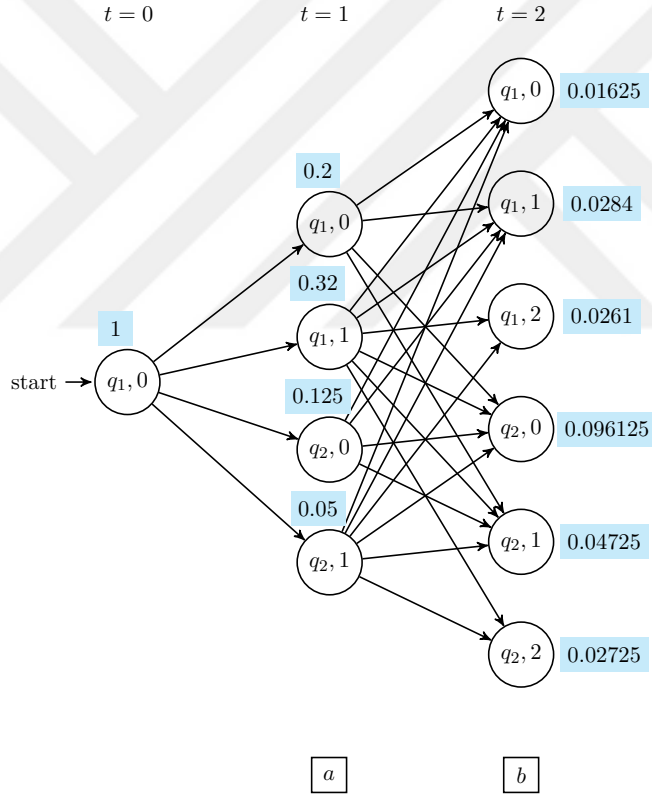


Figure 4.4: The trellis of the preliminary forward flow

In the first phase of the adapted forward algorithm, the calculation is performed as forward calculation in *HMM* but considering counter values. The likelihood of a particular hidden control state-counter pair at time step  $t$  is calculated with the condition on only initial

state  $S_0$ . The initial state is the only known point where the branching of the paths starts. This information provides the following theorem that calculates the preliminary forward flow.

**Theorem 4.2.1.**  $\hat{\alpha}_t(q, c)$  computes the joint probability condition on  $S_0$  as

$$Pr(o_{1:t}, S_t = (q, c) | S_0)$$

up to time  $t$ .

*Proof.* We will prove this equation

$$\hat{\alpha}_t(q, c) = Pr(o_{1:t}, S_t | S_0) \quad (4.12)$$

is equivalently equal to

$$\hat{\alpha}_t(q, c) = \sum_{S_{t-1} \in Q} Pr(o_{1:t-1}, S_{t-1} | S_0) Pr(S_t | S_{t-1}) Pr(o_t | S_t) \quad (4.13)$$

At  $t = 0$ , it is initial step and we know that the path will start from  $S_0$  without emitted symbol and only the hidden state which is equal to initial terminal and has zero counter value probability is 1, otherwise zero. If  $t = 0$ , then we have

$$\hat{\alpha}_0(q, c) = Pr(o_{1:0}, S_0 | S_0)$$

The observation sequence  $o_{1:0}$  is irrelevant, it can be removed, then we get

$$\hat{\alpha}_0(q, c) = Pr(S_0 | S_0) = 1$$

If and only if  $S_0 = q$  and  $c = 0$ .

At the base step  $t = 1$ ,

$$\begin{aligned} \hat{\alpha}_1(q, c) &= Pr(o_{1:1}, S_1 | S_0) \\ &= Pr(o_1, S_1 | S_0) \end{aligned} \quad (4.14)$$

Applied the chain rule of the probability theorem,

$$\hat{\alpha}_1(q, c) = Pr(o_1|S_1, S_0)Pr(S_1|S_0)$$

Here, the observation symbol  $o_1$  is conditionally independent with  $S_0$  given  $S_1$  according to *head – to – tail* rule of *d – separation*.

$$\hat{\alpha}_1(q, c) = Pr(o_1|S_1)Pr(S_1|S_0)$$

$Pr(S_1|S_0)$  represents the transition function where  $S_1$  is next state and  $S_0$  is the current hidden variable. We already know that the initial counter is zero for the initial state  $S_0$  and the transition matrix  $A^0$  is enabled due to that counter value. The next counter value might be zero or one depending on  $\Delta$ .  $Pr(o_1|S_1)$  represents the first symbol of the observation sequence is emitted by hidden state  $S_1$ .

Let assume that the algorithm provides correct calculation for  $t = k$  time step. We need to prove that the algorithm also provides the following equivalence for time  $t = k + 1$ .

$$\hat{\alpha}_{k+1}(q, c) = \sum_{S_k \in Q} Pr(o_{1:k}, S_k|S_0)Pr(S_{k+1}|S_k)Pr(o_{k+1}|S_{k+1}) \quad (4.15)$$

First, add time  $t = k + 1$  into Equation 4.12;

$$\hat{\alpha}_{k+1}(q, c) = Pr(o_{1:k+1}, S_{k+1}|S_0) \quad (4.16)$$

Applying chain rule of the probability theorem, then we have this equivalence;

$$\hat{\alpha}_{k+1}(q, c) = Pr(o_{1:k+1}|S_{k+1}, S_0)Pr(S_{k+1}|S_0) \quad (4.17)$$

According to the *d – separation* rule,  $o_{k+1} \perp\!\!\!\perp S_0|S_{k+1}$ ;

$$\hat{\alpha}_{k+1}(q, c) = Pr(o_{1:k}|S_{k+1}, S_0)Pr(S_{k+1}|S_0)Pr(o_{k+1}|S_{k+1}) \quad (4.18)$$

Applying again chain rule, then we can get;

$$\hat{\alpha}_{k+1}(q, c) = Pr(o_{1:k}, S_{k+1}|S_0)Pr(o_{k+1}|S_{k+1}) \quad (4.19)$$

Now we can apply sum rule of the probability over hidden state  $S_k$  as;

$$\hat{\alpha}_{k+1}(q, c) = \sum_{S_k} Pr(o_{1:k}, S_k, S_{k+1} | S_0) Pr(o_{k+1} | S_{k+1}) \quad (4.20)$$

Again applying chain rule;

$$\hat{\alpha}_{k+1}(q, c) = \sum_{S_k} Pr(o_{1:k}, S_{k+1} | S_k, S_0) Pr(S_k | S_0) Pr(o_{k+1} | S_{k+1}) \quad (4.21)$$

According to the  $d$  – *separation* rule,  $S_{k+1} \perp\!\!\!\perp S_0 | S_k$ ;

$$\hat{\alpha}_{k+1}(q, c) = \sum_{S_k} Pr(o_{1:k} | S_k, S_0) Pr(S_k | S_0) Pr(S_{k+1} | S_k) Pr(o_{k+1} | S_{k+1}) \quad (4.22)$$

If we apply again chain rule then we will get the equivalence of (4.15);

$$\hat{\alpha}_{k+1}(q, c) = \sum_{S_k} Pr(o_{1:k}, S_k | S_0) Pr(S_{k+1} | S_k) Pr(o_{k+1} | S_{k+1}) \quad (4.23)$$

And the proof is successfully done. □

### Second Phase: Normalisation for Adopted Forward Algorithm

In this section, the second phase of the algorithm is calculating the joint probability distribution of observation sequence  $o_{1:t}$  and the hidden state-counter pairs  $S_t = (q, c)$  given the terminals  $S_0$  and  $S_T$ . This phase is called '*Normalisation*' to normalize the probabilities of the each paths which are linked to final state coming through starting state. The partial calculation of normalisation is denoted as  $\alpha_t(q, c)$  to represent the normalization probability of  $(q, c)$  for each step  $t$ .

In this phase, there is one more knowledge regarding the final terminal where the model terminates at time  $T$ . According to this knowledge, all trivial paths that are not reached to the final terminal are removed until to the starting terminal of the model. The final terminal, which has zero counter value, is a sort of the starting point to calculate the second phase of the adopted *forward* algorithm. This calculation contains two steps which are

**Base Step:**

$$\alpha_T(q, c) = \begin{cases} \frac{\hat{\alpha}_T(q, c)}{Pr(S_T | S_0)} & : S_T = q = q_F \text{ and } c = 0 \\ 0 & : \text{otherwise} \end{cases}$$

**Recursion Step:**

**if**  $c = 0$  :

$$\alpha_t(q, c) = \sum_{q' \in Q} \sum_{\Delta=0}^1 \frac{\alpha_{t+1}(q', c + \Delta) A^0(q, \Delta, q') \hat{\alpha}_t(q, c)}{\hat{\alpha}_{t+1}(q', c + \Delta)} \quad (4.24)$$

**if**  $c > 0$  :

$$\alpha_t(q, c) = \sum_{q' \in Q} \sum_{\Delta=-1}^1 \frac{\alpha_{t+1}(q', c + \Delta) A^+(q, \Delta, q') \hat{\alpha}_t(q, c)}{\hat{\alpha}_{t+1}(q', c + \Delta)} \quad (4.25)$$

At the base step  $t = T$ , the calculation of normalisation starts in the final state with zero counter  $S_T = (q_F, 0)$ . The rest of the hidden state-counter pairs are discarded because this model ends at a designated final state at  $t = T$  as shown in Figure 4.5a. The accepting state is  $q_2$  with zero counter value in this example. The probability of the normalization of  $(q_2, 0)$ , according to the equation of the base step, is found by dividing the preliminary probability of the forward flow of  $(q_2, 0)$  with the total probability of existing all paths between the final and initial terminals.

The probability of the preliminary of the final state is  $\hat{\alpha}_2(q_2, 0) = 0.096125$  and the total probability of transitions between initial and final is calculated by summing the multiplication of the probability of all existing paths between the terminals as it is done in Figure 4.5b. The probability of this calculation is computed with used only transition matrices where the activation of matrices depends on the counter value.

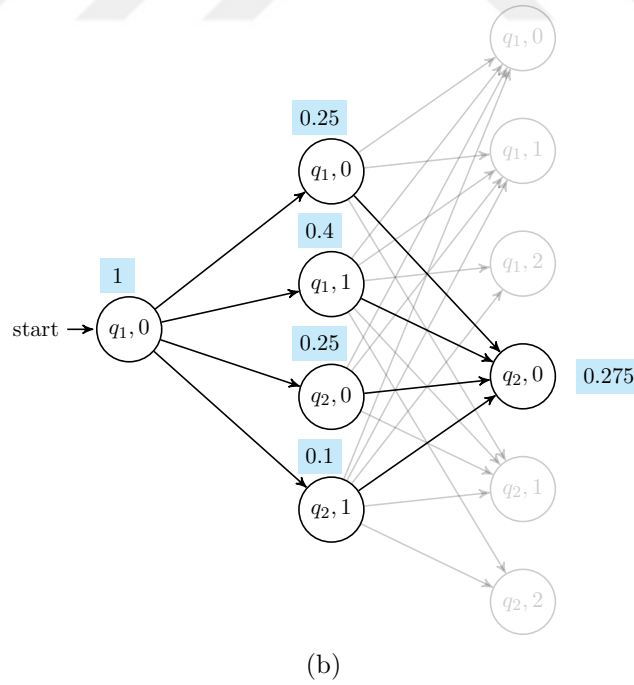
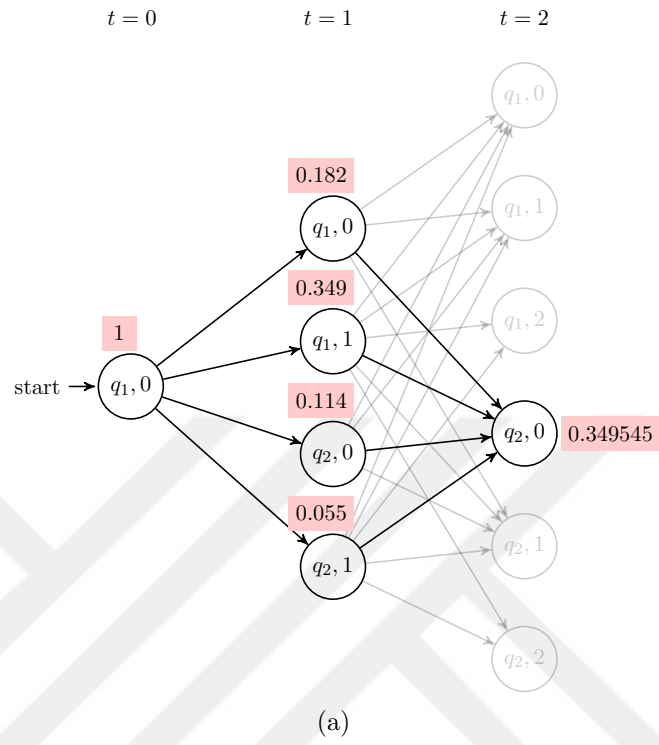


Figure 4.5: Normalisation calculation of adapted forward algorithm

At the iteration step, there are three important reminders regarding the calculation to normalize the probability of valid paths. One of them is that the calculation starts from the final state which has zero counter value, and the other is that the time step is decremented by one for each iteration. Therefore, in this phase, the normalization calculation at a particular time step  $t$  will represent the future normalization value for the calculation which is performed in next time step  $t - 1$ . And the last important reminder is that we need all the preliminary probability values of forward flow that must be already calculated in the first phase. Because these values are used to proportioning the probability of hidden variable-counter pairs that there is a valid path between them. For instance, in Figure 4.5a the calculation of normalization in next step  $t = 1$

$$\begin{aligned}\alpha_1(q_1, 0) &= \frac{\alpha_2(q_2, 0)A^0(q_1, 0, q_2)\hat{\alpha}_1(q_1, 0)}{\hat{\alpha}_2(q_2, 0)} \\ &= \frac{0.35 * 0.25 * 0.2}{0.096125} \\ &= 0.182\end{aligned}$$

The calculation of normalization is continued until it reaches the initial state or the time scale is being zero. Throughout the sequence, we need to know where this calculation begins and ends. At that point, the terminals of the model become involved to make a sense for the calculation as mentioned in the following theorem.

**Theorem 4.2.2.**  $\alpha_t(q, c)$  computes recursively the joint probability condition on  $S_0$  and  $S_T$  as  $Pr(o_{1:t}, S_t = (q, c)|S_0, S_T)$  up to time  $t$ .

*Proof.* We will prove below equation

$$\alpha_t(q, c) = Pr(o_{1:t}, S_t|S_0, S_T) \quad (4.26)$$

is equivalently equal to

$$\alpha_t(q, c) = \sum_{S_{t+1}} \frac{Pr(o_{1:t+1}, S_{t+1}|S_0, S_T)Pr(S_{t+1}|S_t)Pr(o_{1:t}, S_t|S_0)}{Pr(o_{1:t+1}, S_{t+1}|S_0)} \quad (4.27)$$

At the base step  $t = T$ ,

$$\begin{aligned}\alpha_T(q, c) &= Pr(o_{1:T}, S_T | S_0, S_T) \\ &= Pr(o_{1:T} | S_0, S_T)\end{aligned}\tag{4.28}$$

The second line of the equation 4.28 is obtained if and only if the hidden state  $q$  is the final state (i.e.  $q = q_F$ ) and the counter value  $c$  equals to zero (i.e.  $c = 0$ ). If the product rule of the probability theorem is applied, then the equation will be;

$$\alpha_T(q, c) = \frac{Pr(o_{1:T}, S_0, S_T)}{Pr(S_0, S_T)}\tag{4.29}$$

Again applying product rule to both numerator and denominator, then we will get;

$$\alpha_T(q, c) = \frac{Pr(o_{1:T}, S_T | S_0)}{Pr(S_T | S_0)}\tag{4.30}$$

The numerator of the above equation is representing the preliminary probability of adopted forward algorithm  $\hat{\alpha}_T(q, c)$ . If we look at the denominator part, it is easily seen that all linked paths between  $S_0$  and  $S_T$  are summing the multiplication of all existing paths over the latent variables without emitting observation.

Now, assuming that the equation 4.26 is true for  $t = k$ , we need to prove that the same equation can provide the correct result for  $t = k - 1$ .

At  $t = k - 1$ , we will plug it to recursion formula, then we will have

$$\alpha_{k-1}(q, c) = \sum_{S_k} \frac{Pr(o_{1:k}, S_k | S_0, S_T) Pr(S_k | S_{k-1}) Pr(o_{1:k-1}, S_{k-1} | S_0)}{Pr(o_{1:k}, S_k | S_0)}\tag{4.31}$$

We start with applying product rule

$$\alpha_{k-1}(q, c) = \sum_{S_k} \frac{Pr(o_{1:k}, S_0, S_T, S_k) Pr(S_k | S_{k-1}) Pr(o_{1:k-1}, S_{k-1} | S_0)}{Pr(S_0, S_T) Pr(o_{1:k}, S_k | S_0)}\tag{4.32}$$

Again applying product rule

$$\alpha_{k-1}(q, c) = \sum_{S_k} \frac{Pr(o_{1:k}, S_0, S_T | S_k) Pr(S_k) Pr(S_k | S_{k-1}) Pr(o_{1:k-1}, S_{k-1} | S_0)}{Pr(S_0, S_T) Pr(o_{1:k}, S_k | S_0)} \quad (4.33)$$

According to  $d$ -separation rule,  $o_k \perp\!\!\!\perp S_0, S_T | S_k$

$$\alpha_{k-1}(q, c) = \sum_{S_k} \frac{Pr(o_{1:k-1}, S_0, S_T | S_k) Pr(S_k) Pr(o_k | S_k) Pr(S_k | S_{k-1}) Pr(o_{1:k-1}, S_{k-1} | S_0)}{Pr(S_0, S_T) Pr(o_{1:k}, S_k | S_0)} \quad (4.34)$$

Applying product rule to merge  $Pr(S_k)$

$$\alpha_{k-1}(q, c) = \sum_{S_k} \frac{Pr(o_{1:k-1}, S_0, S_T, S_k) Pr(o_k | S_k) Pr(S_k | S_{k-1}) Pr(o_{1:k-1}, S_{k-1} | S_0)}{Pr(S_0, S_T) Pr(o_{1:k}, S_k | S_0)} \quad (4.35)$$

Applying chain rule to last item of the numerator

$$\alpha_{k-1}(q, c) = \sum_{S_k} \frac{Pr(o_{1:k-1}, S_0, S_T, S_k) Pr(o_k | S_k) Pr(S_k | S_{k-1}) Pr(o_{1:k-1} | S_{k-1}, S_0) Pr(S_{k-1} | S_0)}{Pr(S_0, S_T) Pr(o_{1:k}, S_k | S_0)} \quad (4.36)$$

We can merge third and fourth items of the numerator with respect to  $S_{k-1}$ .  $S_k$  is separated with observation sequence  $o_{1:k-1}$  and starting state  $S_0$ , condition on  $S_{k-1}$ .

$$\alpha_{k-1}(q, c) = \sum_{S_k} \frac{Pr(o_{1:k-1}, S_0, S_T, S_k) Pr(o_k | S_k) Pr(o_{1:k-1}, S_k | S_{k-1}, S_0) Pr(S_{k-1} | S_0)}{Pr(S_0, S_T) Pr(o_{1:k}, S_k | S_0)} \quad (4.37)$$

Applying again chain rule to last two items of the numerator for merging

$$\alpha_{k-1}(q, c) = \sum_{S_k} \frac{Pr(o_{1:k-1}, S_0, S_T, S_k) Pr(o_k | S_k) Pr(o_{1:k-1}, S_k, S_{k-1} | S_0)}{Pr(S_0, S_T) Pr(o_{1:k}, S_k | S_0)} \quad (4.38)$$

Applying chain rule to last item of the denominator

$$\alpha_{k-1}(q, c) = \sum_{S_k} \frac{Pr(o_{1:k-1}, S_0, S_T, S_k) Pr(o_k | S_k) Pr(o_{1:k-1}, S_k, S_{k-1} | S_0)}{Pr(S_0, S_T) Pr(o_{1:k} | S_k, S_0) Pr(S_k | S_0)} \quad (4.39)$$

According to  $d$  – separation rule,  $o_k \perp\!\!\!\perp S_0 | S_k$  at the denominator

$$\alpha_{k-1}(q, c) = \sum_{S_k} \frac{Pr(o_{1:k-1}, S_0, S_T, S_k) Pr(o_k | S_k) Pr(o_{1:k-1}, S_k, S_{k-1} | S_0)}{Pr(S_0, S_T) Pr(o_{1:k-1} | S_k, S_0) Pr(S_k | S_0) Pr(o_k | S_k)} \quad (4.40)$$

Eliminating each  $Pr(o_k | S_k)$  from both numerator and denominator

$$\alpha_{k-1}(q, c) = \sum_{S_k} \frac{Pr(o_{1:k-1}, S_0, S_T, S_k) Pr(o_{1:k-1}, S_k, S_{k-1} | S_0)}{Pr(S_0, S_T) Pr(o_{1:k-1} | S_k, S_0) Pr(S_k | S_0)} \quad (4.41)$$

Applying product rule to first argument of the numerator

$$\alpha_{k-1}(q, c) = \sum_{S_k} \frac{Pr(o_{1:k-1}, S_0, S_T | S_k) Pr(S_k) Pr(o_{1:k-1}, S_k, S_{k-1} | S_0)}{Pr(S_0, S_T) Pr(o_{1:k-1} | S_k, S_0) Pr(S_k | S_0)} \quad (4.42)$$

According to  $d$  – separation rule,  $S_T \perp\!\!\!\perp S_0 | S_k$  and also  $S_T \perp\!\!\!\perp o_{1:k-1} | S_k$  at the numerator

$$\alpha_{k-1}(q, c) = \sum_{S_k} \frac{Pr(o_{1:k-1}, S_0 | S_k) Pr(S_k) Pr(S_T | S_k) Pr(o_{1:k-1}, S_k, S_{k-1} | S_0)}{Pr(S_0, S_T) Pr(o_{1:k-1} | S_k, S_0) Pr(S_k | S_0)} \quad (4.43)$$

Applying product rule to first two items of numerator to merge  $Pr(S_k)$

$$\alpha_{k-1}(q, c) = \sum_{S_k} \frac{Pr(o_{1:k-1}, S_k, S_0) Pr(S_T | S_k) Pr(o_{1:k-1}, S_k, S_{k-1} | S_0)}{Pr(S_0, S_T) Pr(o_{1:k-1} | S_k, S_0) Pr(S_k | S_0)} \quad (4.44)$$

Applying chain rule to last two items of denominator to merge  $Pr(S_k | S_0)$

$$\alpha_{k-1}(q, c) = \sum_{S_k} \frac{Pr(o_{1:k-1}, S_k, S_0) Pr(S_T | S_k) Pr(o_{1:k-1}, S_k, S_{k-1} | S_0)}{Pr(S_0, S_T) Pr(o_{1:k-1}, S_k | S_0)} \quad (4.45)$$

Again applying product rule to the first item of the numerator

$$\alpha_{k-1}(q, c) = \sum_{S_k} \frac{Pr(o_{1:k-1}, S_k | S_0) Pr(S_0) Pr(S_T | S_k) Pr(o_{1:k-1}, S_k, S_{k-1} | S_0)}{Pr(S_0, S_T) Pr(o_{1:k-1}, S_k | S_0)} \quad (4.46)$$

If we look both numerator and denominator, then it is easily seen that we can eliminate

each  $Pr(o_{1:k-1}, S_k | S_0)$

$$\alpha_{k-1}(q, c) = \sum_{S_k} \frac{Pr(S_0)Pr(S_T | S_k)Pr(o_{1:k-1}, S_k, S_{k-1} | S_0)}{Pr(S_0, S_T)} \quad (4.47)$$

Product rule can be easily applied to merge first and last arguments of the numerator

$$\alpha_{k-1}(q, c) = \sum_{S_k} \frac{Pr(S_T | S_k)Pr(o_{1:k-1}, S_k, S_{k-1}, S_0)}{Pr(S_0, S_T)} \quad (4.48)$$

Applying product rule to last item of the numerator

$$\alpha_{k-1}(q, c) = \sum_{S_k} \frac{Pr(S_T | S_k)Pr(o_{1:k-1}, S_{k-1}, S_0 | S_k)Pr(S_k)}{Pr(S_0, S_T)} \quad (4.49)$$

According to  $d$ -separation  $S_T$  is separated with whole  $(o_{1:k-1}, S_{k-1}, S_0, \text{condition on } S_k)$ . We can merge these two items according to this separation. After the merge we can apply product rule to merge  $Pr(S_k)$

$$\alpha_{k-1}(q, c) = \sum_{S_k} \frac{Pr(o_{1:k-1}, S_{k-1}, S_k, S_0, S_T)}{Pr(S_0, S_T)} \quad (4.50)$$

If we apply again product rule to eliminate the denominator item then we will have

$$\alpha_{k-1}(q, c) = \sum_{S_k} Pr(o_{1:k-1}, S_{k-1}, S_k | S_0, S_T) \quad (4.51)$$

We can remove the sum rule of the probability over  $S_t$ .

$$\alpha_{k-1}(q, c) = Pr(o_{1:k-1}, S_{k-1} | S_0, S_T) \quad (4.52)$$

The equation 4.52 is equivalently equal to equation 4.26 at the time step  $t = k - 1$ .  $\square$

The time complexity of the original version of Forward algorithm is  $O(N^2 T)$  in *HMM*. The adapted version of this algorithm has loops over length of given observation sequence  $T$ , length of previous hidden states  $N$ , length of target hidden state  $N$  and number of possible counter value  $M$ . Therefore, the time complexity of adapted Forward algorithm is  $O(N^2 KT)$ , where  $M \leq T$ . The complexity of original algorithm increases because of the counter factor  $M$ . Note that  $M$  is not necessarily proportional to  $T$  because, normally,

transitions that raise the counter relate to  $\Delta$  factor. This factor may keep the counter increment on a minimum scale where the incremented and decremented acts happen consecutively to the sequence. In fact, the maximum counter value may even be limited by a constant. In such cases, the algorithm has a linear time complexity depending on the length of the input.

### 4.2.2 Adapted Backward Algorithm

Another original dynamic programming algorithm that is used on *HMM* is the *backward* algorithm that computes the conditional probability of the future observation  $o_{t+1:T}$  with the condition on given the value of  $S_t$ . The counter values and the particular terminals must be considered in the adapted version of this algorithm in a *HP1CA*. The adapted version is addressed with two phases as adopted version of the forward algorithm is done.

#### ***The First Phase Preliminary Backward calculation***

This calculation illustrates the connected paths where it is laid between the terminals (i.e.  $S_0$  and  $S_T$ ) of the model. The calculation is represented as

$$\hat{\beta}_t(q, c) = Pr(S_T = (q_F, 0) | S_t = (q, c), S_0) \quad (4.53)$$

which is conditioned on hidden state-counter pair  $S_t = (q, c)$ , and starting  $S_0$ .

The algorithm starts the computation at the designated final terminal  $S_T = (q_F, 0)$ . According to this given knowledge then we can calculate the preliminary of adapted backward flow through two steps which are *base* step and *recursion* step. At the time  $t = T$ , the calculation must start at the position of the final terminal. If  $t = T$  is plugged into the equation 4.53 then the following re-formation of the same equation will be obtained

$$\begin{aligned} \hat{\beta}_T(q, c) &= Pr(S_T = (q_F, 0) | S_T = (q, c), S_0) \\ &= Pr(S_T = (q_F, 0) | S_T = (q, c)) \end{aligned}$$

The value of  $S_0$  is trivial due to *Markov* property that declares the current state depends only on the previous state. If the position of the current state  $q$  is in the final state  $q_F$  and the counter value  $c$  is equal to zero, then the preliminary backward value is equal to one at the step *base* step.

- *Base Step:*

$$\hat{\beta}_T(q, c) = \begin{cases} 1 : & q = q_F \text{ and } c = 0 \\ 0 : & \text{otherwise} \end{cases}$$

- *Recursion Step:*

if  $c = 0$  :

$$\hat{\beta}_t(q, c) = \sum_{q' \in Q} \sum_{\Delta=0}^1 \hat{\beta}_{t+1}(q', c + \Delta) A^0(q, \Delta, q') \quad (4.54)$$

if  $c > 0$  :

$$\hat{\beta}_t(q, c) = \sum_{q' \in Q} \sum_{\Delta=-1}^1 \hat{\beta}_{t+1}(q', c + \Delta) A^+(q, \Delta, q') \quad (4.55)$$

There is no emitting any output symbol at the first phase of the adapted version of the backward algorithm. The calculation starts at the final terminal (i.e.  $S_T = (q_F, 0)$ ) of the model in which illustrates the path between the final and terminals for each step of the recursive discrete time interval. For instance, let we have same parameters as given in sub-section 4.2.1 as shown in Figure 4.3. At the base step, the probability of preliminary backward of the final terminal is one. At time step  $t = 1$ , there are only four possible hidden state-counter pairs linked to the terminal state which are  $(q_1, 0)$ ,  $(q_1, 1)$ ,  $(q_2, 0)$ , and  $(q_2, 1)$  as shown in Figure 4.6.

It is easily seen that the adopted backward probability of a particular hidden state-counter pair is determined by the sum of the multiplication of the transition matrix which is enabled according to the counter value of the particular hidden control state and the future preliminary backward probability associated with that particular hidden state. At time step  $t = 1$ ,  $\hat{\beta}_1(q_1, 0)$  is

$$\begin{aligned} \hat{\beta}_1(q_1, 0) &= \hat{\beta}_2(q_2, 0) A^0(q_1, 0, q_2) \\ &= 1 * 0.3 = 0.3 \end{aligned}$$

And  $\hat{\beta}_0(q_1, 0)$  at  $t = 0$  is

$$\begin{aligned}
 \hat{\beta}_0(q_1, 0) &= \hat{\beta}_1(q_1, 0)A^0(q_1, 0, q_1) + \hat{\beta}_1(q_1, 1)A^0(q_1, 1, q_1) \\
 &\quad + \hat{\beta}_1(q_2, 0)A^0(q_1, 0, q_2) + \hat{\beta}_1(q_2, 1)A^0(q_1, 1, q_2) \\
 &= 0.25 * 0.25 + 0.3 * 0.4 + 0.25 * 0.25 + 0.3 * 0.1 \\
 &= 0.275
 \end{aligned}$$

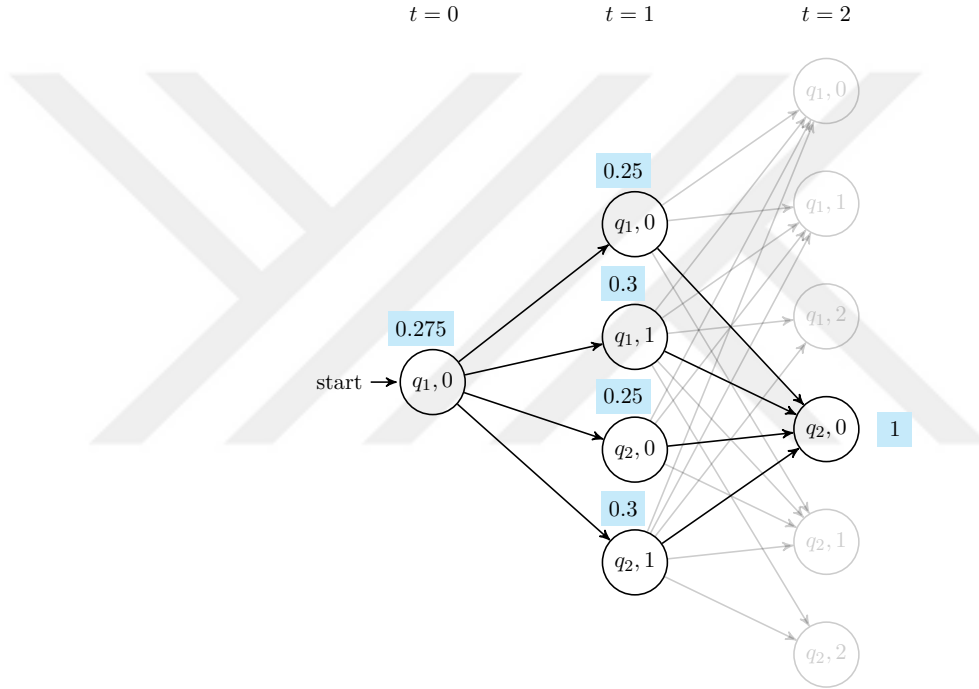


Figure 4.6: The trellis diagram of preliminary calculation of adapted backward algorithm

At the time step  $t = 0$ , the preliminary backward probability of the initial terminal of the model is equal to the probability of the total path which is between the initial and the final terminals of the model. These results illustrate that the initial and final terminals of a Model-1 type *HP1CA* are linked through particular state-counter pairs. The trivial paths which are not able to reach to the final or initial terminal due to improper counter value are discarded.

**Theorem 4.2.3.** *Preliminary backward algorithm computes the probability of the valid paths from final terminal to initial terminal by conditioning a particular hidden control*

*state-counter pair and initial terminal*

$$\hat{\beta}_t(q, c) = Pr(S_T | S_t = (q, c), S_0) \quad (4.56)$$

*Proof.* At the base step  $t = T$  we have

$$\hat{\beta}_T(q, c) = Pr(S_T = (q_F, 0) | S_T = (q, c), S_0) = Pr(S_T = (q_F, 0) | S_T = (q, c)) = 1 \quad (4.57)$$

According to the knowledge of designated terminals of the model the structure of each paths begins at starting state and they must reach to the final state. Equation 4.57 represents the probability of being in final state condition on starting state is equal to 1 if and only if the  $q = q_F$  is designated final state and the counter value  $c = 0$ .

At the recursion part  $0 < t \leq T - 1$ ; we assume that the Equation 4.56 provides correct calculation at  $t = k$  and now we need to prove that the same equation also provides correct calculation for  $t = k - 1$ . We can start the proof by applying sum rule over the future hidden control states  $S_k$ ,

$$\hat{\beta}_{k-1}(S_{k-1}) = \sum_{S_k} Pr(S_T, S_k | S_{k-1}, S_0) \quad (4.58)$$

Applying chain rule to move the  $S_k$  to the right hand side of the condition;

$$\hat{\beta}_{k-1}(S_{k-1}) = \sum_{S_k} Pr(S_T | S_k, S_{k-1}, S_0) Pr(S_k | S_{k-1}, S_0) \quad (4.59)$$

The first order of the Markov assumption the future state only depends on the current state. If we apply this assumption to the second argument of the Equation 4.59  $S_k$  does not depend on  $S_0$

$$\hat{\beta}_{k-1}(S_{k-1}) = \sum_{S_k} Pr(S_T | S_k, S_{k-1}, S_0) Pr(S_k | S_{k-1}) \quad (4.60)$$

The hidden state  $S_k$  is more close to final state  $S_T$  than  $S_{k-1}$ . In proportion to  $d$ -seperation,  $S_T \perp\!\!\!\perp S_{k-1} | S_k$

$$\hat{\beta}_{k-1}(S_{k-1}) = \sum_{S_k} Pr(S_T | S_k, S_0) Pr(S_k | S_{k-1}) \quad (4.61)$$

The first argument of Equation 4.61 is the preliminary backward probability at the future time step  $t = k$ . The second element of the same equation represents the state transition probability which depends on the counter value of the current hidden control state at time step  $t = k - 1$ . Under those circumstances we can plug all the argument into the Equation 4.61, then we have the recursion formula of the preliminary backward calculation for  $t = k - 1$ ,

$$\hat{\beta}_{k-1}(q, c) = \sum_{q' \in Q} \sum_{\Delta} \hat{\beta}_k(q', c + \Delta) A^{\delta(c)}(q, \Delta, q') \quad (4.62)$$

where

$$\delta(c) = \begin{cases} 0 & : c = 0 \\ + & : c > 0 \end{cases}$$

and

$$\Delta = \begin{cases} \{0, 1\} & : c = 0 \\ \{-1, 0, 1\} & : c > 0 \end{cases}$$

□

### ***The Second Phase: Normalization of Backward Calculation***

In this phase the future observation sequence  $o_{t+1:T}$  is emitted by the conditionally with the latent variables on the designated paths where are illustrated in the first phase of the backward calculation. This calculation is shown as

$$\beta_t(q, c) = Pr(o_{t+1:T} | S_t = (q, c), S_0, S_T) \quad (4.63)$$

The probability of emission of an output symbol is considered in this phase, unlike the first phase. Because reinforcing the connection between the initial and final terminal is determined in the first phase and it assists to discard the paths which are never reached to the final terminal. Only the valid hidden variable-counter pairs can produce an output symbol through those existing connections. Therefore, in this second phase, the *backward* probabilities of valid hidden variable-counter pairs can be normalized by adjusting the emission distribution to the calculation. The phase consists two steps which are *base* and *recursion* steps;

- *Base Step:*

$$\beta_T(q, c) = \begin{cases} 1 : & q = q_F \text{ and } c = 0 \\ 0 : & \text{otherwise} \end{cases}$$

- *Recursion Step:*

if  $c = 0$  :

$$\beta_t(q, c) = \sum_{q' \in Q} \sum_{\Delta=0}^1 \beta_{t+1}(q', \Delta) A^0(q, \Delta, q') B(q', o_{t+1}) \frac{\hat{\beta}_{t+1}(q', \Delta)}{\hat{\beta}_t(q, c)} \quad (4.64)$$

if  $c > 0$  :

$$\beta_t(q, c) = \sum_{q' \in Q} \sum_{\Delta=-1}^1 \beta_{t+1}(q', c + \Delta) A^+(q, \Delta, q') B(q', o_{t+1}) \frac{\hat{\beta}_{t+1}(q', c + \Delta)}{\hat{\beta}_t(q, c)} \quad (4.65)$$

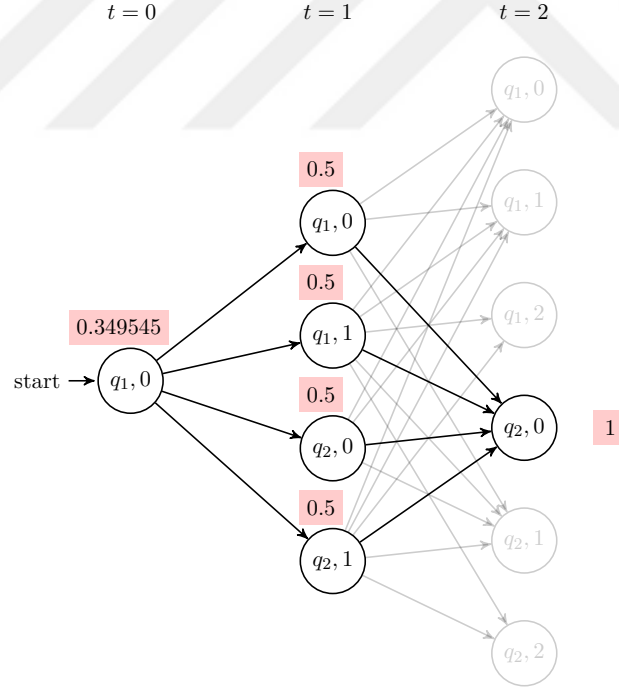


Figure 4.7: Normalisation calculation of adapted backward algorithm

The calculation of the normalisation of backward flow is also started at the last time step  $T$

as in the first phase of the adopted backward algorithm. The point where the computation starts is being in the final state which has zero counter value. The preliminary probability of backward flow is needed to be used in the second phase. The emission of an output symbol is also considered in the normalisation part.

At time  $t = 1$ , the probability of the normalization of  $\beta_1(q_2, 1)$  is

$$\begin{aligned}\beta_1(q_2, 1) &= \beta_2(q_2, 0)A^+(q_2, -1, q_2)B(q_2, b)\frac{\hat{\beta}_2(q_2, 0)}{\hat{\beta}_1(q_2, 1)} \\ &= 1 * 0.3 * 0.5 * \frac{1}{0.3} \\ &= 0.5\end{aligned}$$

The important part of this second phase is the probability value of the normalization of the backward calculation of the initial terminal is equal to the normalization of the forward flow of the final terminal. That equation provides that the probability value of the given observation sequence  $(ab)$  is same even that is calculated by using forward or backward calculation flow.

**Theorem 4.2.4.** *The normalize backward flow paths where it is lied between the terminals  $S_0$  and  $S_T$  is calculated as Equation 4.63. Each recursion cycle, the equation provide the following calculation*

$$\begin{aligned}\beta_t(q, c) &= \sum_{S_{t+1}} \beta_{t+1}(S_{t+1})A^{\delta(c)}(S_t, \Delta, S_{t+1})B(S_{t+1}, o_{t+1})\frac{\hat{\beta}_{t+1}(S_{t+1})}{\hat{\beta}_t(S_t)} \\ &= \sum_{S_{t+1}} Pr(o_{t+2:T}|S_{t+1}, S_0, S_T)Pr(S_{t+1}|S_t)Pr(o_{t+1}|S_{t+1})\frac{Pr(S_T|S_{t+1}, S_0)}{Pr(S_T|S_t, S_0)}\end{aligned}\tag{4.66}$$

where

$$\delta(c) = \begin{cases} 0 & \text{and } \Delta \in \{0, 1\} & : (c) = 0 \\ + & \text{and } \Delta \in \{-1, 0, 1\} & : (c) > 0 \end{cases}$$

*Proof.* At the base step  $t = T$ , we have

$$\beta_T(q, c) = Pr(o_{T+1:T}|S_T = (q, c), S_0, S_T)\tag{4.67}$$

where  $o_{T+1:T}$  is trivial that we can remove it from the equation then the updated equation is

$$\beta_T(q, c) = 1 \quad (4.68)$$

The second row of the Equation 4.68 is obtained if and only if the hidden state  $q$  is final state  $q_F$  and the counter value  $c$  is equal to zero. There is no any observation at time  $t = T + 1$ . The last observation is emitted by the final terminal of the model. That means we have already reached to the final terminal thus the probability of the normalization of the backward flow must be equal to one at base step.

At the recursion part; we assume that the Equation 4.66 provides correct calculation at  $t = k$  and now we need to prove that it also provides correct calculation for time step  $t = k - 1$ . We can start with to applying product rule over the first argument of the Equation 4.66

$$\beta_{k-1}(S_{k-1}) = \sum_{S_k} \frac{Pr(o_{k+1:T}, S_k, S_0, S_T) Pr(S_k | S_{k-1}) Pr(o_k | S_k) Pr(S_T | S_k, S_0)}{Pr(S_k, S_0, S_T) Pr(S_T | S_{k-1}, S_0)} \quad (4.69)$$

Again applying product rule to the first argument of the numerator

$$\beta_{k-1}(S_{k-1}) = \sum_{S_k} \frac{Pr(o_{k+1:T}, S_0, S_T | S_k) Pr(S_k) Pr(S_k | S_{k-1}) Pr(o_k | S_k) Pr(S_T | S_k, S_0)}{Pr(S_k, S_0, S_T) Pr(S_T | S_{k-1}, S_0)} \quad (4.70)$$

Merging two arguments of Equation 4.70 which are seperated condition on  $S_k$

$$\beta_{k-1}(S_{k-1}) = \sum_{S_k} \frac{Pr(o_{k:T}, S_0, S_T | S_k) Pr(S_k) Pr(S_k | S_{k-1}) Pr(S_T | S_k, S_0)}{Pr(S_k, S_0, S_T) Pr(S_T | S_{k-1}, S_0)} \quad (4.71)$$

Now again applying product rule to convert the conditionally distribution on  $S_k$  to joint probability

$$\beta_{k-1}(S_{k-1}) = \sum_{S_k} \frac{Pr(o_{k:T}, S_0, S_T, S_k) Pr(S_k | S_{k-1}) Pr(S_T | S_k, S_0)}{Pr(S_k, S_0, S_T) Pr(S_T | S_{k-1}, S_0)} \quad (4.72)$$

Applying product rule to the last element of the numerator

$$\beta_{k-1}(S_{k-1}) = \sum_{S_k} \frac{Pr(o_{k:T}, S_0, S_T, S_k) Pr(S_k | S_{k-1}) Pr(S_T, S_k, S_0)}{Pr(S_k, S_0, S_T) Pr(S_T | S_{k-1}, S_0) Pr(S_k, S_0)} \quad (4.73)$$

In Equation 4.73 there are same element in both numerator and denominator which is  $Pr(S_k, S_0, S_T)$ . We can eliminate these same arguments

$$\beta_{k-1}(S_{k-1}) = \sum_{S_k} \frac{Pr(o_{k:T}, S_0, S_T, S_k) Pr(S_k | S_{k-1})}{Pr(S_T | S_{k-1}, S_0) Pr(S_k, S_0)} \quad (4.74)$$

Applying product rule to the first element of the numerator

$$\beta_{k-1}(S_{k-1}) = \sum_{S_k} \frac{Pr(o_{k:T}, S_T | S_k, S_0) Pr(S_k, S_0) Pr(S_k | S_{k-1})}{Pr(S_T | S_{k-1}, S_0) Pr(S_k, S_0)} \quad (4.75)$$

Eliminating  $Pr(S_k, S_0)$  from both numerator and denominator

$$\beta_{k-1}(S_{k-1}) = \sum_{S_k} \frac{Pr(o_{k:T}, S_T | S_k, S_0) Pr(S_k | S_{k-1})}{Pr(S_T | S_{k-1}, S_0)} \quad (4.76)$$

Applying product rule to the denominator

$$\beta_{k-1}(S_{k-1}) = \sum_{S_k} \frac{Pr(o_{k:T}, S_T | S_k, S_0) Pr(S_k | S_{k-1}) Pr(S_{k-1}, S_0)}{Pr(S_T, S_{k-1}, S_0)} \quad (4.77)$$

Again applying the another product rule to the last argument of the numerator

$$\beta_{k-1}(S_{k-1}) = \sum_{S_k} \frac{Pr(o_{k:T}, S_T | S_k, S_0) Pr(S_k | S_{k-1}) Pr(S_0 | S_{k-1}) Pr(S_{k-1})}{Pr(S_T, S_{k-1}, S_0)} \quad (4.78)$$

The second and the third arguments of the numerator are seperated by conditionally on  $S_{k-1}$ . These two elements might be merge conditioned on  $S_{k-1}$  due to  $d$ -seperaton of  $S_0 \perp\!\!\!\perp S_k | S_{k-1}$

$$\beta_{k-1}(S_{k-1}) = \sum_{S_k} \frac{Pr(o_{k:T}, S_T | S_k, S_0) Pr(S_0, S_k | S_{k-1}) Pr(S_{k-1})}{Pr(S_T, S_{k-1}, S_0)} \quad (4.79)$$

Converting the conditional distribution to joint distribution probability by applying product rule to the last two arguments of the numerator

$$\beta_{k-1}(S_{k-1}) = \sum_{S_k} \frac{Pr(o_{k:T}, S_T | S_k, S_0) Pr(S_0, S_k, S_{k-1})}{Pr(S_T, S_{k-1}, S_0)} \quad (4.80)$$

Applying product rule to the first element of the numerator

$$\beta_{k-1}(S_{k-1}) = \sum_{S_k} \frac{Pr(o_{k:T}, S_T, S_k, S_0) Pr(S_0, S_k, S_{k-1})}{Pr(S_k, S_0) Pr(S_T, S_{k-1}, S_0)} \quad (4.81)$$

Applying product rule

$$\beta_{k-1}(S_{k-1}) = \sum_{S_k} \frac{Pr(o_{k:T}, S_T | S_k, S_0) Pr(S_k, S_0) Pr(S_{k-1} | S_k, S_0) Pr(S_k, S_0)}{Pr(S_k, S_0) Pr(S_T, S_{k-1}, S_0)} \quad (4.82)$$

Merging two separated arguments

$$\beta_{k-1}(S_{k-1}) = \sum_{S_k} \frac{Pr(o_{k:T}, S_T, S_{k-1} | S_k, S_0) Pr(S_k, S_0)}{Pr(S_T, S_{k-1}, S_0)} \quad (4.83)$$

Applying product rule

$$\beta_{k-1}(S_{k-1}) = \sum_{S_k} \frac{Pr(o_{k:T}, S_T, S_{k-1}, S_k, S_0)}{Pr(S_T, S_{k-1}, S_0)} \quad (4.84)$$

Applying product rule

$$\beta_{k-1}(S_{k-1}) = \sum_{S_k} \frac{Pr(o_{k:T}, S_k | S_{k-1}, S_0, S_T) Pr(S_{k-1}, S_0, S_T)}{Pr(S_T, S_{k-1}, S_0)} \quad (4.85)$$

Eliminating the same arguments

$$\beta_{k-1}(S_{k-1}) = \sum_{S_k} Pr(o_{k:T}, S_k | S_{k-1}, S_0, S_T) \quad (4.86)$$

Removing the sum rule over  $S_k$

$$\beta_{k-1}(S_{k-1}) = Pr(o_{k:T} | S_{k-1}, S_0, S_T) \quad (4.87)$$

Then we proved that the Equation 4.66 equals to Equation 4.63 at time  $t = k - 1$ .  $\square$

### 4.2.3 Adapted Baum-Welch Algorithm

It is worth emphasizing that Baum-Welch algorithm is used to estimate the parameters of the model with exploiting *forward* and *backward* algorithms on HMM. Indeed, the adapted

version of this algorithm relies on the counter value as relies as in adapted versions of *forward* and *backward* algorithms. This original algorithm is considered as a special case of the *Expectation Maximization*( EM ) algorithm therefore the same aspect will be considered on the adapted version with counter variable is taken in consideration.

Suppose that the length of each observation sequence is  $T$  where is represented as  $O^{(i)} = (o_1^{(i)}, \dots, o_T^{(i)})$ . We have the dataset that contains  $D$  dimension of observation sequences  $\mathcal{O} = (O^{(1)}, O^{(2)}, \dots, O^{(D)})$ . Let series of the states which is generating the observation sequence is presented as  $S^{(i)} = (s_1^{(i)}, \dots, s_T^{(i)})$  drawn from set of states  $\mathcal{S} = (S^{(1)}, S^{(2)}, \dots, S^{(D)})$ . The all counter values are natural number  $c \in \mathbb{N}$  drawn from set of the counters  $\mathcal{C} = (C^{(1)}, C^{(2)}, \dots, C^{(D)})$ . An HP1CA  $\theta = (B, A^0, A^+)$  is parameterized by two assumptions:

**Markov Assumption:**

According to the Markov assumption, the current state depends only on previous state where we can write the conditional distribution explicitly as (i.e. we can say it is equal to the probability of a state sequence );

$$A^0(q, \Delta, q') = Pr(s_{t+1} = q', c_{t+1} = \Delta | s_t = q, c_t = 0) \quad (4.88)$$

$$\sum_{q'} \sum_{\Delta=0}^1 A^0(q, \Delta, q') = 1, \forall q \quad (4.89)$$

where it is enabled if the counter  $c_t$  equals to zero. The counter changing either stays zero or increases one.

$$A^+(q, \Delta, q') = Pr(s_{t+1} = q', c_{t+1} = c_t + \Delta | s_t = q, c_t > 0) \quad (4.90)$$

$$\sum_{q'} \sum_{\Delta=-1}^1 A^+(q, \Delta, q') = 1; \quad \forall q \quad (4.91)$$

where it is enabled if the counter  $c_t$  is non-zero. The counter changing stays same or increases one or decrease one.

**Output Assumption:**

Observation symbol  $o_t = o$  is emitted by hidden states as discrete time  $t$ . The conditional

distribution could be shown as;

$$B(q, o) = Pr(o_t = o | s_t = q) \quad (4.92)$$

where;

$$\sum_o B(q, o) = 1; \quad \forall q \quad (4.93)$$

We can use the posterior parameters ( $\theta$ ) to maximize the new parameters ( $\theta'$ ) of the model then we might write the joint probability distribution over hidden, counter and observation variables given by

$$Pr(S, C, \mathcal{O} | \theta) = \prod_{d=1}^D \prod_{t=1}^T B(o_t^{(d)}, S_t^{(d)}) A^{\delta(C_t^{(d)})}(S_t^{(d)}, C_{t+1}^{(d)} - C_t^{(d)}, S_{t+1}^{(d)}) \quad (4.94)$$

where

$$\delta(C_t^{(d)}) = \begin{cases} 0, & C_t^{(d)} = 0 \\ +, & C_t^{(d)} > 0 \end{cases}$$

We want to determine the parameters of the model by maximizing log-likelihood by using observation sequence  $\mathcal{O} = (O^{(1)}, O^{(2)}, \dots, O^{(D)})$

$$l(\mathcal{O}, \theta) = \log Pr(\mathcal{O} | \theta) \quad (4.95)$$

But we do not know explicit form of  $Pr(\mathcal{O} | \theta)$ , instead we can obtain the likelihood function from the joint probability distribution by marginalizing over hidden and counter variables

$$Pr(S, C, \mathcal{O} | \theta) = \sum_{S \in \mathcal{S}} \sum_{C \in \mathcal{C}} Pr(S, C, \mathcal{O} | \theta) \quad (4.96)$$

If we put the likelihood function into Equation 4.95

$$l(\mathcal{O}, \theta) = \log \left[ \sum_{S \in \mathcal{S}} \sum_{C \in \mathcal{C}} Pr(S, C, \mathcal{O} | \theta) \right] \quad (4.97)$$

Directly maximizing  $l(\mathcal{O}, \theta)$  of this form is difficult because of the logarithm term can not be further reduced. Instead of examining through all possible hidden variables and maximizing their sum, the adopted version of Baum-Welch (BW) algorithm is going to be used to maximize the log likelihood.

The BW algorithm processes an iterative procedure to estimate the  $\theta$  from only  $\mathcal{O}$ . It works by maximizing a proxy to the log-likelihood, and updating the current model to be closer to the optimal model. The algorithm is repeating the following steps:

- **E Step:** Compute  $\mathcal{Q}(\theta, \theta') = \sum_{S \in \mathcal{S}} \sum_{C \in \mathcal{C}} \log[Pr(\mathcal{O}, S, C|\theta)] Pr(S, C|\mathcal{O}, \theta')$
- **M Step:** Set  $\theta' = \operatorname{argmax}_{\theta} \mathcal{Q}(\theta, \theta')$

Before the maximization, it is convenient to introduce two calculations for E step.

**Theorem 4.2.5.** We use  $\gamma_t(q, c) = Pr(S_t|O, S_0, S_T)$  to denote the calculation of the marginal posterior distribution of a hidden variable  $S_t$ .

$$\gamma_t(q, c) = \frac{\hat{\alpha}_t(q, c)\beta_t(q, c)\hat{\beta}_t(q, c)}{\hat{\alpha}_t(q_F, 0)} \quad (4.98)$$

*Proof.* The posterior marginal distribution calculation of  $\gamma_t(q, c)$  is

$$\gamma_t(q, c) = Pr(S_t = (q, c)|O, S_0, S_T = (q_F, 0)) \quad (4.99)$$

The proof can be started by applying the product rule to make only the initial terminal being in the condition part of this calculation.

$$\gamma_t(q, c) = \frac{Pr(O, S_t, S_T|S_0)}{Pr(O, S_T|S_0)} \quad (4.100)$$

The given whole observation sequence  $O$  can be expanded into two-piece as;

$$\gamma_t(q, c) = \frac{Pr(o_{1:t}, o_{t+1:T}, S_t, S_T|S_0)}{Pr(O, S_T|S_0)} \quad (4.101)$$

If the product rule is applied again to move the particular state into the condition part;

$$\gamma_t(q, c) = \frac{Pr(o_{1:t}, o_{t+1:T}, S_T|S_t, S_0)Pr(S_t|S_0)}{Pr(O, S_T|S_0)} \quad (4.102)$$

According to the *d-separation* rule, the first part of the observation sequence,  $o_{1:t}$ , is independent from the future observation sequence  $o_{t+1:T}$  and the final terminal  $S_T$  by condition on the particular hidden state-counter pair  $S_t$ ;

$$\gamma_t(q, c) = \frac{Pr(o_{1:t}|S_t, S_0)Pr(o_{t+1:T}, S_T|S_t, S_0)Pr(S_t|S_0)}{Pr(O, S_T|S_0)} \quad (4.103)$$

The product rule can be applied to the second argument of the numerator of equation 4.103;

$$\gamma_t(q, c) = \frac{Pr(o_{1:t}|S_t, S_0)Pr(o_{t+1:T}|S_t, S_0, S_T)Pr(S_T|S_t, S_0)Pr(S_t|S_0)}{Pr(O, S_T|S_0)} \quad (4.104)$$

The first item (i.e.  $Pr(o_{1:t}|S_t, S_0)$ ) and the last item (i.e.  $Pr(S_t|S_0)$ ) of the equation 4.104 is equal to  $Pr(o_{1:t}, S_t|S_0)$  where the product rule is applied.

$$\gamma_t(q, c) = \frac{Pr(o_{1:t}, S_t|S_0)Pr(o_{t+1:T}|S_t, S_0, S_T)Pr(S_T|S_t, S_0)}{Pr(O, S_T|S_0)} \quad (4.105)$$

The first argument of the numerator of the equation 4.105 is equal to the calculation of the preliminary forward flow 4.8, the second item is the definition of the calculation of the normalization backward algorithm 4.63, and the last argument of the numerator is equal to the preliminary calculation of the backward flow 4.53.

$$\gamma_t(q, c) = \frac{\hat{\alpha}_t(q, c)\beta_t(q, c)\hat{\beta}_t(q, c)}{Pr(O, S_T|S_0)} \quad (4.106)$$

The denominator of the equation 4.106 is equal to the probability of the preliminary forward of the final terminal. The given whole observation sequence can be shown as  $o_{1:T}$  and the  $S_T$  is the final state with zero counter value. On the other hand, the last observation symbol must be emitted by the final state. According to this knowledge the denominator part of this equation is representing the preliminary forward probability of the whole observation sequence that is ended being in the final terminal.

$$\gamma_t(q, c) = \frac{\hat{\alpha}_t(q, c)\beta_t(q, c)\hat{\beta}_t(q, c)}{\hat{\alpha}_T(q_F, 0)} \quad (4.107)$$

□

If return to the same example that is handled in the section 4.2.1, the gamma calculation of all particular hidden state-counter pairs which existing between the terminals of the model is shown in Figure 4.8.

For instance, there is only one initial terminal and the probability of being in the initial terminal at  $t = 0$  given the observation sequence  $(a, b)$  is

$$\begin{aligned}\gamma_0(q_1, 0) &= \frac{1 * 0.34954545 * 0.275}{0.096125} \\ &= \frac{0.096125}{0.096125} \\ &= 1\end{aligned}$$

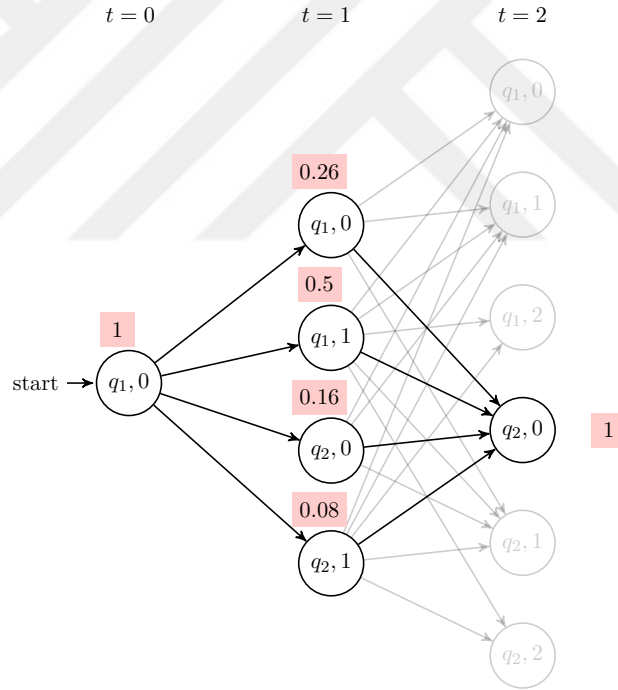


Figure 4.8: Gamma calculation of being in state-counter pair

The other calculation of *E-step* is finding the probabilities of two particular hidden state-counter pairs according to the given observation sequence.

**Theorem 4.2.6.**  $\xi_t((q', c'), (q, c)) = Pr(S_{t-1}, S_t | O, S_0, S_T)$  calculates the joint posterior

distribution of two successive hidden variables of the model where  $S_{t-1} = (q', c')$  is the hidden state-counter pair at time  $t - 1$  and  $S_t = (q, c)$  is the hidden state-counter pair at time  $t$ .

$$\xi_t((q', c'), (q, c)) = \frac{\hat{\alpha}_{t-1}(q', c') A^{\delta(c')}(q', \Delta, q) B(q, o_t) \hat{\beta}_t(q, c) \beta_t(q, c)}{\hat{\alpha}_T(q_F, 0)} \quad (4.108)$$

*Proof.* The posterior joint distribution of two successive hidden variables is

$$\xi_t((q', c'), (q, c)) = Pr(S_{t-1} = (q', c'), S_t = (q, c) | O, S_0, S_T) \quad (4.109)$$

The proof can be started by applying product rule to keep the starting point on the condition part

$$\xi_t(S_{t-1}, S_t) = \frac{Pr(S_{t-1}, S_t, O, S_T | S_0)}{Pr(O, S_T | S_0)} \quad (4.110)$$

The whole observation sequence can be expanded into two parts

$$\xi_t(S_{t-1}, S_t) = \frac{Pr(o_{1:t}, o_{t+1:T}, S_{t-1}, S_t, S_T | S_0)}{Pr(O, S_T | S_0)} \quad (4.111)$$

The product rule again can be applied to the numerator;

$$\xi_t(S_{t-1}, S_t) = \frac{Pr(o_{1:t}, o_{t+1:T}, S_T | S_{t-1}, S_t, S_0) Pr(S_{t-1}, S_t | S_0)}{Pr(O, S_T | S_0)} \quad (4.112)$$

According to  $d - separation$  rule, the single output symbol  $o_t$  is conditionally independent with all other output symbols of the observation sequence and the terminals of the model as  $o_t \perp\!\!\!\perp o_{1:t-1}, o_{t+1:T}, S_T | S_t$  and  $o_t \perp\!\!\!\perp S_0, S_{t-1} | S_t$ .

$$\xi_t(S_{t-1}, S_t) = \frac{Pr(o_{1:t-1}, o_{t+1:T}, S_T | S_{t-1}, S_t, S_0) Pr(S_{t-1}, S_t | S_0) Pr(o_t | S_t)}{Pr(O, S_T | S_0)} \quad (4.113)$$

Again according to the  $d - separation$  rule, the prior observation sequence part,  $o_{1:t-1}$ , is independent to the future observation sequence  $o_{t+1:T}$ , final terminal  $S_T$ , and the particular hidden variable  $S_t$  conditionally on  $S_{t-1}$  where  $o_{1:t-1} \perp\!\!\!\perp o_{t+1:T}, S_T | S_{t-1}$ ,  $o_{1:t-1} \perp\!\!\!\perp S_t | S_{t-1}$ .

$$\xi_t(S_{t-1}, S_t) = \frac{Pr(o_{1:t-1} | S_{t-1}, S_0) Pr(o_{t+1:T}, S_T | S_{t-1}, S_t, S_0) Pr(S_{t-1}, S_t | S_0) Pr(o_t | S_t)}{Pr(O, S_T | S_0)} \quad (4.114)$$

In the second argument of the equation 4.114, the hidden variable  $S_{t-1}$  is irrelevant item on the condition part due to the future observation sequence and the final terminal are independent from this hidden variable conditionally on  $S_t$ .

$$\xi_t(S_{t-1}, S_t) = \frac{Pr(o_{1:t-1}|S_{t-1}, S_0)Pr(o_{t+1:T}, S_T|S_t, S_0)Pr(S_{t-1}, S_t|S_0)Pr(o_t|S_t)}{Pr(O, S_T|S_0)} \quad (4.115)$$

The product rule can be applied to the second element of the numerator;

$$\xi_t(S_{t-1}, S_t) = \frac{Pr(o_{1:t-1}|S_{t-1}, S_0)Pr(o_{t+1:T}|S_t, S_0, S_T)Pr(S_T|S_t, S_0)Pr(S_{t-1}, S_t|S_0)Pr(o_t|S_t)}{Pr(O, S_T|S_0)} \quad (4.116)$$

If the product rule is performed to the third item of the equation 4.116, then the following equation is achieved;

$$\xi_t(S_{t-1}, S_t) = \frac{Pr(o_{1:t-1}|S_{t-1}, S_0)Pr(o_{t+1:T}|S_t, S_0, S_T)Pr(S_T|S_t, S_0)Pr(S_t|S_{t-1}, S_0)Pr(S_{t-1}|S_0)Pr(o_t|S_t)}{Pr(O, S_T|S_0)} \quad (4.117)$$

The first and the fifth items of the equation 4.117 is equal to  $Pr(o_{1:t-1}, S_{t-1}|S_0)$  where the product rule is applied;

$$\xi_t(S_{t-1}, S_t) = \frac{Pr(o_{1:t-1}, S_{t-1}|S_0)Pr(o_{t+1:T}|S_t, S_0, S_T)Pr(S_T|S_t, S_0)Pr(S_t|S_{t-1}, S_0)Pr(o_t|S_t)}{Pr(O, S_T|S_0)} \quad (4.118)$$

According to *Markov* property, the initial terminal  $S_0$  is trivial in  $Pr(S_t|S_{t-1}, S_0)$ . Thus, it can be removed from the condition part.

$$\xi_t(S_{t-1}, S_t) = \frac{Pr(o_{1:t-1}, S_{t-1}|S_0)Pr(o_{t+1:T}|S_t, S_0, S_T)Pr(S_T|S_t, S_0)Pr(S_t|S_{t-1})Pr(o_t|S_t)}{Pr(O, S_T|S_0)} \quad (4.119)$$

At the end of the proof, all arguments of the numerator of the equation 4.119 are preliminary forward calculation  $\hat{\alpha}_{t-1}(S_{t-1})$ ,  $\beta_t(S_t)$ ,  $\hat{\beta}_t(S_t)$ ,  $A^{\delta(c')}(S_{t-1}, \Delta, S_t)$ , and  $B(S_t, o_t)$ , respectively. If these all arguments are plugged into the same equation then the following formula is obtained;

$$\xi_t((q', c'), (q, c)) = \frac{\hat{\alpha}_{t-1}(q', c')\beta_t(q, c)\hat{\beta}_t(q, c)A^{\delta(c')}(q', \Delta, q)B(q, o_t)}{Pr(O, S_T|S_0)} \quad (4.120)$$

where

$$\delta(c') = \begin{cases} 0 & : c' = 0 \\ + & : c' > 0 \end{cases}$$

□

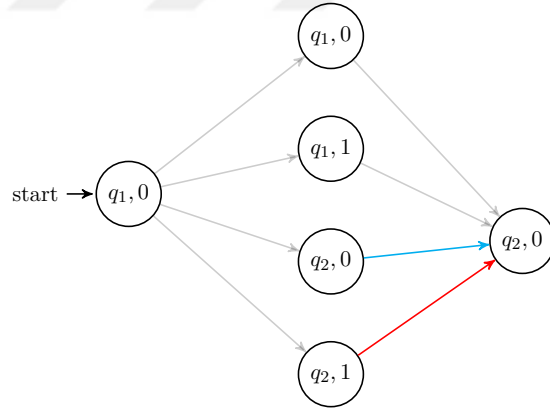
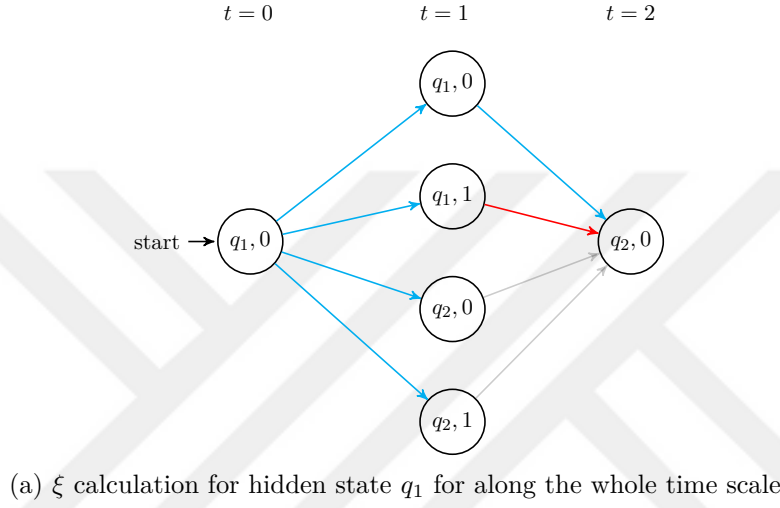


Figure 4.9:  $\xi$  calculation for all hidden states given the observation sequence  $(a, b)$

The expected number of the transitions from a particular hidden state-counter pair (e.g.  $(q', c')$ ) to another particular hidden state-counter pair (e.g.  $(q, c)$ ) can be shown in Figure 4.9. All existing transitions which is spread from  $(q_1, 0)$  to other hidden variables are shown

as cyan-colored arrows in Figure 4.9a. The red arrows are representing the exiting paths where the counter value of the particular hidden variable where the transition is started the spreading from is non-zero.

For instance, let the particular hidden variable is  $(q_1, 0)$  and the next hidden variable is  $(q_2, 0)$  as shown in Figure 4.9a. The probability calculation of the all existing paths between those two hidden variable along the whole time scale is where at  $t = 1$ ,

$$\begin{aligned}\xi_1((q_1, 0), (q_2, 0)) &= \frac{\hat{\alpha}_0(q_1, 0)\beta_1(q_2, 0)\hat{\beta}_1(q_2, 0)A^0(q_1, 0, q_2)B(q_2, a)}{\hat{\alpha}_2(q_2, 0)} \\ &= \frac{1 * 0.5 * 0.25 * 0.25 * 0.5}{0.096125} \\ &= 0.16255\end{aligned}$$

and the other existing path between the same particular hidden variables at  $t = 2$ ;

$$\begin{aligned}\xi_2((q_1, 0), (q_2, 0)) &= \frac{\hat{\alpha}_1(q_1, 0)\beta_2(q_2, 0)\hat{\beta}_2(q_2, 0)A^0(q_1, 0, q_2)B(q_2, b)}{\hat{\alpha}_2(q_2, 0)} \\ &= \frac{0.2 * 1 * 1 * 0.25 * 0.5}{0.096125} \\ &= 0.26\end{aligned}$$

The important part of this calculation is the difference between the counter values that is between two particular hidden variables. That difference is playing a substantial role to determine the  $\Delta$  values which are pointing to probability value between the same hidden variables into the enabled transition matrix. For instance, if the same example is handled again, then it is easily seen that the transition matrix  $A^0$  is enabled and  $\Delta \in \{0\}$ . In the *M-step* of this algorithm, these two values of the calculation are used while updating the value of  $A^0(q_1, 0, q_2)$ .

In *M-step*,  $Q(\theta, \theta')$  is maximized with respect to  $\theta = \{\hat{A}^0, \hat{A}^+, \hat{B}\}$ . Maximization with the parameter of the  $\theta$  is achieved using Lagrange multiplier. First,

$$Pr(S, C, \mathcal{O}) = Pr(\mathcal{O})Pr(S, C|\mathcal{O})$$

we can write;

$$Pr(S, C|\mathcal{O}) = \frac{Pr(S, C, \mathcal{O})}{Pr(\mathcal{O})} \quad (4.121)$$

Plugging this into the *M-step*

$$\begin{aligned} Pr(S, C|\mathcal{O}) &= \operatorname{argmax}_{\theta} \sum_{S \in \mathcal{S}} \sum_{C \in \mathcal{C}} \log[Pr(\mathcal{O}, S, C|\theta)] Pr(S, C|\mathcal{O}, \theta') \\ &= \operatorname{argmax}_{\theta} \sum_{S \in \mathcal{S}} \sum_{C \in \mathcal{C}} \log[Pr(\mathcal{O}, S, C|\theta)] \frac{Pr(S, C, \mathcal{O}|\theta')}{Pr(\mathcal{O}|\theta')} \\ &= \operatorname{argmax}_{\theta} \hat{Q}(\theta, \theta') \end{aligned} \quad (4.122)$$

$Pr(\mathcal{O}|\theta')$  could be discard because of it does not depend on the hidden states  $S$  and counters  $C$ . Taking logarithm of Equation 4.94

$$\begin{aligned} \log Pr(S, C, \mathcal{O}|\theta) &= \sum_{d=1}^D \left[ \sum_{t=1}^T \log B(o_t^{(d)}, S_t^{(d)}) + \sum_{\{t|C_t^{(d)}=0\}} \log A^0(S_t^{(d)}, C_{t+1}^{(d)} - C_t^{(d)}, S_{t+1}^{(d)}) \right. \\ &\quad \left. + \sum_{\{t|C_t^{(d)}>0\}} \log A^+(S_t^{(d)}, C_{t+1}^{(d)} - C_t^{(d)}, S_{t+1}^{(d)}) \right] \end{aligned} \quad (4.123)$$

The emission distribution does not depend on the counter value. The counter parameter might be discarded from the formula of emission distribution. Use Lagrange multiplier to optimize the formula, respectively.

$$\begin{aligned} \mathcal{L}(\theta, \theta', \lambda, \epsilon^0, \epsilon^+) &= \mathcal{Q}(\theta, \theta') - \sum_q \lambda_q \left( \sum_o B(o, q) - 1 \right) \\ &\quad - \sum_q \epsilon_q^0 \left( \sum_{\Delta=0}^1 \sum_{q'} A^0(q, \Delta, q') - 1 \right) \\ &\quad - \sum_q \epsilon_q^+ \left( \sum_{\Delta=-1}^1 \sum_{q'} A^+(q, \Delta, q') - 1 \right) \end{aligned} \quad (4.124)$$

Partial derivation with  $B(o, q)$

$$\frac{\partial \mathcal{L}(\theta, \theta')}{\partial B(o, q)} = \frac{\partial}{\partial B(o, q)} \left( \sum_{S \in \mathcal{S}} \sum_{d=1}^D \sum_{t=1}^T \log B(o_t^{(d)}, S_t^{(d)}) Pr(S|\mathcal{O}, \theta') \right) - \lambda_q = 0 \quad (4.125)$$

$\mathcal{I}(o)$  denote an indicator function which is 1 if  $o_t = o$  is true, otherwise 0.

$$= \sum_{d=1}^D \sum_{t=1}^T \frac{Pr(S_t^{(d)} = q|\mathcal{O}, \theta') \mathcal{I}(o_t^{(d)} = o)}{B(o, q)} - \lambda_q = 0 \quad (4.126)$$

Partial derivation with  $\lambda_q$

$$\frac{\partial \mathcal{L}(\theta, \theta')}{\partial \lambda_q} = - \left( \sum_{o \in \mathcal{O}} B(o, q) - 1 \right) = 0 \quad (4.127)$$

Some algebra yields

$$\begin{aligned} &= 1 - \sum_{o \in \mathcal{O}} B(o, q) \\ &= 1 - \sum_{o \in \mathcal{O}} \frac{1}{\lambda_q} \sum_{d=1}^D \sum_{t=1}^T Pr(S_t^{(d)} = q|\mathcal{O}, \theta') \mathcal{I}(o_t^{(d)} = o) \equiv 0 \\ \lambda_q &= \sum_{o \in \mathcal{O}} \sum_{d=1}^D \sum_{t=1}^T Pr(S_t^{(d)} = q|\mathcal{O}, \theta') \mathcal{I}(o_t^{(d)} = o) \end{aligned} \quad (4.128)$$

Here  $\lambda_q$  does not depend to  $o$

$$\lambda_q = \sum_{d=1}^D \sum_{t=1}^T Pr(S_t^{(d)} = q|\mathcal{O}, \theta') \quad (4.129)$$

The equation 4.129 can be plugged into the equation 4.128, then the updated formulae of emission matrix is achieved as

$$\hat{B}(q, o) = \frac{\sum_{d=1}^D \sum_{t=1}^T Pr(S_t^{(d)} = q|\mathcal{O}, \theta') \mathcal{I}(o_t^{(d)} = o)}{\sum_{d=1}^D \sum_{t=1}^T Pr(S_t^{(d)} = q|\mathcal{O}, \theta')} \quad (4.130)$$

where  $\forall o \in \mathcal{O}$  and  $\forall q \in \mathcal{Q}$ .

The numerator of the updated formula of the emission matrix equals to the Theorem 4.2.5. And the denominator of the same formula represents the number of being in state  $S_t^{(d)}$  according to whole observation sequences. Thus, the updated formula might be represented as

$$\hat{B}(q, o) = \frac{\sum_{d=1}^D \sum_{t=1}^T \gamma_t^{(d)}(S_t^{(d)}) \mathcal{J}(o_t^{(d)} = o)}{\sum_{d=1}^D \sum_{t=1}^T \gamma_t^{(d)}(S_t^{(d)})} \quad (4.131)$$

For instance, let same example is handled to illustrate the estimation of the emission matrix for the given observation sequence  $(a, b)$ .  $\gamma$  calculation has the probability of being in each particular hidden state-counter pair for each individual time step  $t$ . The counter value might not be considered due to the probability of emission an output symbol depends only on the hidden state instead of state-counter pair.

The estimation of each observation symbols is emitted by the hidden variable  $q_2$  according the given observation sequence  $(a, b)$  is calculated as;

- The estimation of the emission probability for the output symbol  $a$  that is emitted by the hidden state  $q_2$  is

$$\begin{aligned} \hat{B}(q_2, a) &= \frac{\gamma_1(q_2, 0) \mathcal{J}(o_1 = a) + \gamma_1(q_2, 1) \mathcal{J}(o_1 = a) + \gamma_2(q_2, 0) \mathcal{J}(o_2 = a)}{\gamma_1(q_2, 0) + \gamma_1(q_2, 1) + \gamma_2(q_2, 0)} \\ &= \frac{0.16 + 0.08 + 0}{0.16 + 0.08 + 1} \\ &= 0.19355 \end{aligned}$$

- The estimation of the emission probability for the output symbol  $b$  that is emitted by the hidden state  $q_2$  is

$$\begin{aligned} \hat{B}(q_2, b) &= \frac{\gamma_1(q_2, 0) \mathcal{J}(o_1 = b) + \gamma_1(q_2, 1) \mathcal{J}(o_1 = b) + \gamma_2(q_2, 0) \mathcal{J}(o_2 = b)}{\gamma_1(q_2, 0) + \gamma_1(q_2, 1) + \gamma_2(q_2, 0)} \\ &= \frac{0 + 0 + 1}{0.16 + 0.08 + 1} \\ &= 0.80645 \end{aligned}$$

It is easily seen that the total probability of the emission for all observation symbol is equal to one for the variable  $q_2$ . The same calculation can be done for the other hidden variable  $q_1$ .

The similar derivative process is performed for the transition matrix  $A^0(q, \Delta, q')$  as it is done for the emission matrix  $B$ .

$$\begin{aligned} \frac{\partial \mathcal{L}(\theta, \theta')}{\partial A^0(q, \Delta, q')} &= \frac{\partial}{\partial A^0(q, \Delta, q')} \left( \sum_{S \in \mathcal{S}} \sum_{C \in \mathcal{C}} \sum_{d=1}^D \sum_{\{t|C_t^{(d)}=0\}} \log A^0(S_t^{(d)}, C_{t+1}^{(d)} - C_t^{(d)}, S_{t+1}^{(d)}) \right. \\ &\quad \left. Pr(S, C | \mathcal{O}, \theta') \right) - \epsilon_q^0 = 0 \end{aligned} \quad (4.132)$$

$$\begin{aligned} &= \frac{\partial}{\partial A^0(q, \Delta, q')} \left( \sum_{S_t} \sum_{C_t} \sum_{d=1}^D \sum_{\{t|C_t^{(d)}=0\}} \log A^0(q, \Delta, q') \right. \\ &\quad \left. Pr(S_t^{(d)} = q, C_t^{(d)} = 0, S_{t+1}^{(d)} = q', C_{t+1}^{(d)} = \Delta | \mathcal{O}, \theta') \right) - \epsilon_q^0 = 0 \end{aligned} \quad (4.133)$$

$$= \sum_{d=1}^D \sum_{\{t|C_t^{(d)}=0\}} \frac{Pr(S_t^{(d)} = q, C_t^{(d)} = 0, S_{t+1}^{(d)} = q', C_{t+1}^{(d)} = \Delta | \mathcal{O}, \theta')}{A^0(q, \Delta, q')} - \epsilon_q^0 = 0 \quad (4.134)$$

Partial derivation with  $\epsilon_q^0$

$$\frac{\partial \mathcal{L}(\theta, \theta')}{\partial \epsilon_q^0} = - \left( \sum_{\Delta=0}^1 \sum_{q'} A^0(q, \Delta, q') - 1 \right) \quad (4.135)$$

This yields

$$\begin{aligned} &= 1 - \sum_{\Delta=0}^1 \sum_{q'} A^0(q, \Delta, q') \\ &= 1 - \sum_{\Delta=0}^1 \sum_{q'} \frac{1}{\epsilon_q^0} \sum_{d=1}^D \sum_{\{t|C_t^{(d)}=0\}} Pr(S_t^{(d)} = q, C_t^{(d)} = 0, S_{t+1}^{(d)} = q', C_{t+1}^{(d)} = \Delta | \mathcal{O}, \theta') \equiv 0 \\ \epsilon_q^0 &= \sum_{\Delta=0}^1 \sum_{q'} \sum_{d=1}^D \sum_{\{t|C_t^{(d)}=0\}} Pr(S_t^{(d)} = q, C_t^{(d)} = 0, S_{t+1}^{(d)} = q', C_{t+1}^{(d)} = \Delta | \mathcal{O}, \theta') \end{aligned} \quad (4.136)$$

Plugged  $\epsilon_q^0$

$$\hat{A}^0(q, \Delta, q') = \frac{\sum_{d=1}^D \sum_{\{t|C_t^{(d)}=0\}} Pr(S_t^{(d)} = q, C_t^{(d)} = 0, S_{t+1}^{(d)} = q', C_{t+1}^{(d)} = \Delta | \mathcal{O}, \theta')}{\sum_{q'} \sum_{d=1}^D \sum_{\{t|C_t^{(d)}=0\}} Pr(S_t^{(d)} = q, C_t^{(d)} = 0, S_{t+1}^{(d)} = q', C_{t+1}^{(d)} = \Delta | \mathcal{O}, \theta')} \quad (4.137)$$

where  $\Delta \in \{0, 1\}$

The numerator of Equation 4.137 represents the transition from state  $S_t^{(d)}$  to hidden state  $S_{t+1}^{(d)}$  and the denominator represents the number of all transitions where comes from hidden state  $S_t^{(d)}$ . The updated formula is illustrated as

$$\hat{A}^0(q, \Delta, q') = \frac{\sum_{d=1}^D \sum_{\{t|C_t^{(d)}=0\}} \xi_t^{(d)}(S_t^{(d)}, S_{t+1}^{(d)})}{\sum_{S_{t+1}^{(d)}} \sum_{d=1}^D \sum_{\{t|C_t^{(d)}=0\}} \xi_t^{(d)}(S_t^{(d)}, S_{t+1}^{(d)})} \quad (4.138)$$

Similar process for  $A^+(q, \Delta, q')$

$$\hat{A}^+(q, \Delta, q') = \frac{\sum_{d=1}^D \sum_{\{t|C_t^{(d)}>0\}} \sum_{c=1}^t Pr(S_t^{(d)} = q, C_t^{(d)} = c, S_{t+1}^{(d)} = q', C_{t+1}^{(d)} = c + \Delta | \mathcal{O}, \theta')}{\sum_{q'} \sum_{d=1}^D \sum_{\{t|C_t^{(d)}>0\}} \sum_{c=1}^t Pr(S_t^{(d)} = q, C_t^{(d)} = c, S_{t+1}^{(d)} = q', C_{t+1}^{(d)} = c + \Delta | \mathcal{O}, \theta')} \quad (4.139)$$

the  $\Delta$  is given as  $\Delta \in \{-1, 0, 1\}$  where it is shown as

$$\hat{A}^+(q, \Delta, q') = \frac{\sum_{d=1}^D \sum_{\{t|C_t^{(d)}>0\}} \xi_t^{(d)}(S_t^{(d)}, S_{t+1}^{(d)})}{\sum_{S_{t+1}^{(d)}} \sum_{d=1}^D \sum_{\{t|C_t^{(d)}>0\}} \xi_t^{(d)}(S_t^{(d)}, S_{t+1}^{(d)})} \quad (4.140)$$

Let estimate the value of transition probability from hidden state  $q_1$  to  $q_2$  while  $\Delta = \{0\}$  in transition matrix  $A^0$ . According to the given observation sequence there are two existing transition between those two particular hidden state-counter pair along the whole time interval as seen in Figure 4.9a. The one transition from  $(q_1, 0)$  to  $(q_2, 0)$  at  $t = 1$  and the other transition is occurred at time step  $t = 2$ . The sum of these two transition probabilities is divided by sum of all transitions that is coming from the hidden variable

$(q_1, 0)$  along the sequence where

$$\begin{aligned}\hat{A}^0(q_1, 0, q_2) &= \frac{\sum_{t=1}^2 \xi_t((q_1, 0), (q_2, 0))}{\sum_{q=q_1}^{q_2} \sum_{t=1}^2 \sum_{\Delta=0}^1 \xi_t((q_1, 0), (q, \Delta))} \\ &= \frac{0.16255 + 0.26}{0.26 + 0.499 + 0.16255 + 0.078 + 0.26} \\ &= 0.3354\end{aligned}$$

The other values of the state transition matrices are estimated with the same calculation technique until the model converge to local optimum. The multiple observation sequences are playing an important role at the learning stage to reach a reliable estimation of the parameters of the model. Assuming that there are several observation sequences  $O^{(1)}, \dots, O^{(D)}$ . In this case,  $\gamma_t(S_t)$  and  $\xi_t(S_{t-1}, S_t)$  are computed for each independent individual observation sequence  $o_1^{(i)}, \dots, o_{T_i}^{(i)}$  where  $i = 1, \dots, D$  and  $T_i$  is the length of the  $i^{th}$  observation sequence.

On a *HMM*, the normalization of the value of the parameters must be done after the calculation of the estimation because of the sum of each row probabilities of the matrices must equal to one. However, no need for this computation in *HP1CA* because the normalization is already done in both adapted *forward* and adapted *backward* algorithms. In this case, after each iteration of the training, the sum of each row of the parameters of the model will be equal to one. Time complexity for one iteration of adapted *Baum-Welch* algorithm is  $\mathcal{O}(N^2MT)$  because this algorithm is governed by both adapted version of *forward* and *backward* flows.

### 4.3 Conclusion

In this chapter, a new counter machine which is a subclass of the probabilistic *pushdown* machines is defined where the control states and counter values are hidden. HMM and P1CA are inspired to build the structure of this model that is called *Hidden Probabilistic One-Counter Machine* ( i.e. *HP1CA*). Therefore, this new stochastic counter machine is the generalization of *P1CA* by using latent variables.

A *HMM* has three independent parameters. One of them is denoted by  $A$  that keeps the value of the distribution probability of hidden state transition. It is called *state transition matrix*. The other parameter of the model is called *emission matrix* which is denoted by  $B$ . This matrix holds the observation probability of an output symbol that is emitted by the hidden variable. The last parameter of the model is *initial vector* which holds the initial probability distribution of the hidden variables of the model. On the other hand, a *P1CA* consists of only two essential parameters which are called transition functions.  $\delta^0$  and  $\delta^+$  represent these transition functions where if the counter value is zero  $\delta^0$  is enabled, and if the counter value is bigger than zero  $\delta^+$  is enabled. The integration of the parameters of both *HMM* and *P1CA* and *P1CA* constitutes three essential parameters which are  $A^0$ ,  $A^+$ , and  $B$ .  $A^0$  and  $A^+$  are state transition matrices that keep the probability distribution of state transition instead of transition functions  $\delta^0$  and  $\delta^+$ .  $B$  is the emission matrix that is used in the same manner as it is used in a *HMM*.

The transition matrices depend on the current counter value at time  $t$  to be enabled in the model along the sequence. At each time step, the counter can be incremented, decremented, or stay the same value as the value of the counter at the previous time step  $t - 1$ . If the previous counter value (i.e.  $c'$ ) is zero, then the transition matrix  $A^0$  is enabled. If a transition is performed when this state transition matrix is active, the current counter (i.e.  $c$ ) value can be incremented or stay zero. Therefore, the counter value can be changed as  $c = c' + \Delta$  where  $\Delta \in \{0, 1\}$ . The same process is applied to the state transition matrix  $A^+$  along the same sequence if the previous counter value is non-zero. In this case, the counter can be changed as  $c = c' + \Delta$  where  $\Delta \in \{-1, 0, 1\}$ . The counter can be decremented due to it is a non-zero natural number. The emission matrix  $B$  of the model keeps the probability distribution of generating an output symbol. The probability of producing an output symbol does not depend on the counter value but only on the hidden state.

The model computes the joint probability of hidden and observed discrete random variables. The raw computation of this joint probability requires  $TN^{T-1}M$  multiplications and additions where  $T$  is the length of the given observation sequence,  $N$  is the number of hidden variables, and  $M$  is the maximum number of the counter value which depends on  $T$ . However, this computational complexity provides an infeasible computation even for small values of  $N$  and  $T$ . In this case, the more efficient way is needed to solve this

calculation. On the other hand, this model is an implementation of *Markov* process version of *P1CA* and a *P1CA* is the sub-class of *PPDA* with only one particular stack symbol. In this case, a *P1CA* recognizes the probabilistic version of *One-Counter Languages*(OCLs) which can be generated by using *Stochastic Context-free Grammar* (SCFG). *Inside-Outside* algorithms are used to estimate the parameters for obtaining a better probability score for parsing. However, these algorithms have a cubic time computational complexity. Reducing the complexity and finding a feasible way to solving the problems of this model can be handled by adaptation the fundamental algorithms of *HMM* which are *forward* and *backward*.

The *forward* and *backward* algorithms are used as a kind of dynamic programming algorithm that allow for reduction of the computational complexity for the likelihood problem for *HMM*. The adapted version of both algorithms consist two essential phases which are called *preliminary* phase and *normalization* phase. The *preliminary* phase of *forward* flow, which is denoted as  $\hat{\alpha}_t(q, c)$ , computes the joint probability distribution of given observation sequence  $o_{1:t}$  and hidden variables  $(q, c)$  condition on initial terminal of the model. The *normalization* phase of the *forward* flow considers the final terminal of the model due to the fact that a sequence has to be terminated in the final state with zero counter. This phase discards all paths that do not reach to the final terminal state from the calculation. In the first phase of the adapted *backward* algorithm, the *preliminary* calculation is performed to illustrate the existing paths that are laid between the initial and the final terminals. The *normalization* phase of the adapted *backward* flow computes the observing all future events  $o_{t+1:T}$  from the particular hidden variable  $(q, c)$  while considering both terminal states of the model. Those both algorithms has  $\mathcal{O}(N^2MT)$  computational complexity where it provides an feasible calculation to the model to compute the joint probability of hidden and observed variables and also it provides an quadratic computation for parsing the OCLs instead cubic time complexity.

The adapted version of *Baum-Welch* algorithm, which is a generalization of the EM algorithm, is used to find the maximum likelihood estimate the parameters of the model. The original version of this algorithm is also used as the same manner in a *HMM*. The adapted version of this algorithm consists of two steps which are called *E-Step* and *M-step*, respectively. In *E-Step*, the number of being in hidden state-counter pairs and the number of transitions between two particular hidden variables are found with two different calcu-

lations which are denoted as  $\gamma$  and  $\xi$ , respectively. Those both calculations are dominated by both the *forward* and *backward* flows. Thus, the complexity for one iteration of the adapted version of *Baum-Welch* algorithm is also quadratic as adapted versions of *forward* and *backward* are. In *M-Step*, updating formulas of the parameters of the model are found by using *Lagrange* multiplier.

The only one initial and one final terminal are used in this model type. However, various other models can be generated by changing the definition of the initial and terminal state of such model while the rest of the structure of the model are kept the same. In the next chapter, five such alternative models will be presented.

## Chapter 5

# Characteristic Feature HP1CA Models

In this chapter, we study the other classes of *HP1CA* which differ on the definition of the initial and the final states and their relation to the adapted algorithms. In the previous chapter, we have investigated the Model-1 type of *HP1CA* that has only one initial terminal and one final terminal states and adaption the algorithms of *HMM* to this type. We show that various new *HP1CA* models can be generated by augmenting the adjustable features of the terminals of the Model-1.

The new models are essentially derived from the Model-1, however, these models differs according to the characteristic feature of the multiple terminals or final terminal with non-zero counter value. We will define five different new *HP1CA* models.

In Section 5.1, we show a class of *HP1CA* can be augmented by applying multiple initial terminals as in *HMM*. The model is called Model-2 that contains multiple initial terminals and one unique final terminal. Each initial and terminal state has zero counter value. Each initial terminals have an initial probability distribution which is described by the initial vector that is denoted by  $\pi$ .

In Section 5.2, a model with multiple final terminals is described. This new model, which we call Model-3, contains only one initial terminal and multiple final terminals states. Of course, both initial and final states have counter equal to zero. In contrast with Model-1,

the transition system of Model-2 associated with *HP1CA* has many terminating points.

In Section 5.3, the multiple terminals are allows at both the initial and the final states of the model that is called Model-4. This is a property of all transition systems of Model-4, that the process can start from multiple starting points and terminate at many terminating points, according to the interpretation of multiple terminals.

In Section 5.4, we go beyond the main structure of the counter machines. This new model, Model-5, contains one unique initial and final state. However, although the counter value of the initial terminal is zero, the counter value of the final terminal may be different from zero in this model. A thorough investigation of the final states with the non-zero counter value allows for multiple terminating points of the process.

In Section 5.5, another new model, Model-6, contains multiple initial states and one final state with nonzero counter value. Indeed, it allows multiple starting points for transitions of the model and multiple branching terminating points that have a non-zero counter value.

## 5.1 Model-2: Multiple initial terminals and unique final terminal

In the previous chapter, we have described the Model-1 that has only one starting terminal with zero counter. Now, we will allow for multiple initial terminals. The interpretation of the initial vector has also been considered on *HMM* by assigned probability value to all feasible initial hidden states. The purpose is all states have a probability at initialization step instead of only one specific starting state, and the probability of the path which is used to reach a latent state at time  $t$  will differ with respect to the chosen starting point.

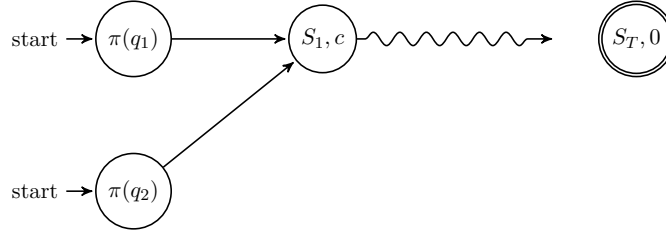


Figure 5.1: Two possible initial point for each designated paths for starting the distribution to reach the final state  $(S_T, 0)$

The adapted version of the initial vector considers that the hidden states store the counter value which must be zero at the initial step, unlike the original version. The set of all initial states are represented by  $Q_0$  and the unique final terminal is  $q_F$ . As seen in Figure 5.1, there are two initial terminals which are  $(q_1, 0)$  and  $(q_2, 0)$ . The probability distribution of the initial states is denoted as

$$\pi(q) = Pr(S_0 = q) \quad (5.1)$$

where  $q \in Q_0$ . This aspect obliges some configuration must be done for *forward*, *backward*, and *Baum-Welch* algorithms.

### 5.1.1 Applied an initial vector to adapted forward algorithm

One of fundamental algorithm of the *HMM forward* is successfully adapted to the *HP1CA* that make use of one counter. In this model, where there is more than one initial terminal instead of an individual initial state, the *forward* calculation is examined in two phases. Now we need to consider that adding an initial vector with multiple starting terminals will require a change to the existing adopted forward algorithm.

#### Preliminary forward calculation

This new *HP1CA* model has initial vector that consists the probability distribution of starting states. The preliminary calculation of the adapted forward is

$$\hat{\alpha}_t(S_t) = Pr(o_{1:t}, S_t | \pi(S_0)) \quad (5.2)$$

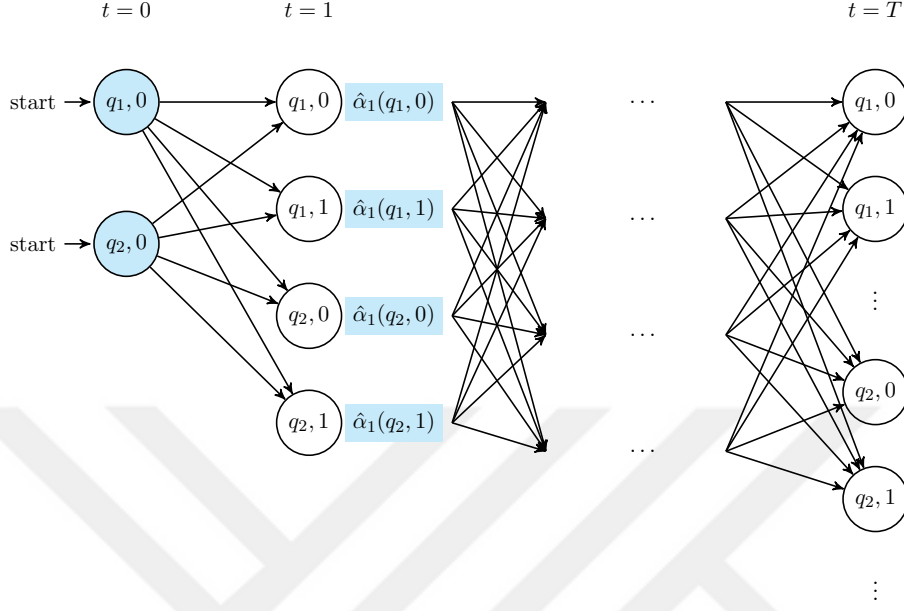


Figure 5.2: Two initial states which  $(q_1, 0)$  and  $(q_2, 0)$  yield multiple starting points for computation of state distribution.

The calculation consists of two steps. The first step is called *base* step where a multiplication is performed over the initial states at the time step  $t = 1$ . The second step performs the *recursive* calculation after *base* step until the length of the given observation sequence (i.e.  $t = T$ ).

- Base Step:

$$\hat{\alpha}_1(q, c) = \sum_{q_0 \in Q_0} \pi(q_0) A^0(q_0, \Delta, q) B(q, o_1) \quad (5.3)$$

- Recursion Step:

if  $c - \Delta = 0$  :

$$\hat{\alpha}_t(q, c) = \sum_{q' \in Q} \sum_{\Delta=0}^1 \hat{\alpha}_{t-1}(q', c - \Delta) A^0(q', \Delta, q) B(q, o_t) \quad (5.4)$$

if  $c - \Delta > 0$  :

$$\hat{\alpha}_t(q, c) = \sum_{q' \in Q} \sum_{\Delta=-1}^1 \hat{\alpha}_{t-1}(q', c - \Delta) A^+(q', \Delta, q) B(q, o_t) \quad (5.5)$$

The recursion part is the same as the recursion part of the Model-1 4.2.1. The following theorem proves this equality

**Theorem 5.1.1.** *In the Model-2 which is created by plugging an initial vector which consist of multiple initial terminals with zero counter value, the base step of the preliminary calculation of the forward flow is performed by summing over initial terminal but the recursion step is computed as same as the recursion part of the Model-1.*

*Proof.* The proof begins with plugging the time  $t = 1$  into the equation 5.2 where

$$\hat{\alpha}_1(q, c) = Pr(o_{1:1}, S_1 = (q, c) | \pi(S_0)) \quad (5.6)$$

After applying the sum rule over each initial terminal

$$\begin{aligned} \hat{\alpha}_1(q, c) &= \sum_{q_0 \in Q_0} Pr(o_1, S_1 = (q, c) | \pi(S_0) = q_0) \\ &= \sum_{q_0 \in Q_0} \pi(q_0) Pr(o_1, S_1 = (q, c) | S_0 = q_0) \end{aligned} \quad (5.7)$$

The rest of the proof of the base step calculation of the preliminary forward flow of Model-2 is now equivalently equal to the base step of the same calculation of the Model-1 4.2.1.  $Pr(o_1, S_1 = (q, c) | S_0 = q_0)$  is already proved that it is equal to multiplication of the state transition probability  $A^0(q_0, \Delta, q)$  and emission probability  $B(q, o_1)$  (i.e. for further detail 4.2.1).

The proof of the recursion part has is essentially the same as in theorem 4.2.1. Because the preliminary forward probabilities at the base step are held in  $\hat{\alpha}_1(q, c)$  for each hidden control states  $q \in \{q_1, q_2\}$  and counter values  $c \in \{0, 1\}$ . At the recursive time step, for instance, time  $t = 2$ , the preliminary forward probability of the current state-counter (i.e.  $(q, c)$ ) pairs are computed by summing the multiplication of previous preliminary forward probability (i.e.  $\hat{\alpha}_1(q', c')$ ), state transition, and emission probabilities over all previous state-counter pairs which reach the current state-pair. Thus, the recursion part of the preliminary forward calculation of Model-2 is

$$\hat{\alpha}_t(q, c) = \sum_{q' \in Q} \hat{\alpha}_{t-1}(q', c') A^{\delta(c')}(q', \Delta, q) B(q, o_t) \quad (5.8)$$

where  $c'$  is the previous counter value and it is equal to  $c - \Delta$ . If it is equal to zero, then  $\Delta \in \{0, 1\}$  and  $\delta(c') = 0$ . If  $c'$  is non-zero, then  $\Delta \in \{-1, 0, 1\}$  and  $\delta(c') = +$ .  $\square$

### Normalization of forward calculation

The normalization phase of the forward algorithm starts the calculations at the final state at time  $t = T$ . There is a unique final terminal with zero counter value in this model. Therefore, the calculation is adjusted by applying only initial vector.

$$\alpha_t(q, c) = Pr(o_{1:t}, S_t = (q, c) | \pi(S_0), S_T) \quad (5.9)$$

Two steps calculation is

- **Base Step:**

$$\alpha_T(q, c) = \begin{cases} \frac{\hat{\alpha}_T(q, c)}{\sum_{q_0 \in Q_0} \pi(q_0) Pr(S_T = (q_F, 0) | S_0 = q_0)} & : q = q_F \text{ and } c = 0 \\ 0 & : \text{otherwise} \end{cases}$$

- **Recursion Step:**

if  $c = 0$  :

$$\alpha_t(q, c) = \sum_{q' \in Q} \sum_{\Delta=0}^1 \frac{\alpha_{t+1}(q', c + \Delta) A^0(q, \Delta, q') \hat{\alpha}_t(q, c)}{\hat{\alpha}_{t+1}(q', c + \Delta)} \quad (5.10)$$

if  $c > 0$  :

$$\alpha_t(q, c) = \sum_{q' \in Q} \sum_{\Delta=-1}^1 \frac{\alpha_{t+1}(q', c + \Delta) A^+(q, \Delta, q') \hat{\alpha}_t(q, c)}{\hat{\alpha}_{t+1}(q', c + \Delta)} \quad (5.11)$$

The normalization calculation of Model-2 is almost the same as the normalization calculation of Model-1 just only the multiple initial terminals must be considered at the base step. At  $t = T$ ,

$$\begin{aligned} \alpha_T(q, c) &= Pr(o_{1:T}, S_T = (q, c) | \pi(S_0), S_T = (q_F, 0)) \\ &= \frac{Pr(o_{1:T}, S_T = (q, c), S_T = (q_F, 0) | \pi(S_0))}{Pr(S_T = (q_F, 0) | \pi(S_0))} \end{aligned} \quad (5.12)$$

If  $q = q_F$  and  $c = 0$  then

$$\begin{aligned}
 \alpha_T(q_F, 0) &= \frac{Pr(o_{1:T}, S_T = (q_F, 0) | \pi(S_0))}{Pr(S_T = (q_F, 0) | \pi(S_0))} \\
 &= \frac{\hat{\alpha}_T(q_F, 0)}{Pr(S_T = (q_F, 0) | \pi(S_0))} \\
 &= \frac{\hat{\alpha}_T(q_F, 0)}{\sum_{q_0 \in Q_0} \pi(q_0) Pr(S_T = (q_F, 0) | S_0 = q_0)}
 \end{aligned} \tag{5.13}$$

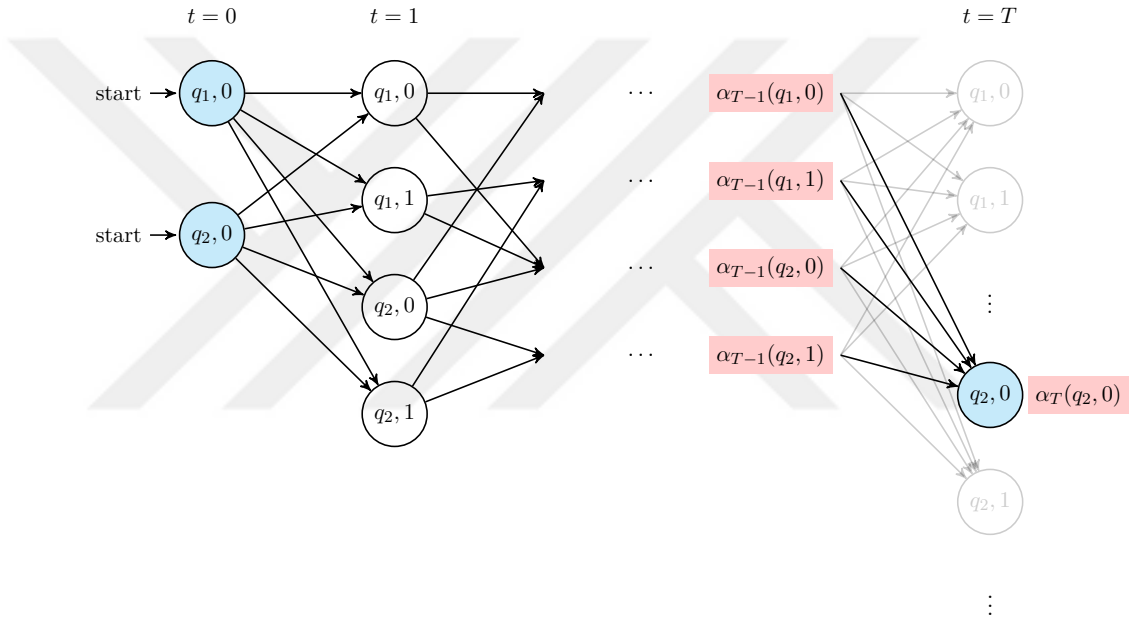


Figure 5.3: Two initial states which  $(q_1, 0)$  and  $(q_2, 0)$  yield multiple starting points for computation of state distribution.

The denominator of equation 5.13 is sum of the all paths along all over initial terminal to final terminal. However, total sum of the probability distribution of initial terminal of the model must equal to one. Therefore, the denominator will produce the same result as the denominator of the base step calculation of the normalization of the forward flow of the Model-1 4.2.1. The proof of the recursive part of the normalization calculation again has same proving steps as Model-1 had. As seen in Figure 5.3,  $\alpha_T(q_2, 0)$  keeps the normalization probability at the base step. The preliminary forward flow calculation is already done by considering the multiple initial terminal and it is held by  $\hat{\alpha}_t(q, c)$  for each

hidden state and counter pairs for each individual time step  $t$ . Hence, the normalization value of forward flow at time  $t = T - 1$  uses future normalization values, preliminary calculations, and state transition value which is enabled by considering counter value.

### 5.1.2 Applied initial vector to adapted backward algorithm

The second fundamental algorithm of the *HMM* is also successfully adapted to *HP1CA*. The *backward* calculation for Model-2 starts to the unique final terminal which has zero counter value  $(q_F, 0)$ . The backward flow computation consists two phases as the first model has and each phases begins at the end of the time interval  $t = T$  which is also equal to the length of the given observation sequence. Due to the backward probability of the current hidden state-counter pair condition on the future backward calculation, the time scale decreases at each time step and the calculation is made towards  $t = 1$ . At the last step (i.e.  $t = 0$ ), the initial vector must be considered for each phase of the *backward* calculation.

#### Preliminary backward calculation

The preliminary backward flow calculation of Model-2 is performed by the following equation with consideration of the initial vector

$$\hat{\beta}_t(q, c) = Pr(S_T = (q_F, 0) | S_t = (q, c), \pi(S_0)) \quad (5.14)$$

According to the *Markov* property the current state only depends on the previous state. Therefore, the preliminary backward probability of the final terminal is equal to one and the rest of the hidden state-counter pairs are zero at the base step  $t = T$ . This part is already proved in equation 4.57.

On the other hand, the product rule might be applied to the 5.14

$$\hat{\beta}_t(q, c) = \frac{Pr(S_T = (q_F, 0), S_t = (q, c) | \pi(S_0))}{Pr(S_t = (q, c) | \pi(S_0))} \quad (5.15)$$

At the base step  $t = T$ , if and only if  $q = q_F$  and  $c = 0$ , then

$$\hat{\beta}_T(q_F, 0) = \frac{Pr(S_T = (q_F, 0) | \pi(S_0))}{Pr(S_T = (q_F, 0) | \pi(S_0))} = 1 \quad (5.16)$$

The recursion part of this calculation has the same formula as the same part of the preliminary backward calculation of the Model-1. Let assume that the current state is  $S_t = (q, c)$  and the future state is  $S_{t+1} = (q', c')$ . In here, the future counter is equivalently equal to  $c + \Delta$  where the  $\Delta$  depends on the current counter value  $c$ . If  $c = 0$  then  $\Delta \in \{0, 1\}$ . If  $c > 0$ , then  $\Delta \in \{-1, 0, 1\}$ . The sum rule over the future hidden state-counter pair might be applied to the equation 5.14.

$$\hat{\beta}_t(q, c) = \sum_{S_{t+1}} Pr(S_T, S_{t+1} | S_t = (q, c), \pi(S_0)) \quad (5.17)$$

Applying the chain rule

$$\hat{\beta}_t(q, c) = \sum_{S_{t+1}} Pr(S_T | S_t, S_{t+1}, \pi(S_0)) Pr(S_{t+1} | S_t, S_0) \quad (5.18)$$

According to the *d-separation*  $S_T \perp\!\!\!\perp S_t | S_{t+1}$  at the first argument of the above equation.

$$\hat{\beta}_t(q, c) = \sum_{S_{t+1}} Pr(S_T | S_{t+1}, \pi(S_0)) Pr(S_{t+1} | S_t, \pi(S_0)) \quad (5.19)$$

The *Markov* property can be applied to the second argument of the equation 5.19

$$\hat{\beta}_t(q, c) = \sum_{S_{t+1}} Pr(S_T | S_{t+1}, \pi(S_0)) Pr(S_{t+1} | S_t) \quad (5.20)$$

The first argument of the equation 5.20 represents the future preliminary backward calculation for hidden state-counter pair  $S_{t+1}$ . The second argument is the state transition with considering the counter value of the hidden states. According to these equations and the proofs, the preliminary backward calculation for the Model-2 consists two steps which are

- *Base Step:*

$$\hat{\beta}_T(q, c) = \begin{cases} 1 : & q = q_F \text{ and } c = 0 \\ 0 : & \text{otherwise} \end{cases}$$

- *Recursion Step:*

if  $c = 0$  :

$$\hat{\beta}_t(q, c) = \sum_{q' \in Q} \sum_{\Delta=0}^1 \hat{\beta}_{t+1}(q', c + \Delta) A^0(q, \Delta, q') \quad (5.21)$$

if  $c > 0$  :

$$\hat{\beta}_t(q, c) = \sum_{q' \in Q} \sum_{\Delta=-1}^1 \hat{\beta}_{t+1}(q', c + \Delta) A^+(q, \Delta, q') \quad (5.22)$$

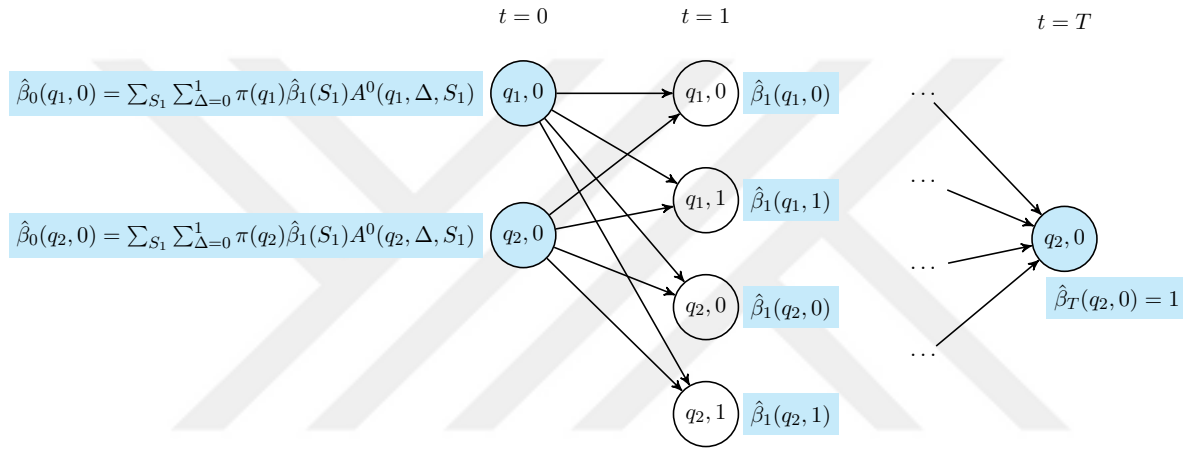


Figure 5.4: The calculation of preliminary backward flow at  $t = 0$ .

Since consideration of the multiple initial terminals in Model-2, at  $t = 0$ , each of the initial terminals has a preliminary backward value which is multiplied by the initial probability of that initial terminal as shown in Figure 5.4. In here  $S_1$  represents the future hidden state-counter pair which could reach to the designated initial terminal.

### Normalization of backward calculation

The normalization phase of the backward algorithm for the Model-2 is performed as the following equation

$$\beta_t(q, c) = Pr(o_{t+1:T} | S_t = (q, c), \pi(S_0), S_T = (q_F, 0)) \quad (5.23)$$

The output probability is considered in this phase as it is considered in the same calculation in Model-1 4.2.2. The calculation is performed beginning the last time step  $t = T$  as it is

done in the preliminary phase of the backward calculation. If the chain rule is applied to equation 5.23, then the following equation is achieved.

$$\beta_t(q, c) = \frac{Pr(o_{t+1:T}, S_t = (q, c), S_T = (q_F, 0) | \pi(S_0))}{Pr(S_t = (q, c), S_T = (q_F, 0) | \pi(S_0))} \quad (5.24)$$

The base step, at  $t = T$ ;

$$\begin{aligned} \beta_T(q, c) &= \frac{Pr(o_{T+1:T}, S_T = (q, c), S_T = (q_F, 0) | \pi(S_0))}{Pr(S_T = (q, c), S_T = (q_F, 0) | \pi(S_0))} \\ &= \frac{Pr(\emptyset, S_T = (q, c), S_T = (q_F, 0) | \pi(S_0))}{Pr(S_T = (q, c), S_T = (q_F, 0) | \pi(S_0))} \end{aligned} \quad (5.25)$$

If and only if  $q = q_F$  and the counter  $c = 0$  in the equation 5.25, then the beta value of  $(q, c)$  is equal to one, otherwise equals to zero.

$$\beta_T(q, c) = \begin{cases} 1 : & q = q_F \text{ and } c = 0 \\ 0 : & \text{otherwise} \end{cases}$$

The recursion part of the normalization has same calculation formulae as used in the normalization phase of the backward flow in Model-1 5.23. The recursion formulae is

$$\beta_t(q, c) = \sum_{S_{t+1}} \beta_{t+1}(S_{t+1}) A^{\delta(c)}(S_t, \Delta, S_{t+1}) B(S_{t+1}, o_{t+1}) \frac{\hat{\beta}_{t+1}(S_{t+1})}{\hat{\beta}_t(S_t)} \quad (5.26)$$

where

$$\delta(c) = \begin{cases} 0 : & c = 0 \\ + : & c > 0 \end{cases}$$

In the equation 5.26,  $S_t$  represents the current hidden state-counter pair (i.e.  $(q, c)$ ), and  $S_{t+1}$  is representing the future hidden state-counter pair. This equation is also equivalently equal to the following calculation where the initial vector is applied.

$$\beta_t(q, c) = \sum_{S_{t+1}} Pr(o_{t+2:T} | S_{t+1}, \pi(S_0), S_T) Pr(S_{t+1} | S_t) Pr(o_{t+1} | S_{t+1}) \frac{Pr(S_T | S_{t+1}, \pi(S_0))}{Pr(S_T | S_t, \pi(S_0))} \quad (5.27)$$

The proof of the equation 5.27 has same proof steps which are already done in proof 4.2.2. At  $t = 0$ , the normalization value of the initial terminals are calculated by multiplying the initial distribution probability of the current initial terminal.

$$\beta_0(q, 0) = \sum_{S_1} \sum_{\Delta=1}^1 \pi(q) \beta_1(S_1) A^0(q, \Delta, S_1) B(S_1, o_1) \frac{\hat{\beta}_1(S_1)}{\hat{\beta}_t(q)} \quad (5.28)$$

where  $q \in Q_0$  and  $S_1$  is the hidden state-counter pair at  $t = 1$ .

### 5.1.3 Applied initial vector to adapted Baum-Welch algorithm

In sub-section 4.2.3, the adapted version of *Baum-Welch* consists two part calculations which are **E-step** and **M-step**. In the expectation step, the total number of visiting a particular state-counter pair and total number of particular paths that come from the particular state-counter pair are calculated. These two calculations consider the terminals of the model while performing the *E-step*.

$\gamma_t(q, c)$  calculates the probability of being in hidden state-counter pair  $(q, c)$  at time step  $t$ , given observation sequence  $O$  and the terminals  $\pi(S_0)$ ,  $S_T = (q_F, 0)$

$$\gamma_t(q, c) = Pr(S_t = (q, c) | O, \pi(S_0), S_T) \quad (5.29)$$

After applying the chain rule to 5.29, then the following equation achieved

$$\gamma_t(q, c) = \frac{Pr(O, S_t, S_T | \pi(S_0))}{Pr(O, S_T | \pi(S_0))} \quad (5.30)$$

$O$  is representing the whole observation sequence. This sequence could be separated into two splits likewise  $(o_{1:t})$ , and  $(o_{t+1:T})$ . If this expanding of observation sequence is applied to the numerator of equation 5.30;

$$\gamma_t(q, c) = \frac{Pr(o_{1:t}, o_{t+1:T}, S_t, S_T | \pi(S_0))}{Pr(O, S_T | \pi(S_0))} \quad (5.31)$$

The product rule can be applied

$$\gamma_t(q, c) = \frac{Pr(o_{1:t}, o_{t+1:T}, S_T | S_t, \pi(S_0)) Pr(S_t | \pi(S_0))}{Pr(O, S_T | \pi(S_0))} \quad (5.32)$$

According to *d-separation*, the first piece of the observation sequence ( $o_{1:t}$ ) is independent with future observation sequence ( $o_{t+1:T}$ ) and final terminal  $S_T$  condition on the particular hidden state-counter pair  $S_t$ ;

$$\gamma_t(q, c) = \frac{Pr(o_{1:t}|S_t, \pi(S_0))Pr(o_{t+1:T}, S_T|S_t, \pi(S_0))Pr(S_t|\pi(S_0))}{Pr(O, S_T|\pi(S_0))} \quad (5.33)$$

The product rule can be applied again to second item of the equation 5.33

$$\gamma_t(q, c) = \frac{Pr(o_{1:t}|S_t, \pi(S_0))Pr(o_{t+1:T}|S_t, \pi(S_0), S_T)Pr(S_T|S_t, \pi(S_0))Pr(S_t|\pi(S_0))}{Pr(O, S_T|\pi(S_0))} \quad (5.34)$$

The product rule is applied to both the first and the last arguments of the equation 5.34

$$\gamma_t(q, c) = \frac{Pr(o_{1:t}, S_t|\pi(S_0))Pr(o_{t+1:T}|S_t, \pi(S_0), S_T)Pr(S_T|S_t, \pi(S_0))}{Pr(O, S_T|\pi(S_0))} \quad (5.35)$$

All the items of the numerator of the equation 5.35 are equal to the definitions of preliminary *forward* calculation in 5.2, normalization of *backward* flow in 5.23, and preliminary *backward* calculation in 5.14. The denominator of the same equation is equal to the preliminary forward probability of the final terminal. The new form of this equation is

$$\gamma_t(q, c) = \frac{\hat{\alpha}_t(q, c)\beta_t(q, c)\hat{\beta}_t(q, c)}{\hat{\alpha}_t(q_F, 0)} \quad (5.36)$$

The  $\gamma$  calculation of the Model-2 equivalently has the same formula of the same calculation in Model-1 4.107. The second calculation of the *E-step* is regarded to find the probabilities of the particular paths that existing between two particular hidden state-counter pairs. This calculation is denoted as it is done in Theorem 4.2.6.

$$\xi_t(S_{t-1}, S_t) = Pr(S_{t-1}, S_t|O, \pi(S_0), S_T) \quad (5.37)$$

where  $S_{t-1}$  represents the particular hidden variable that has a transition to the next hidden variable  $S_{t+1}$ . The same steps will be proceeded as it is done while proving the Theorem 4.2.6 and the following equation is strongly achieved that is equal to the equation 4.120

$$\xi_t((q', c'), (q, c)) = \frac{\hat{\alpha}_{t-1}(q', c')\beta_t(q, c)\hat{\beta}_t(q, c)A^{\delta(c')}(q', \Delta, q)B(q, o_t)}{Pr(O, S_T|S_0)} \quad (5.38)$$

where

$$\delta(c') = \begin{cases} 0 & : c' = 0 \\ + & : c' > 0 \end{cases}$$

In *M-step*, the calculation of the joint probability distribution over hidden variables and observed information is defined in equation 4.122. However, the initial vector must be considered to this calculation for Model-2. The joint probability is calculated as

$$Pr(S, C, \mathcal{O}|\theta) = \prod_{d=1}^D \left( \pi(S_0^{(d)}) B(S_1^{(d)}, o_1^{(d)}) A^0(S_0^{(d)}, C_1^{(d)}, S_1^{(d)}) \right. \\ \left. \prod_{t=2}^T B(o_t^{(d)}, S_t^{(d)}) A^{\delta(C_t^{(d)})}(S_t^{(d)}, C_{t+1}^{(d)} - C_t^{(d)}, S_{t+1}^{(d)}) \right) \quad (5.39)$$

Assuming multiple sequences are used in Model-2 as it is used in Model-1 where it is represented as  $\mathcal{O} = (O^{(1)}, O^{(2)}, \dots, O^{(D)})$ . If the logarithm of the equation 5.39 is taken, the following long equation is achieved;

$$\log Pr(S, C, \mathcal{O}|\theta) = \sum_{d=1}^D \left( \log \pi(S_0^{(d)}) + \sum_{t=1}^T \log B(o_t^{(d)}, S_t^{(d)}) \right. \\ + \sum_{\{t|C_t^{(d)}=0\}} \log A^0(S_t^{(d)}, C_{t+1}^{(d)} - C_t^{(d)}, S_{t+1}^{(d)}) \\ \left. + \sum_{\{t|C_t^{(d)}>0\}} \log A^+(S_t^{(d)}, C_{t+1}^{(d)} - C_t^{(d)}, S_{t+1}^{(d)}) \right) \quad (5.40)$$

If the equation 5.40 is plugged into equation 4.122 to find the maximized new parameters of the model.

$$\hat{Q}(\theta, \theta') = \sum_{S \in \mathcal{S}} \sum_{d=1}^D \log \pi(S_0^{(d)}) Pr(S|\mathcal{O}, \theta') + \sum_{S \in \mathcal{S}} \sum_{d=1}^D \sum_{t=1}^T \log B(o_t^{(d)}, S_t^{(d)}) Pr(S|\mathcal{O}, \theta') \\ + \sum_{S \in \mathcal{S}} \sum_{C \in \mathcal{C}} \sum_{d=1}^D \sum_{\{t|C_t^{(d)}=0\}} \log A^0(S_t^{(d)}, C_{t+1}^{(d)} - C_t^{(d)}, S_{t+1}^{(d)}) Pr(S|\mathcal{O}, \theta') \\ + \sum_{S \in \mathcal{S}} \sum_{C \in \mathcal{C}} \sum_{d=1}^D \sum_{\{t|C_t^{(d)}>0\}} \log A^0(S_t^{(d)}, C_{t+1}^{(d)} - C_t^{(d)}, S_{t+1}^{(d)}) Pr(S|\mathcal{O}, \theta') \quad (5.41)$$

Before applying *Lagrange* multiplier it is worth that to specify the sum of the initial distribution probability is equal to one;

$$\sum_{q \in Q_0} \pi(q) = 1 \quad (5.42)$$

If *Lagrange* multiplier is applied to equation 5.41,

$$\begin{aligned} \mathcal{L}(\theta, \theta', \eta, \lambda, \epsilon^0, \epsilon^+) &= \mathcal{Q}(\theta, \theta') - \eta_q \left( \sum_{q \in Q_0} \pi(q) - 1 \right) - \sum_q \lambda_q \left( \sum_o B(o, q) - 1 \right) \\ &\quad - \sum_q \epsilon_q^0 \left( \sum_{\Delta=0}^1 \sum_{q'} A^0(q, \Delta, q') - 1 \right) \\ &\quad - \sum_q \epsilon_q^+ \left( \sum_{\Delta=-1}^1 \sum_{q'} A^+(q, \Delta, q') - 1 \right) \end{aligned} \quad (5.43)$$

The upgrade formulas of both the state transitions and emission matrices of this model will be the same formulae as the Model-1 has. In this case, only the initial vector part should be considered in equation 5.43. If the partial derivation of equation 5.43 is taken over  $\pi(q)$ , then

$$\begin{aligned} \frac{\partial \mathcal{L}(\theta, \theta')}{\partial \pi(q)} &= \frac{\partial}{\partial \pi(q)} \left( \sum_{S \in \mathcal{S}} \sum_{d=1}^D \log \pi(S_0^{(d)}) Pr(S|\mathcal{O}, \theta') \right) - \eta_q = 0 \\ &= \sum_{d=1}^D \frac{Pr(S_0^{(d)} = q|\mathcal{O}, \theta')}{\pi(q)} - \eta_q = 0 \end{aligned} \quad (5.44)$$

If the partial derivation of equation 5.43 is taken over  $\eta_q$ , then

$$\frac{\partial \mathcal{L}(\theta, \theta')}{\partial \eta_q} = - \left( \sum_{q \in Q_0} \pi(q) - 1 \right) = 0 \quad (5.45)$$

Some algebra yields

$$\begin{aligned}
&= 1 - \sum_{q \in Q_0} \pi(q) \\
&= 1 - \sum_{q \in Q_0} \frac{1}{\eta_q} \sum_{d=1}^D \Pr(S_0^{(d)} = q | \mathcal{O}, \theta') \equiv 0 \\
\eta_q &= \sum_{q \in Q_0} \sum_{d=1}^D \Pr(S_0^{(d)} = q | \mathcal{O}, \theta') = \sum_{d=1}^D \sum_{q \in Q_0} \Pr(S_0^{(d)} = q | \mathcal{O}, \theta')
\end{aligned} \tag{5.46}$$

If the equation 5.46 is plugged into the equation 5.44, then the formulae of the upgrading the initial vector is obtained;

$$\hat{\pi}(q) = \frac{\sum_{d=1}^D \Pr(S_0^{(d)} = q | \mathcal{O}, \theta')}{\sum_{d=1}^D \sum_{q \in Q_0} \Pr(S_0^{(d)} = q | \mathcal{O}, \theta')} \tag{5.47}$$

The multiple observation sequence yields and  $\theta' = \pi(S_0), S_T$

$$\hat{\pi}(q) = \frac{\sum_{d=1}^D \Pr(S_0^{(d)} = q | O^{(d)}, \pi(S_0), S_T)}{\sum_{d=1}^D \sum_{q \in Q_0} \Pr(S_0^{(d)} = q | O^{(d)}, \pi(S_0), S_T)} \tag{5.48}$$

The denominator of the equation 5.48 is equivalently equal to dimension of the observation sequences  $D$ . Because,  $\sum_{q \in Q_0} \Pr(S_0^{(d)} = q | O^{(d)}, \pi(S_0), S_T)$  means that the sum of the all probability of being in initial terminal at  $t = 0$ .

$$\sum_{q \in Q_0} \Pr(S_0^{(d)} = q | O^{(d)}, \pi(S_0), S_T) = \sum_{q \in Q_0} \gamma_0^{(d)}(q, 0) = 1 \tag{5.49}$$

The last form of the initial vector updating formulae is

$$\hat{\pi}(q) = \frac{\sum_{d=1}^D \gamma_0^{(d)}(q, 0)}{D} \tag{5.50}$$

The rest of the equation 5.43 is following the same steps for achieving the same updating formulas for transition and emission matrices as they are done in Model-1.

## 5.2 Model-3: Multiple final terminals and unique initial terminal

A thorough investigation of the structure of multiple terminals shows us that we can establish this approach to the final terminal. This model consists multiple final terminals which have zero counter value and only one initial terminal.  $Q_F$  represents the set of the final terminals of the model where

$$Q_F = \{(q, c) | c = 0\}$$

The initial terminal is again denoted by  $q_0$ . The final state can be among of the set of hidden states of the model  $q \in Q$ . This multiple aspect allows that a particular branching can be terminated in one of the accepting states.

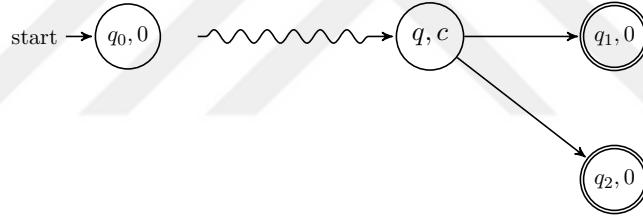


Figure 5.5: Two possible initial point for each designated paths for starting the distribution to reach the final state

The processing of the transition system starts the initial terminal with zero counter value  $(q_0, 0)$  at  $t = 0$ . The counter value still establishes to enable the transition matrices throughout the time interval (i.e. processing the sequence) as this is done in previous models. The model can be terminated at one of the final terminals of the model at time  $t = T$ .

The adapted version of the algorithms can be applied to this model. The first phase of the adapted forward algorithm is the only calculation that is not depending on the final terminal. The other crucial calculations must consider the multiple final terminals while computing the probability of the given observation sequence.

### 5.2.1 Applying adapted *forward* algorithm to Model-3

The both phases of the adapted forward algorithm can be applied for multiple final terminals.

#### Preliminary forward calculation

The calculation of preliminary forward flow is naturally processed due to there is no contribution by the final terminal. As it has been introduced before, the calculation of the preliminary forward flow is

$$\hat{\alpha}_t(S_t) = Pr(o_{1:t}, S_t | S_0) \quad (5.51)$$

The proof of this calculation has the same steps as they are successfully done in Model-1. As seen in Figure 5.6, the preliminary forward flow (i.e.  $\hat{\alpha}$ ) holds the probability of the particular hidden state-counter pairs while emitting one output symbol for each time step  $t$ . The point to be taken into consideration in this calculation is that there is more than one final terminal with zero counter which are illustrated with cyan color at time  $t = T$  in Figure 5.6.

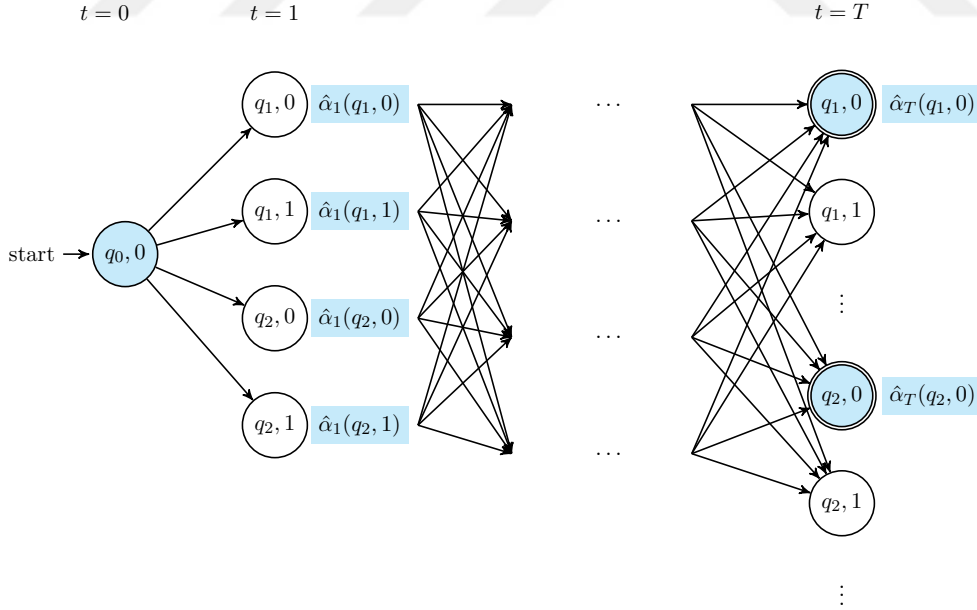


Figure 5.6: Two final states which  $(q_1, 0)$  and  $(q_2, 0)$  yield multiple ending points for the computation of state distribution.

### Normalisation of forward calculation

The normalization of forward flow calculates:

$$\alpha_t(q, c) = Pr(o_{1:t}, S_t = (q, c) | S_0, S_T \in Q_F) \quad (5.52)$$

where  $Q_F = \{(q, c) | c = 0\}$  and  $q \in Q$ .

The calculation is started at the final time step  $T$ . However, there is more than one final states that provide multiple starting points to normalize the forward calculation.

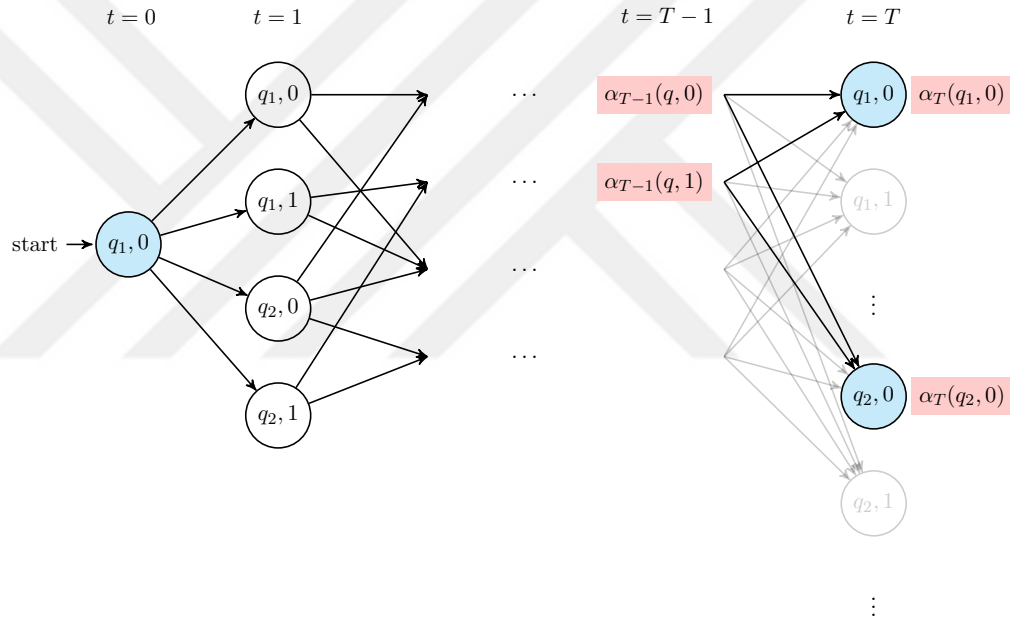


Figure 5.7: Two final states which  $(q_1, 0)$  and  $(q_2, 0)$  yield multiple starting points for computation of forward flow normalization.

At the base step  $t = T$ , the calculation of the normalization after plugged  $t = T$  into equation 5.52 is ;

$$\alpha_T(q, c) = Pr(o_{1:T}, S_T = (q, c) | S_0, S_T \in Q_F) \quad (5.53)$$

We can apply product rule

$$\alpha_T(q, c) = \frac{Pr(o_{1:T}, S_T = (q, c), S_T \in Q_F | S_0)}{Pr(S_T \in Q_F | S_0)} \quad (5.54)$$

If  $q \in Q_F$  and  $c = 0$  then

$$\begin{aligned}
 \alpha_T(q, 0) &= \frac{Pr(o_{1:T}, S_T = (q, 0) \in Q_F, S_T \in Q_F | S_0)}{Pr(S_T \in Q_F | S_0)} \\
 &= \frac{Pr(o_{1:T}, S_T = (q, 0) \in Q_F | S_0)}{Pr(S_T = (q, 0) \in Q_F | S_0)} \\
 &= \frac{\hat{\alpha}_T(q, 0)}{\sum_{q' \in Q_F} Pr(S_T = (q', 0) | S_0)}
 \end{aligned} \tag{5.55}$$

According to the equation 5.55, the difference between Model-2 and this model type is the denominator of the base step calculation. In Model-2, the summing was over the multiple initial states with considering the probability of initial distribution. But in this model, the summing is over all particular accepting states instead of initial terminals.

In the recursion part of the second phase of the forward flow, the calculation is denoted as:

$$\alpha_t(q, c) = Pr(o_{1:t}, S_t = (q, c) | S_0, S_T \in Q_F) \tag{5.56}$$

This equation is equal to:

$$\begin{aligned}
 \alpha_t(S_t) &= \sum_{S_{t+1}} \frac{\alpha_{t+1}(S_{t+1}) A^{\delta(c)}(S_t, \Delta, S_{t+1}) \hat{\alpha}_t(S_t)}{\hat{\alpha}_{t+1}(S_{t+1})} \\
 &= \sum_{S_{t+1}} \frac{Pr(o_{1:t+1}, S_{t+1} | S_0, S_T \in Q_F) Pr(S_{t+1} | S_t) Pr(o_{1:t}, S_t | S_0)}{Pr(o_{1:t+1}, S_{t+1} | S_0)}
 \end{aligned} \tag{5.57}$$

The proving of the normalization calculation of the Model-3 has the same proof steps as the previous models have. However, at time  $t = T - 1$ , the sum of the future normalization value of the existing final states are taken due to this model consists of multiple final terminals. For instance, let the particular hidden state-counter pair is  $(q, c)$  at time step  $T - 1$ . At the future time step (i.e.  $t = T$ ) the position of the hidden state-counter pair is being in one of the final terminals as shown in Figure 5.7. According to the feature of the final terminal of this model, the counter value is zero while being in final terminals. Therefore, the counter value of the current particular hidden variable must be zero or one where it can be shown as  $c \in \{0, 1\}$ .

Let assume that the counter value  $c = 1$  for the hidden variable  $q$  at  $t = T - 1$ . The normalization value of this particular hidden variable is calculated as

$$\alpha_{T-1}(q, 1) = \frac{\alpha_T(q_1, 0)A^+(q, -1, q_1)\hat{\alpha}_{T-1}(q, 0)}{\hat{\alpha}_T(q_1, 0)} + \frac{\alpha_T(q_2, 0)A^+(q, -1, q_2)\hat{\alpha}_{T-1}(q, 0)}{\hat{\alpha}_T(q_2, 0)}$$

We conjecture that the rest of the calculation is naturally processed until time step  $t = 0$ . Because, after the calculation of the normalization is successfully done at  $t = T - 1$ , then the recursion procedure normally processes through the existing path up to time  $t = 0$ .

### 5.2.2 Applying adapted *backward* algorithm to Model-3

Another dynamic programming algorithm is adopted version of *backward* algorithm that could be applied to Model-3 to compute the probability of the given observation sequence via backward flow with two phases.

#### Preliminary backward calculation

The first phase of the backward calculation considers the multiple final terminals which means that the computation of the backward flow might start in one of the final terminals of the model.

$$\hat{\beta}_t(q, c) = Pr(S_T \in Q_F | S_t = (q, c), S_0) \quad (5.58)$$

This first phase of the calculation consists two main steps which are *base* and *recursion* as the previous models have. At the base step (i.e. at  $t = T$ ),

$$\begin{aligned} \hat{\beta}_T(q, c) &= Pr(S_T \in Q_F | S_T = (q, c), S_0) \\ &= Pr(S_T \in Q_F | S_T = (q, c)) \\ &= 1 \end{aligned} \quad (5.59)$$

where if  $q \in Q_F$  and  $c = 0$ , otherwise it is equal to zero. The initial terminal is removed in the second line of the equation 5.59 according to *Markov* property. At the base step, the preliminary backward probability of each final terminal of the model equal to one as shown in Figure 5.8. The recursion part of the preliminary backward calculation has the

same formulae as the previous models have:

$$\hat{\beta}_t(S_t) = \sum_{S_{t+1}} \hat{\beta}_{t+1}(S_{t+1}) A^{\delta(c)}(S_t, \Delta, S_{t+1}) \quad (5.60)$$

where  $S_t$  represents the current state-counter pair  $(q, c)$ , and  $S_{t+1}$  represents the future state-counter pair  $(q', c')$  that  $c'$  is actually equal to  $c + \Delta$ . The transition matrices could be enabled by depending on the current counter value  $c$  where

$$\delta(c) = \begin{cases} 0 & : c = 0 \\ + & : c > 0 \end{cases}$$

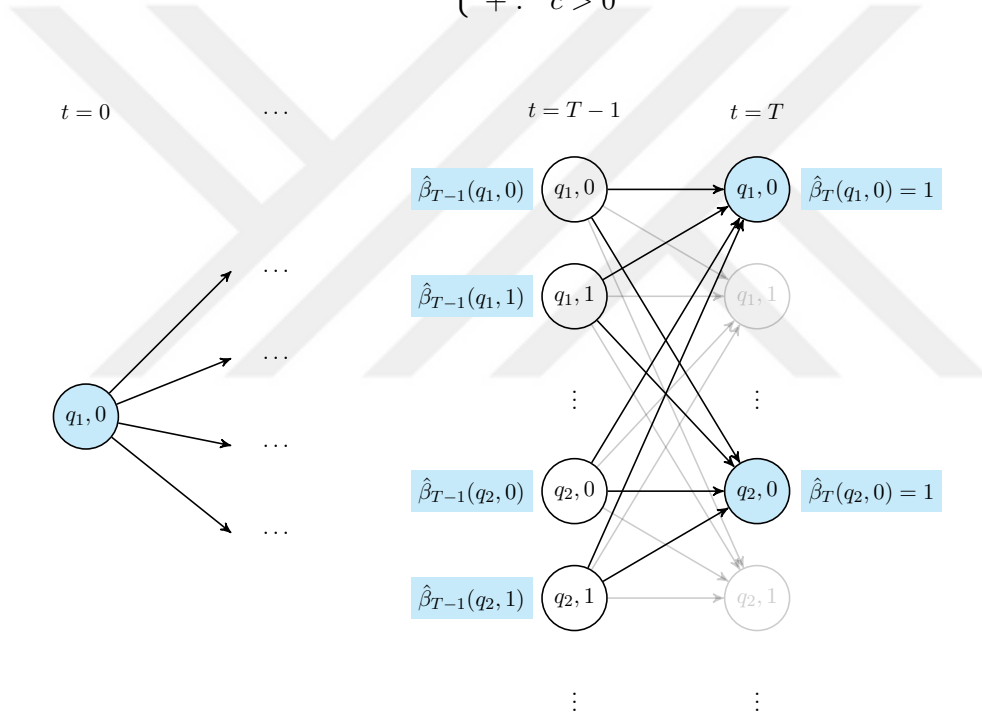


Figure 5.8: The calculation of preliminary backward flow at  $t = T - 1$ .

The recursion calculation is performed by summing all the multiplication of the preliminary *backward* probability of the future hidden variables with the transition probability of the paths that link to all final terminals of the model. For instance, preliminary *backward* value of the hidden variable  $(q_1, 0)$  at  $T - 1$  is calculated by

$$\hat{\beta}_{T-1}(q_1, 0) = \hat{\beta}_T(q_1, 0) A^0(q_1, 0, q_1) + \hat{\beta}_T(q_2, 0) A^0(q_1, 0, q_2)$$

As seen that the recursion part of the preliminary *backward* computation could be performed by marginalizing over the future state  $S_{t+1}$  on the equation 5.58;

$$\hat{\beta}_t(S_t) = \sum_{S_{t+1}} Pr(S_T \in Q_F, S_{t+1} | S_t, S_0) \quad (5.61)$$

The chain rule could be applied as

$$\hat{\beta}_t(S_t) = \sum_{S_{t+1}} Pr(S_T \in Q_F | S_{t+1}, S_t, S_0) Pr(S_{t+1} | S_t, S_0) \quad (5.62)$$

According to Markov property; the state only depends on the previous state. Therefore, the last item of equation 5.62 the future hidden variable  $S_{t+1}$  depends only on the closest hidden variable which is  $S_t$ ;

$$\hat{\beta}_t(S_t) = \sum_{S_{t+1}} Pr(S_T \in Q_F | S_{t+1}, S_t, S_0) Pr(S_{t+1} | S_t) \quad (5.63)$$

In proportion to *d - separation*,  $S_T \in Q_F \perp\!\!\!\perp S_t | S_{t+1}$

$$\hat{\beta}_t(S_t) = \sum_{S_{t+1}} Pr(S_T \in Q_F | S_{t+1}, S_0) Pr(S_{t+1} | S_t) \quad (5.64)$$

The first argument of equation 5.64 is equal to  $\hat{\beta}_{t+1}(S_{t+1})$  at time  $t + 1$ . The last item of the same equation is the state transition probability that depends the counter value;

$$\hat{\beta}_t(S_t) = \sum_{S_{t+1}} \hat{\beta}_{t+1}(S_{t+1}) A^{\delta(c)}(S_t, \Delta, S_{t+1}) \quad (5.65)$$

As it is seen that the obtained formulae of the preliminary *backward* flow calculation of this model is same as the formulae used for the same calculation of the previous models.

### Normalize Backward

The second phase of the adopted *backward* algorithm is normalized the probability of the preliminary calculation by considering the emission probability distribution. The normalization calculation is represented as

$$\beta_t(q, c) = Pr(o_{t+1:T} | S_t = (q, c), S_0, S_T \in Q_F) \quad (5.66)$$

At the *base* step of the normalization, only the probability of the final terminals is equal to one. The proof of this information is already done in the same calculation of the Model-2 in equation 5.25. The only difference is that there are multiple final states instead of multiple initial terminals.

The formula used in the *recursion* step for this model is also the same equation as used in Model-2. Again, the reason is this model has multiple final terminals instead of multiple initial terminals. In this calculation, when the previous models are taken into consideration, preliminary backward flow calculation is included as

$$\beta_t(S_t) = \sum_{S_{t+1}} Pr(o_{t+2:T}|S_{t+1}, S_0, S_T \in Q_F) Pr(S_{t+1}|S_t) Pr(o_{t+1}|S_{t+1}) \frac{Pr(S_T \in Q_F|S_{t+1}, S_0)}{Pr(S_T \in Q_F|S_t, S_0)} \quad (5.67)$$

where  $S_t$  is the current state-counter pair and  $S_{t+1}$  is the future hidden variable. The proof steps could be same as it is already done in Chapter 4 in sub-section 4.2.2. Therefore, the normalization calculation of *backward* flow of Model-3 has two steps calculation where

- *Base Step*:

$$\beta_T(q, c) = \begin{cases} 1 : & q \in Q_F \text{ and } c = 0 \\ 0 : & \text{otherwise} \end{cases}$$

- *Recursion Step*:

if  $c = 0$  :

$$\beta_t(q, c) = \sum_{q' \in Q} \sum_{\Delta=0}^1 \beta_{t+1}(q', c + \Delta) A^0(q, \Delta, q') B(q', o_{t+1}) \frac{\hat{\beta}_{t+1}(q', c + \Delta)}{\hat{\beta}_t(q, c)} \quad (5.68)$$

if  $c > 0$  :

$$\beta_t(q, c) = \sum_{q' \in Q} \sum_{\Delta=-1}^1 \beta_{t+1}(q', c + \Delta) A^0(q, \Delta, q') B(q', o_{t+1}) \frac{\hat{\beta}_{t+1}(q', c + \Delta)}{\hat{\beta}_t(q, c)} \quad (5.69)$$

### 5.2.3 Applying adapted *Baum-Welch* algorithm to Model-3

The adapted version of this algorithm consists of two main calculations which are the expectation calculation that is denoted as ***E-step*** and maximization calculation which is

denoted as ***M-step***. The purpose of this algorithm is used in this model as it is used in the same manner in the previous models.

In *E-Step*,  $\gamma_t(q, c)$  calculates the probability of being in hidden variable  $(q, c)$  at time step  $t$ , depending on the given observation sequence  $O$ , initial terminal  $S_0 = (q_0, 0)$ , and multiple final terminals  $S_T \in Q_F$  where

$$\gamma_t(q, c) = Pr(S_t = (q, c) | O, S_0, S_T \in Q_F) \quad (5.70)$$

The same proof steps of the Model-2 in sub-section 5.1.3 could be applied to equation 5.70 and the following equation is obtained

$$\gamma_t(S_t) = \frac{Pr(o_{1:t}, S_t | S_0) Pr(o_{t+1:T} | S_t, S_0, S_T \in Q_F) Pr(S_T \in Q_F | S_t, S_0)}{Pr(O, S_T \in Q_F | S_0)} \quad (5.71)$$

The first argument of the numerator of the equation 5.71 is the definition of the preliminary calculation of *forward* flow in equation 5.51, the second argument is the normalization calculation of *backward* flow in equation 5.66, and the last item is the definition of the preliminary calculation of *backward* flow in equation 5.58, respectively.

$$\gamma_t(S_t) = \frac{\hat{\alpha}_t(S_t) \beta_t(S_t) \hat{\beta}_t(S_t)}{Pr(O, S_T \in Q_F | S_0)} \quad (5.72)$$

The denominator of the equation 5.72 is the sum of the preliminary *forward* flow probability of all designated final terminals of the model. If expand the observation sequence  $O$  as  $o_{1:T}$  then the denominator of the equation 5.72 would be changed as

$$Pr(o_{1:T}, S_T \in Q_F | S_0) \quad (5.73)$$

The equation 5.73 is equal to the calculation of the preliminary *forward* when  $t = T$  puts into the equation 5.51

$$\hat{\alpha}_T(q, c) = Pr(o_{1:T}, S_T = (q, c) | S_0) \quad (5.74)$$

If and only if the particular hidden state  $q$  is the one of the final terminal from the set of final terminal  $Q_F$  and the counter value  $c$  equals to zero, then the equation 5.73 is equal

to sum of the probability of preliminary *forward* of all final terminals.

$$\gamma_t(S_t) = \frac{\hat{\alpha}_t(S_t)\beta_t(S_t)\hat{\beta}_t(S_t)}{\sum_{S_T \in Q_F} \hat{\alpha}_T(S_T, 0)} \quad (5.75)$$

The other calculation of the *E-step* is

$$\xi_t(S_{t-1}, S_t) = Pr(S_{t-1}, S_t | O, S_0, S_T \in Q_F) \quad (5.76)$$

The proof of this calculation is also has the same steps as Model-1 has. However, only the denominator of the calculation must consider the multiple final terminals as the first calculation of *E-step* of this model.

$$\xi_t((q', c'), (q, c)) = \frac{\hat{\alpha}_{t-1}(q', c')\beta_t(q, c)\hat{\beta}_t(q, c)A^{\delta(c')}(q', \Delta, q)B(q, o_t)}{\sum_{S_T \in Q_F} \hat{\alpha}_T(S_T, 0)} \quad (5.77)$$

where

$$\delta(c') = \begin{cases} 0 & : c' = 0 \\ + & : c' > 0 \end{cases}$$

In *M-step*, the calculation of the the joint probability distribution is defined as

$$Pr(S, C, O | \theta) = \prod_{d=1}^D \prod_{t=1}^T B(o_t^{(d)}, S_t^{(d)}) A^{\delta(C_t^{(d)})}(S_t^{(d)}, C_{t+1}^{(d)} - C_t^{(d)}, S_{t+1}^{(d)}) \quad (5.78)$$

The formulas used in Model-1 to estimate the parameters of the model are eligible in this model since the multiple final terminals have no effect such as the initial vector in the calculation of the joint probability distribution as shown in the equation 5.78. Because until this stage, the calculation of  $\gamma$  and  $\xi$  have already obtained for this model, and the available update formulas fully surrounded by  $\gamma$  and  $\xi$  calculations as it is discussed in Chapter 4.

### 5.3 Model-4: Initial vector and multiple final terminals

Based on the structure of both Model-2 and Model-3 we can build another *HPICA* type which has multiple initial and final terminals. This new model is a combination of Model-2 and Model-3. The set of the initial terminal is denoted  $Q_0$  and the set of the final terminal

is represented as  $Q_F = \{(q, c) | c = 0\}$ . Each set of terminals, of course, has zero counter value.

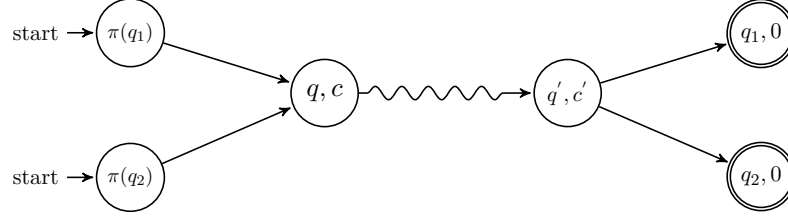


Figure 5.9: Model-4 provides multiple initial points and multiple terminate points for all designated paths that are valid throughout the length of the given observation sequence.

Two crucial edges of the model provide multiple initial and terminating points for a valid path. The branching can start in one of the initial states and it can terminate in one of the final terminals of the model at the last time step, which may vary depending on the length of the given observation.

The existing adapted version of the three fundamental algorithms could also be applied to Model-4. The important part is now both terminals of the model are multiple rather than a single terminal. We set out to investigate the definition of the adapted algorithms must be changed up to consider multiple terminals for both edges of the model. We shall again present that the characteristic feature of the terminal does not affect the purpose of the algorithms but only does over the structure of the model.

### 5.3.1 Applying adapted *forward* algorithm to Model-4

As it known that this algorithm consists two phases which are preliminary *forward* calculation and normalization *forward* flow.

#### Preliminary *forward* calculation

This phase computes the joint probability of given observation sequence  $o_{1:t}$  and hidden variable  $S_t = (q, c)$  at time  $t$  with condition on initial vector  $\pi(S_0)$

$$\hat{\alpha}_t(S_t) = Pr(o_{1:t}, S_t | \pi(S_0)) \quad (5.79)$$

It has been proved in the section 5.1.1 of Model-2. According to the definition the calculation of preliminary forward conditions only on initial vector and the proof of formula has been done in Theorem 5.1.1.

At the initial step  $t = 0$ , each initial terminal has an initial probability distribution which is held in the initial vector  $\pi$ . In the base step  $t = 1$ , the sum of the multiplication of initial, state transition, and emission probabilities are computed by considering the multiple starting terminals.

The preliminary calculation of *forward* flow in Model-4 has the same computing formula as Model-2 has where

- Base Step:

$$\hat{\alpha}_1(q, c) = \sum_{q_0 \in Q_0} \pi(q_0) A^0(q_0, \Delta, q) B(q, o_1) \quad (5.80)$$

- Recursion Step:  
if  $c - \Delta = 0$  :

$$\hat{\alpha}_t(q, c) = \sum_{q' \in Q} \sum_{\Delta=0}^1 \hat{\alpha}_{t-1}(q', c - \Delta) A^0(q', \Delta, q) B(q, o_t) \quad (5.81)$$

- if  $c - \Delta > 0$  :

$$\hat{\alpha}_t(q, c) = \sum_{q' \in Q} \sum_{\Delta=-1}^1 \hat{\alpha}_{t-1}(q', c - \Delta) A^+(q', \Delta, q) B(q, o_t) \quad (5.82)$$

### Normalisation *forward* calculation

The operating behaviour of this phase has effectively same manner up to previous models. After included the multiple terminals to both initial and final step, the following definition could be obtained for this phase

$$\alpha_t(q, c) = Pr(o_{1:t}, S_t = (q, c) | \pi(S_0), S_T \in Q_F) \quad (5.83)$$

In the previous model, Model-2, the feature of multiple initial terminals was defined. However, the new feature of the final terminal where the multiple final terminals are additionally

included in this model is defined. Therefore, we establish that a simple change at the base step which is the first step of normalization calculation.

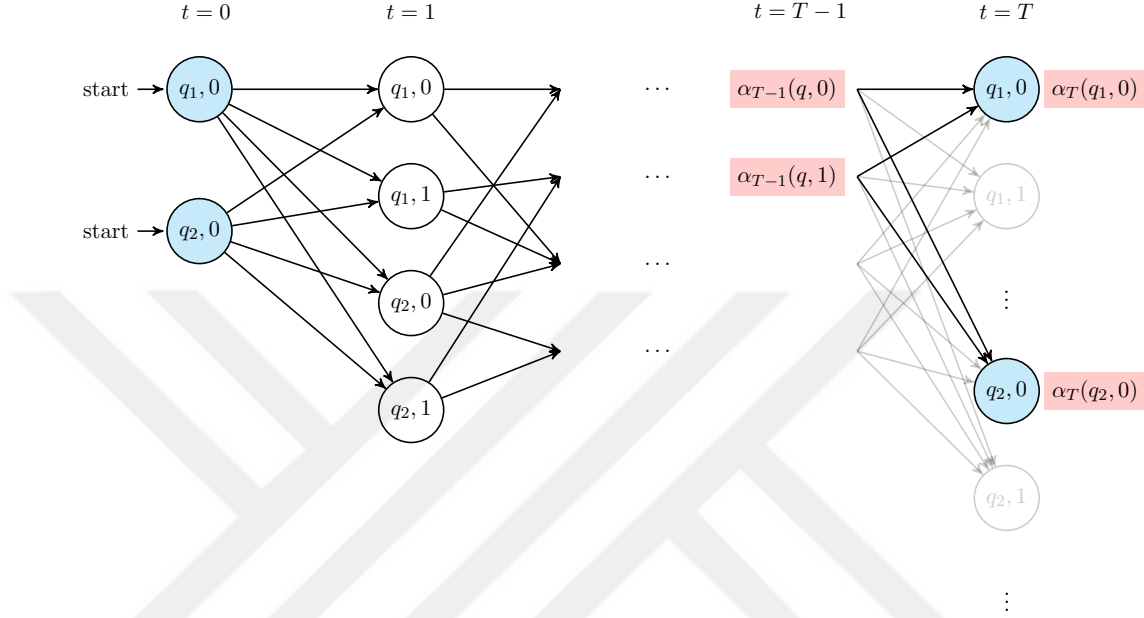


Figure 5.10: Multiple terminals at both side of initial and final that yield multiple starting and terminating points for computation of the normalization of forward flow.

This change allows us to conclude that the behavior of multiple terminals in both edges induces a summation over both initial and final terminals of all valid paths probabilities at the denominator of the formula of the base step. It can be started by assigned time as  $t = T$  to start the base step of the normalization;

$$\alpha_T(q, c) = Pr(o_{1:T}, S_T = (q, c) | \pi(S_0), S_T \in Q_F) \quad (5.84)$$

The product rule can be applied as

$$\alpha_T(q, c) = \frac{Pr(o_{1:T}, S_T = (q, c), S_T \in Q_F | \pi(S_0))}{Pr(S_T \in Q_F | \pi(S_0))} \quad (5.85)$$

If  $q \in Q_F$  and  $c = 0$  then;

$$\alpha_T(q, 0) = \frac{Pr(o_{1:T}, S_T = (q, 0), S_T \in Q_F | \pi(S_0))}{Pr(S_T \in Q_F | \pi(S_0))}$$

$$\alpha_T(q, 0) = \frac{Pr(o_{1:T}, S_T = (q, 0) \in Q_F | \pi(S_0))}{Pr(S_T \in Q_F | \pi(S_0))} \quad (5.86)$$

The numerator is the preliminary forward calculation if only if the state-counter pair is  $(q, 0) \in Q_F$

$$\alpha_T(q, 0) = \frac{\hat{\alpha}_T(q, 0)}{Pr(S_T \in Q_F | \pi(S_0))} \quad (5.87)$$

The denominator of the equation 5.87 is equal to the sum of all only paths that are existed between the multiple initial and final terminals.

$$\alpha_T(q, 0) = \frac{\hat{\alpha}_T(q, 0)}{\sum_{q_0 \in Q_0} \pi(q_0) \left( \sum_{q_F \in Q_F} Pr(S_T = (q_F, 0) | S_0 = q_0) \right)} \quad (5.88)$$

The recursion calculation of this phase might be defined again by considering the new feature of the initial and final terminals:

$$\alpha_t(q, c) = Pr(o_{1:t}, S_t = (q, c) | \pi(S_0), S_T \in Q_F) \quad (5.89)$$

If the recursion calculation of the normalization phase is referenced from same calculation of the previous models, then the equation 5.89 equals to the following equation

$$\alpha_t(S_t) = \sum_{S_{t+1}} \frac{Pr(o_{1:t+1}, S_{t+1} | \pi(S_0), S_T \in Q_F) Pr(S_{t+1} | S_t) Pr(o_{1:t}, S_t | \pi(S_0))}{Pr(o_{1:t+1}, S_{t+1} | \pi(S_0))} \quad (5.90)$$

where  $S_t$  is the current hidden variable and  $S_{t+1}$  is the next hidden variable, respectively. The proof steps are totally same as it is done in the previous models. All the processes which are the *head – to – tail* rule of *d – separation*, *Markov* property, sum-product-chain rules could be applied one to one to prove the equation 5.90, however, there is no need to do it over again here. Thus, the recursion formula of the normalization *forward* calculation is

$$\alpha_t(S_t) = \sum_{S_{t+1}} \frac{\alpha_{t+1}(S_{t+1}) A^{\delta(c)}(S_t, \Delta, S_{t+1}) \hat{\alpha}_t(S_t)}{\hat{\alpha}_{t+1}(S_{t+1})} \quad (5.91)$$

where

$$\delta(c) = \begin{cases} 0 & : c = 0 \\ + & : c > 0 \end{cases}$$

### 5.3.2 Applying adapted *backward* algorithm to Model-4

The *backward* flow starts the calculation from one of the final terminals where it could be chosen among the set of the final hidden variables. The calculation is started at the end of the time interval (i.e.  $t = T$ ) until the time step is reached to  $t = 1$  by decremented at each step. This calculation also consists of two main phases to compute the probability of the future output information that could be emitted by a particular hidden variable according to the multiple initial and final terminals of the model.

#### Preliminary *backward* calculation

The first phase calculation of adapted *backward* algorithm could be evaluated as the following equation after the new feature of both initial and final terminal is included

$$\hat{\beta}_t(q, c) = Pr(S_T \in Q_F | S_t = (q, c), \pi(S_0)) \quad (5.92)$$

In the *base* step, the preliminary *backward* probability of each particular final state which has zero counter value must be one. To prove this information could be started by assigned the last time value  $T$  to the time step  $t$  as  $t = T$  into the equation 5.92

$$\hat{\beta}_T(q, c) = Pr(S_T \in Q_F | S_T = (q, c), \pi(S_0)) \quad (5.93)$$

After applying the product rule, the equation 5.93 could be evolved as

$$\hat{\beta}_T(q, c) = \frac{Pr(S_T \in Q_F, S_T = (q, c) | \pi(S_0))}{Pr(S_T = (q, c) | \pi(S_0))} \quad (5.94)$$

If the particular hidden state  $q$  is a final state from the set of the final terminals  $Q_F$  as  $q \in Q_F$ , and the counter value of the same hidden variable  $c$  is equal to zero, then

$$\begin{aligned} \hat{\beta}_T(q, 0) &= \frac{Pr(S_T \in Q_F, S_T = (q, 0) \in Q_F | \pi(S_0))}{Pr(S_T = (q, 0) \in Q_F | \pi(S_0))} \\ &= \frac{Pr(S_T = (q, 0) \in Q_F | \pi(S_0))}{Pr(S_T = (q, 0) \in Q_F | \pi(S_0))} \\ &= 1 \end{aligned} \quad (5.95)$$

$\forall q \in Q_F$ . The recursion part of the preliminary *backward* calculation has same formula to compute the *backward* flow recursively as all previous models have.

$$\begin{aligned}\hat{\beta}_t(q, c) &= Pr(S_T \in Q_F | S_t = (q, c), \pi(S_0)) \\ &= \sum_{S_{t+1}} Pr(S_T \in Q_F | S_{t+1}, \pi(S_0)) Pr(S_{t+1} | S_t)\end{aligned}\quad (5.96)$$

where the first element of the equation 5.96 is the probability of the preliminary *backward* flow of the future variable  $S_{t+1}$ , and the second argument of the same equation is the probability of state transition which depends on the current counter value  $c$ . The recursion formula of this phase is

$$\hat{\beta}_t(S_t) = \sum_{S_{t+1}} \hat{\beta}_{t+1}(S_{t+1}) A^{\delta(c)}(S_t, \Delta, S_{t+1}) \quad (5.97)$$

where

$$\delta(c) = \begin{cases} 0 & : c = 0 \\ + & : c > 0 \end{cases}$$

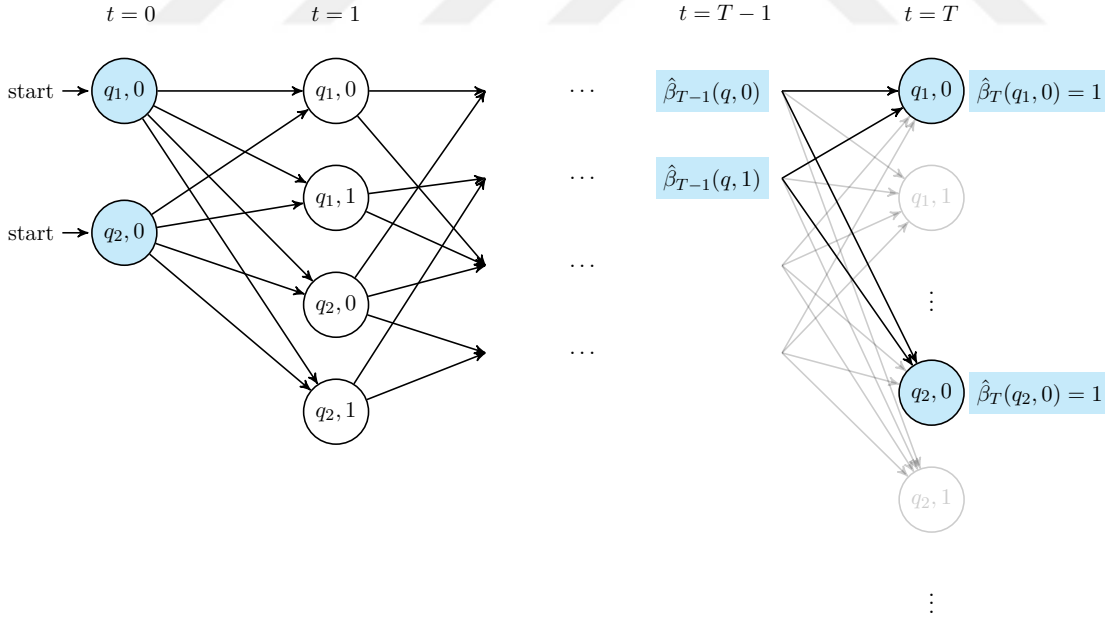


Figure 5.11: Multiple final terminals provides multiple starting points for computation of the both preliminary and normalization of *backward* flow.

The recursion part could be flowed throughout the length of the given observation sequence where it computes the preliminary values of each particular hidden variables at each time step until reaches to the time step  $t = 1$ . At the last time step,  $t = 0$ , the preliminary *backward* value of each initial states of the model could be computed as shown in the following equation.

$$\hat{\beta}_0(q_0, 0) = \pi(q_0) \sum_{S_1} \sum_{\Delta=0}^1 \hat{\beta}_1(S_1) A^0(q_0, \Delta, S_1) \quad (5.98)$$

where  $\forall q_0 \in Q_0$ .

### Normalize *backward* calculation

The normalization phase of the *backward* algorithm is processed for the Model-4 as the following equation after included the multiple both initial and final terminals

$$\beta_t(q, c) = Pr(o_{t+1:T} | S_t = (q, c), \pi(S_0), S_T \in Q_F) \quad (5.99)$$

The calculation starts at the last time step  $t = T$ . Therefore, in the *base* step, the normalization value of each individual final terminal must be one.

$$\beta_T(q, c) = Pr(o_{T+1:T} | S_T = (q, c), \pi(S_0), S_T \in Q_F) \quad (5.100)$$

The same steps which are also performed in the preliminary part of the same algorithm could be applied to prove the information that is regarded as the value of the normalization *backward* at the base step. If the product rule is applied to equation 5.100, then the following definition could be achieved

$$\beta_T(q, c) = \frac{Pr(\emptyset, S_T = (q, c), S_T \in Q_F | \pi(S_0))}{Pr(S_T = (q, c), S_T \in Q_F | \pi(S_0))} \quad (5.101)$$

If and only if  $q \in Q_F$  and  $c = 0$  then

$$\begin{aligned} \beta_T(q, c) &= \frac{Pr(S_T = (q, c) \in Q_F, S_T \in Q_F | \pi(S_0))}{Pr(S_T = (q, c) \in Q_F, S_T \in Q_F | \pi(S_0))} \\ &= 1 \end{aligned} \quad (5.102)$$

As in the recursion calculation of the preliminary phase, the recursion part of the second phase also is operated smoothly with the backward flow until the time  $t = 1$ . The same phase of the same calculation of the previous models could be referenced to achieve the recursive formula. With the same proof steps, the equation 5.100 is equal the following equation as proportionally

$$\beta_t(S_t) = \sum_{S_{t+1}} Pr(o_{t+2:T}|S_{t+1}, \pi(S_0), S_T \in Q_F) Pr(S_{t+1}|S_t) Pr(o_{t+1}|S_{t+1}) \frac{Pr(S_T \in Q_F|S_{t+1}, \pi(S_0))}{Pr(S_T \in Q_F|S_t, \pi(S_0))} \quad (5.103)$$

where  $S_t$  is the current hidden state-counter pair and  $S_{t+1}$  is the next hidden variable. Therefore, the recursion formula could be represented as

$$\beta_t(S_t) = \sum_{S_{t+1}} \beta_{t+1}(S_{t+1}) A^{\delta(c)}(S_t, \Delta, S_{t+1}) B(S_{t+1}, o_{t+1}) \frac{\hat{\beta}_{t+1}(S_{t+1})}{\hat{\beta}_t(S_t)} \quad (5.104)$$

where

$$\delta(c) = \begin{cases} 0 & : c = 0 \\ + & : c > 0 \end{cases}$$

### 5.3.3 Applying adapted *Baum-Welch* algorithm to Model-4

In *E-step* of the adapted version of *Baum-Welch*, there are two main calculations that collaborate with adapted *forward* and adapted *backward* algorithms to determine the number of being in a particular hidden variable and the number of transitions between two particular hidden variables, respectively. The first calculation provides the expectation probability of being in the particular hidden state-counter pair at the particular time  $t$ .

$$\gamma_t(q, c) = Pr(S_t = (q, c) | O, \pi(S_0), S_T \in Q_F) \quad (5.105)$$

If the same processes that are used in the previous two models(i.e. Model-2 and Model-3) are applied to obtain a formula this calculation, a most likely the same formula will be obtained as follows

$$\gamma_t(S_t) = \frac{\hat{\alpha}_t(S_t) \beta_t(S_t) \hat{\beta}_t(S_t)}{\sum_{S_T \in Q_F} \hat{\alpha}_T(S_T, 0)} \quad (5.106)$$

The same processes and the same references could be used for the calculation of the probability of the existing path between two particular hidden variables of the model.

$$\xi_t((q', c'), (q, c)) = Pr(S_{t-1} = (q', c'), S_t = (q, c) | O, \pi(S_0), S_T \in Q_F) \quad (5.107)$$

The equation 5.107 could be shaped as a simple definition to easily understand after applied the same proving process

$$\xi_t((q', c'), (q, c)) = \frac{\hat{\alpha}_{t-1}(q', c') \beta_t(q, c) \hat{\beta}_t(q, c) A^{\delta(c')}(q', \Delta, q) B(q, o_t)}{\sum_{S_T \in Q_F} \hat{\alpha}_T(S_T, 0)} \quad (5.108)$$

In *M-step*, the initial vector is additionally included to the calculation of the joint probability distribution over hidden variables and observed information as it is represented in equation 5.39. It has already been shown in Model-3 that the multiple final terminal feature has no participating on this joint calculation. As it is mentioned in section 5.1.3 of Model-2, the main parameters of the model which are  $A^0$ ,  $A^+$ , and  $B$  have same updating formula as Model-1 has. The additional parameter of this model  $\pi$  has also same calculation as the equation 5.50 due to it is surrounded by only  $\gamma$  value at the particular time  $t = 0$ .

## 5.4 Model-5: One initial terminal with zero counter and one particular final state with non-zero counter

The characteristic feature of the final terminal substantially is formed with zero counter value in each previous *HP1CA* models. However, we a little reinforce the conventional accepting rule, which is that the counter value of the accepting state is zero, of the counter machines in this new model. We shall introduce a suitable notation of the final terminal for this model by considering non-zero counter values. The final terminal of the model is denoted as

$$Q_F = \{(q_F, c) | c \geq 0\} \quad (5.109)$$

where  $q_F$  is the hidden state that is chosen from the set of control states of the model, and the counter value could be a non-zero natural number. The model has one initial state with zero counter value that is denoted as  $q_0$  as same as it is represented in Model-1.

The processing of the model starts from the initial terminal  $S_0 = (q_0, 0)$  at  $t = 0$ . The model must be terminated at the final state (i.e.  $S_T \in Q_F$ ) and the counter of the state might be a non-zero value.

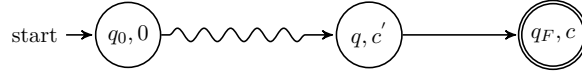


Figure 5.12: The counter value of the final terminal  $q_F$  might be non-zero instead zero value as in the previous models.

The inference of the counter value of this model could easily go to infinity number without any determined threshold. A boundary must be included to prevent occurring this unexpected condition while processing the model.

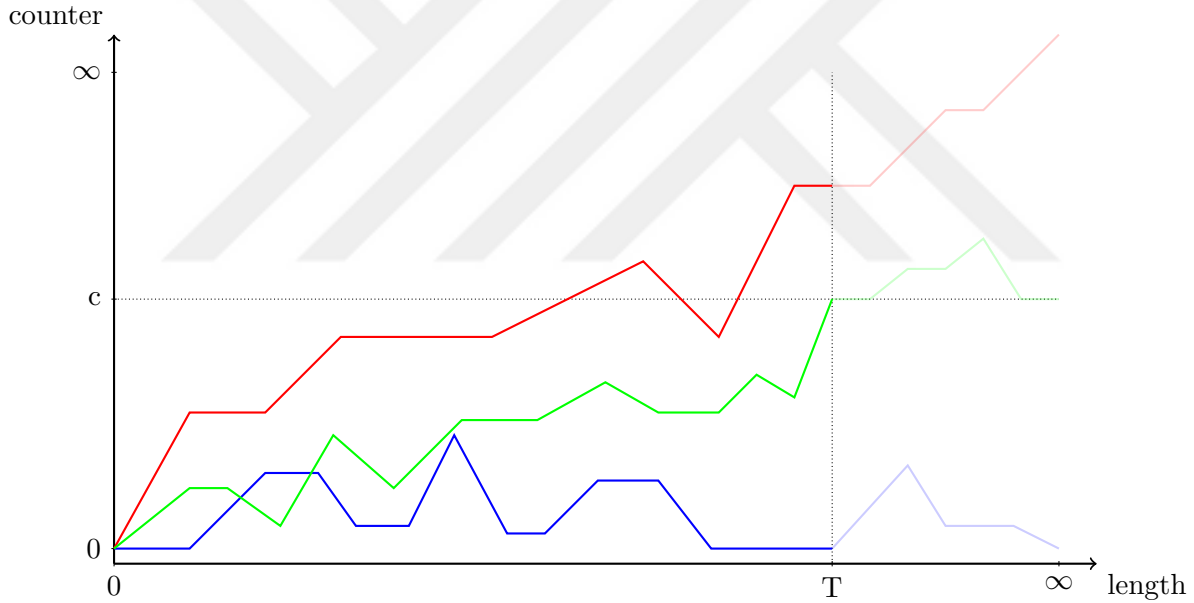


Figure 5.13: Three different type of Model-5 which pretending to generate the  $T$  length output sequence

This threshold is the length of the given observation sequence. The branching is terminated in the final terminal with a non-zero counter value when the last output symbol of the given output sequence is emitted. For instance, let  $T$  represents the length of the given observation sequence as shown in Figure 5.13. There is a worst-case scenario regarding this feature of the counter value in this model. It is that the model terminates in a different

hidden variable but the counter value is non-zero at time step  $t = T$ . It is shown by the red line in Figure 5.13. If this scenario occurs, then the probability of the output sequence that is processed is equal to zero.

We consider the adapted version of the fundamental algorithms of *HMM* to apply in the correspondence of this particular model to estimate the parameters. We establish that every executable algorithm of Model-1 can be applied to this model.

#### 5.4.1 Applying adapted *forward* algorithm to Model-5

The calculation still consists two sub-calculations that are defined preliminary *forward* calculation and normalization *forward* flow, respectively.

##### Preliminary *forward* calculation

The definition of this phase is same as it is defined in equation 4.8 in Model-1.

$$\hat{\alpha}_t(S_t) = Pr(o_{1:t}, S_t | S_0) \quad (5.110)$$

The final terminal is irrelevant regarding this calculation part of the *forward* flow. On the other hand, only one initial terminal in this model, therefore the proof of the preliminary *forward* calculation correspond with the proof of the same calculation in Model-1. The formulas of the base and recursion steps of this calculation are also the same as in Model-1.

##### Normalisation *forward* calculation

The second phase of the adapted *forward* calculates:

$$\alpha_t(q, c) = Pr(o_{1:t}, S_t = (q, c) | S_0, S_T \in Q_F) \quad (5.111)$$

The final terminal is unique however the counter value shows an alteration depends on the length of the given output sequence. In this case, the model must be terminated in the final terminal, however, the counter value of the final state might be a non-zero natural number that is been between  $0 \leq c \leq T$  where  $T$  represents the length of the output sequence.

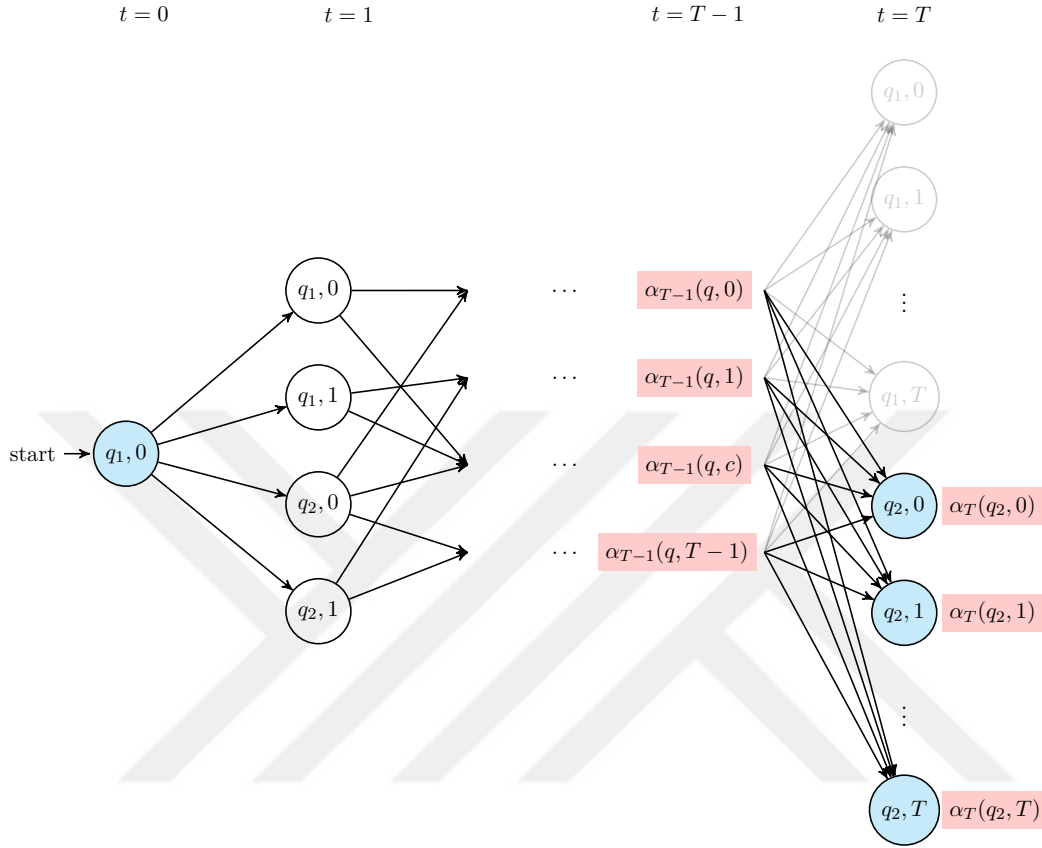


Figure 5.14: Multiple counter values that yields multiple final points where each one of them possibly being a final terminal to the model for computation of the normalization of *forward* calculation.

Since the counter value in the last terminal varies depending on the length of the processed observation sequence, the normalization calculation of the final terminal is performed according to each designated counter value. The computation of the second phase starts the calculation at  $t = T$ .

$$\alpha_T(q, c) = \Pr(o_{1:T}, S_T = (q, c) | S_0, S_T \in Q_F) \quad (5.112)$$

After the product rule is applied to equation 5.112;

$$\alpha_T(q, c) = \frac{\Pr(o_{1:T}, S_T = (q, c), S_T \in Q_F | S_0)}{\Pr(S_T \in Q_F | S_0)} \quad (5.113)$$

If the hidden variable pair  $(q, c) \in Q_F$  where  $q = q_F$  and the counter value  $0 \leq c \leq T$ , then

$$\begin{aligned}\alpha_T(q_F, c) &= \frac{Pr(o_{1:T}, S_T = (q_F, c), S_T \in Q_F | S_0)}{Pr(S_T \in Q_F | S_0)} \\ &= \frac{Pr(o_{1:T}, S_T = (q_F, c) | S_0)}{Pr(S_T \in Q_F | S_0)} \\ &= \frac{\hat{\alpha}_T(q_F, c)}{Pr(S_T \in Q_F | S_0)}\end{aligned}\tag{5.114}$$

The denominator of the equation 5.114 is the sum of the probability of each path that exists between the initial terminal and the final terminal that has multiple counter values.

$$\alpha_T(q_F, c) = \frac{\hat{\alpha}_T(q_F, c)}{\sum_{c'=0}^T Pr(S_T = (q_F, c') | S_0)}\tag{5.115}$$

In the recursion part of this calculation uses the same recursive formula as the previous models use. The only difference is the multiple counter values of the final terminal provide various points to start normalizing the *forward* flow:

$$\begin{aligned}\alpha_t(S_t) &= \sum_{S_{t+1}} \frac{Pr(o_{1:t+1}, S_{t+1} | S_0, S_T \in Q_F) Pr(S_{t+1} | S_t) Pr(o_{1:t}, S_t | S_0)}{Pr(o_{1:t+1}, S_{t+1} | S_0)} \\ &= \sum_{S_{t+1}} \frac{\alpha_{t+1}(S_{t+1}) A^{\delta(c)}(S_t, \Delta, S_{t+1}) \hat{\alpha}_t(S_t)}{\hat{\alpha}_{t+1}(S_{t+1})}\end{aligned}\tag{5.116}$$

where

$$\delta(c) = \begin{cases} 0 & : c = 0 \\ + & : c > 0 \end{cases}$$

#### 5.4.2 Applying adapted *backward* algorithm to Model-5

The other adapted dynamic programming algorithm *backward* algorithm could be applied to Model-5 by consisting two main computation phases which are called preliminary *backward* calculation and normalization of *backward* flow.

### Preliminary *backward* calculation

The first phase of the adapted version of *backward* algorithm computes;

$$\hat{\beta}_t(q, c) = Pr(S_T \in Q_F | S_t = (q, c), S_0) \quad (5.117)$$

At base step  $t = T$ , the preliminary *backward* value of the final terminal corresponding the each designated counter value is one;

$$\hat{\beta}_T(q_F, c) = 1 \quad (5.118)$$

$\forall c$  where  $0 \leq c \leq T$ . In recursion part, the first phase computes:

$$\begin{aligned} \hat{\beta}_t(q, c) &= Pr(S_T \in Q_F | S_t = (q, c), S_0) \\ &= \sum_{S_{t+1}} \hat{\beta}_{t+1}(S_{t+1}) A^{\delta(c)}(S_t, \Delta, S_{t+1}) \end{aligned} \quad (5.119)$$

where

$$\delta(c) = \begin{cases} 0 & : c = 0 \\ + & : c > 0 \end{cases}$$

### Normalization of *backward* calculation

The second phase of adapted *backward* algorithm consists two calculation steps to normalize the *backward* flow. The second phase computes;

$$\beta_t(q, c) = Pr(o_{t+1:T} | S_t = (q, c), S_0, S_T \in Q_F) \quad (5.120)$$

In the base step, only final terminal with all possible counter values has the normalization value at the last time step  $t = T$ . This value is equal to one if and only if the hidden variable  $q$  is the final terminal  $q_F$  of the model, and the counter value might be non-zero.

$$\beta_T(q_F, c) = 1 \quad (5.121)$$

$\forall c$  where  $0 \leq c \leq T$ .

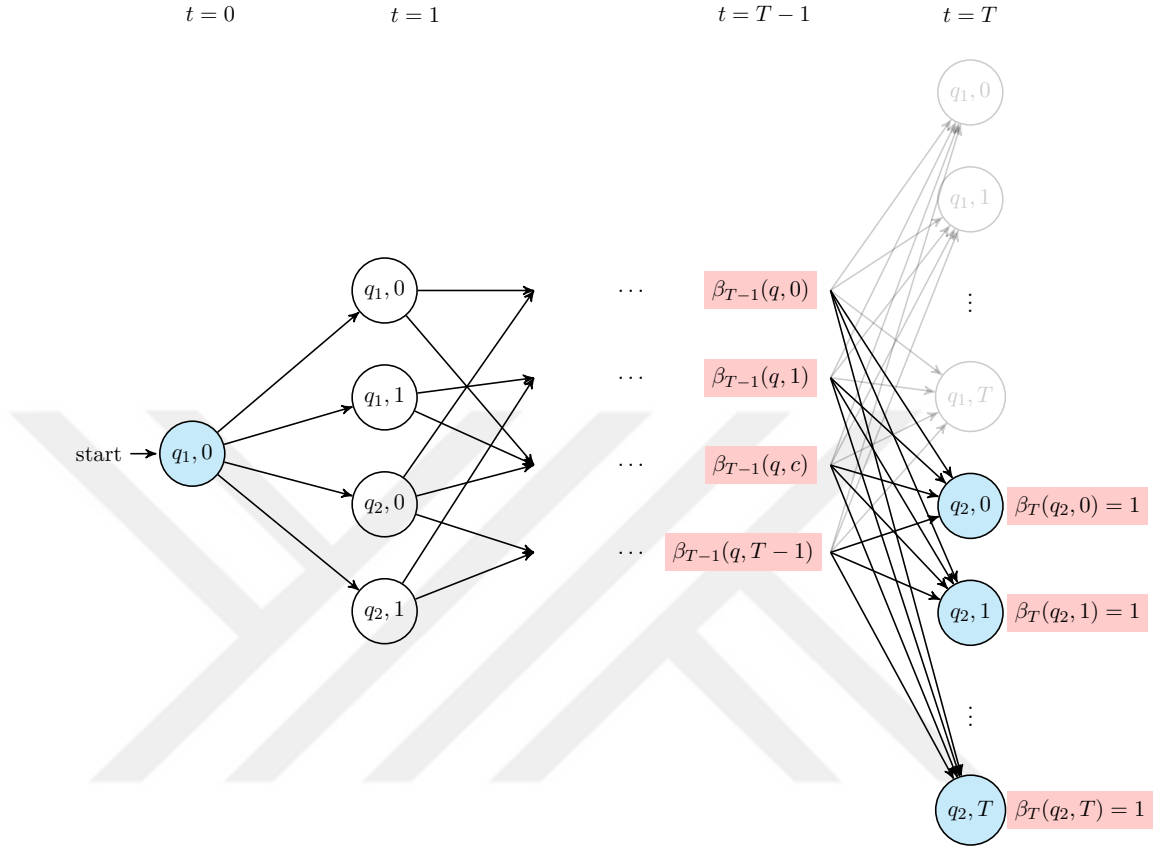


Figure 5.15: The value of the normalization of *backward* flow is one for if the particular hidden variable is equal to final terminal with non-zero counter value.

The recursion part of this calculation the following formula is used as;

$$\beta_t(q, c) = \sum_{S_{t+1}} Pr(o_{t+2:T}|S_{t+1}, S_0, S_T \in Q_F)) Pr(S_{t+1}|S_t) Pr(o_{t+1}|S_{t+1}) \frac{Pr(S_T \in Q_F|S_{t+1}, S_0)}{Pr(S_T \in Q_F|S_t, S_0)} \quad (5.122)$$

which is equal to;

$$\beta_t(S_t) = \sum_{S_{t+1}} \beta_{t+1}(S_{t+1}) A^{\delta(c)}(S_t, \Delta, S_{t+1}) B(S_{t+1}, o_{t+1}) \frac{\hat{\beta}_{t+1}(S_{t+1})}{\hat{\beta}_t(S_t)} \quad (5.123)$$

where

$$\delta(c) = \begin{cases} 0 & : c = 0 \\ + & : c > 0 \end{cases}$$

The same step is followed as they are done in previous models for the same calculation part of the second phase of adapted *backward* flow.

### 5.4.3 Applying adapted *Baum-Welch* algorithm to Model-5

According to the definition of the adapted version of the *Baum-Welch* algorithm in Model-1 it is used to estimate the parameters of the model. The algorithm could be applied to update the parameters of Model-5 by processing on two steps which are *E-step* and *M-step*.

In *E-step*,  $\gamma$  and  $\xi$  calculations are obtained by using the same proof steps that are expressed in previous models (e.g. Model-1). However, the multiple counter property obliges a small changing to the denominator of both calculations. As it is defined in previous models,  $\gamma$  calculation is;

$$\gamma_t(q, c) = Pr(S_t = (q, c) | O, S_0, S_T \in Q_F) \quad (5.124)$$

Assuming same proving steps are applied to achieve the formula of  $\gamma$  calculation, then the following equation would be obtained;

$$\gamma_t(S_t) = \frac{\hat{\alpha}_t(S_t)\beta_t(S_t)\hat{\beta}_t(S_t)}{Pr(O, S_T \in Q_F | S_0)} \quad (5.125)$$

If the observation sequence  $O$  might be expanded as  $o_{1:T}$  at the denominator of the equation 5.125, then the denominator of the same equation becomes equivalently equal to the corresponding definition of the preliminary *forward* value of the final terminal with the non-zero counter value. Thus, the sum of the preliminary value of the final state over feasible counter values must be included to the expressing of the formula as

$$\gamma_t(S_t) = \frac{\hat{\alpha}_t(S_t)\beta_t(S_t)\hat{\beta}_t(S_t)}{\sum_{c'=0}^T \hat{\alpha}_T(q_F, c')} \quad (5.126)$$

The other calculation of *E-step* is  $\xi$  calculation that defines as

$$\xi_t(S_{t-1}, S_t) = Pr(S_{t-1}, S_t | O, S_0, S_T \in Q_F) \quad (5.127)$$

where  $S_{t-1}$  is the current hidden state-counter pair and  $S_t$  is the next hidden variable. The formula of  $\xi$  calculation could also be obtained if the same proving steps are followed

where it is already done in Model-1. Assuming the same proof steps are processed to this model, then the following equation would be achieved;

$$\xi_t((q', c'), (q, c)) = \frac{\hat{\alpha}_{t-1}(q', c')\beta_t(q, c)\hat{\beta}_t(q, c)A^{\delta(c')}(q', \Delta, q)B(q, o_t)}{Pr(O, S_T \in Q_F | S_0)} \quad (5.128)$$

where

$$\delta(c') = \begin{cases} 0 & : c' = 0 \\ + & : c' > 0 \end{cases}$$

The denominator of the equation 5.128 needs a small change considering multiple feasible counter values likewise it is done in  $\gamma$  calculation by expanding the given observation sequence  $O$  like  $o_{1:T}$ . According to this aspect, the denominator of the equation would be equal to the sum of the preliminary value of the final terminal over feasible counter values. Then the equation

$$\xi_t((q', c'), (q, c)) = \frac{\hat{\alpha}_{t-1}(q', c')\beta_t(q, c)\hat{\beta}_t(q, c)A^{\delta(c')}(q', \Delta, q)B(q, o_t)}{\sum_{c'=0}^T \hat{\alpha}_T(q_F, c')} \quad (5.129)$$

In  $M$ -step, the calculation of the joint probability distribution is defined as represented in equation 5.78. It could be expressed that consisting of multiple counter value on the final terminal has no effect on this joint probability calculation. Thus, the formulas of the estimating parameters have the same form as they are described in Model-1. Because the updating formulas fully consist of  $\gamma$  and  $\xi$  calculations which are already obtained for Model-5.

## 5.5 Model-6: Initial vector and one particular final state with non-zero counter

The non-zero counter value feature of the final terminal is defined in previous section 5.4. The same feature will also be in this new model, but this time, multiple initial terminals with zero counter value will be used instead of the only one initial terminal with zero counter value. The probability of the initial state distribution is kept in a vector that is denoted as

$$\pi(q_0) = Pr(S_0 = q_0) \quad (5.130)$$

where  $q_0$  is one of the initial terminals that belongs to the set of  $Q_0$ . The final state is unique but it could have non-zero counter value where

$$Q_F = \{(q_F, c) | c \geq 0\} \quad (5.131)$$

The model starts the processing by picking the initial distribution probability corresponding to the initial terminals from the initial vector  $\pi$ . The processing continues, depending on the length of the given observation, through the paths that exist between one of the initial terminals and final terminal with a feasible non-zero counter value.

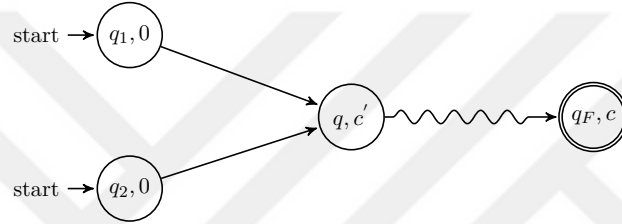


Figure 5.16: There are multiple initial terminals and one unique final terminal with non-zero counter value.

The features of both initial and final terminals oblige some changing must be done for the adapted three fundamental algorithms for this model. However, the features of both the terminal side of the model are been already defined in corresponding Model-2 and Model-5, respectively. Thus, the feature of both models could be merged in this final model Model-6.

### 5.5.1 Applying adapted *forward* algorithm to Model-6

The first fundamental algorithm is adapted *forward* algorithm where it consists of two main phases to calculate the probability of the given observation sequence that is emitted by the hidden variable of the model with considering all possible counter values.

#### Preliminary *forward* calculation

In this phase, the joint probability of the partial observation sequence  $o_{1:t}$  and the particular hidden variable  $S_t$  by condition on the all possible initial terminals  $\pi(S_0)$  is computed as

$$\hat{\alpha}_t(q, c) = Pr(o_{1:t}, S_t = (q, c) | \pi(S_0)) \quad (5.132)$$

The preliminary *forward* calculation has same definition as the equation 5.2 where it is described in Model-2. Thus, this calculation has the same formula and the proof steps that are applied for achieving the corresponding to formulas of the preliminary calculation.

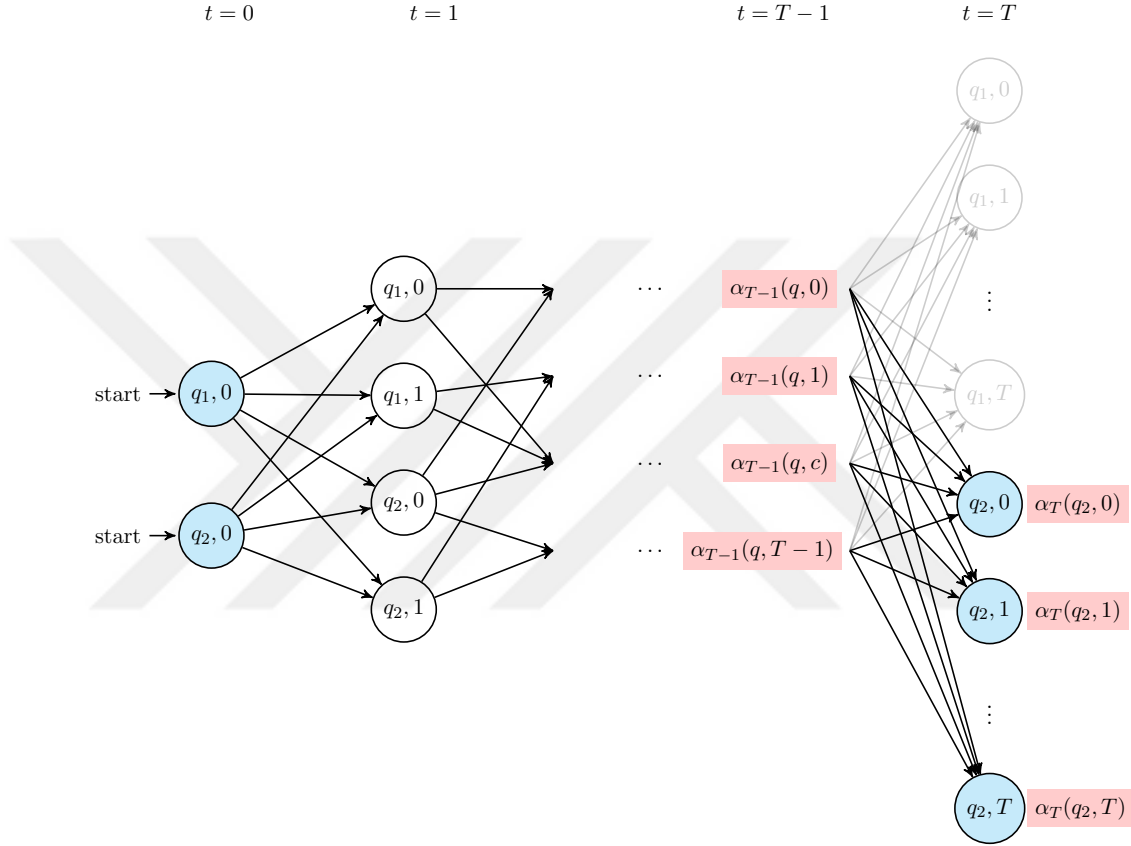


Figure 5.17: Multiple initial terminal and multiple counter values that yield multiple starting and terminated points to the model for computation of the normalization of *forward* calculation.

### Normalisation *forward* calculation

The final terminal is considered as another knowledge for the model that proposes a point where it is terminated.

$$\alpha_t(q, c) = Pr(o_{1:t}, S_t = (q, c) | \pi(S_0), S_T \in Q_F) \quad (5.133)$$

At the base step, the calculation starts, depending on the length of the observation sequence, at the last time step  $t = T$ :

$$\alpha_T(q, c) = Pr(o_{1:T}, S_T = (q, c) | \pi(S_0), S_T \in Q_F) \quad (5.134)$$

If the product rule is applied to equation 5.134, then

$$\alpha_T(q, c) = \frac{Pr(o_{1:T}, S_T = (q, c), S_T \in Q_F | \pi(S_0))}{Pr(S_T \in Q_F | \pi(S_0))} \quad (5.135)$$

If the hidden variable pair  $(q, c) \in Q_F$  where  $q = q_F$  and the counter value  $0 \leq c \leq T$ , then

$$\begin{aligned} \alpha_T(q_F, c) &= \frac{Pr(o_{1:T}, S_T = (q_F, c), S_T \in Q_F | \pi(S_0))}{Pr(S_T \in Q_F | \pi(S_0))} \\ &= \frac{Pr(o_{1:T}, S_T = (q_F, c) | \pi(S_0))}{Pr(S_T \in Q_F | \pi(S_0))} \end{aligned} \quad (5.136)$$

The numerator of the second line of the equation 5.136 is equal to preliminary *forward* flow over each individual counter value of the unique final terminal of the model. The denominator of the same equation is equal to the sum of the probability of all existing paths that are valid between the starting and the final states over both initial terminals and feasible counter values.

$$\alpha_T(q_F, c) = \frac{\hat{\alpha}_T(q_F, c)}{\sum_{q_0 \in Q_0} \pi(q_0) \left( \sum_{c'=0}^T Pr(S_T = (q_F, c') | S_0 = q_0) \right)} \quad (5.137)$$

The recursion calculation has strongly same formulas as Model-5 has:

$$\alpha_t(S_t) = \sum_{S_{t+1}} \frac{Pr(o_{1:t+1}, S_{t+1} | \pi(S_0), S_T \in Q_F) Pr(S_{t+1} | S_t) Pr(o_{1:t}, S_t | \pi(S_0))}{Pr(o_{1:t+1}, S_{t+1} | \pi(S_0))} \quad (5.138)$$

This equation is equal to:

$$\alpha_t(q, c) = \sum_{S_{t+1}} \frac{\alpha_{t+1}(S_{t+1}) A^{\delta(c)}(S_t, \Delta, S_{t+1}) \hat{\alpha}_t(S_t)}{\hat{\alpha}_{t+1}(S_{t+1})} \quad (5.139)$$

where

$$\delta(c) = \begin{cases} 0 & : c = 0 \\ + & : c > 0 \end{cases}$$

### 5.5.2 Applying adapted *backward* algorithm to Model-6

As it is happened in the previous models, the adapted version of *backward* algorithm could be successfully applied to Model-6 by computing into two phase calculations.

#### Preliminary *backward* calculation

This phase computes

$$\hat{\beta}_t(q, c) = Pr(S_T \in Q_F | S_t = (q, c), \pi(S_0)) \quad (5.140)$$

At base step  $t = T$ , the probability of the final terminal corresponding the feasible counter value is one. This knowledge is also defined in equation 5.118 in Model-5

$$\hat{\beta}_T(q_F, c) = 1 \quad (5.141)$$

$\forall c$  where  $0 \leq c \leq T$ . In recursion part, the first phase computes totally has same recursive formula:

$$\begin{aligned} \hat{\beta}_t(q, c) &= Pr(S_T \in Q_F | S_t = (q, c), \pi(S_0)) \\ &= \sum_{S_{t+1}} \hat{\beta}_{t+1}(S_{t+1}) A^{\delta(c)}(S_t, \Delta, S_{t+1}) \end{aligned} \quad (5.142)$$

where

$$\delta(c) = \begin{cases} 0 & : c = 0 \\ + & : c > 0 \end{cases}$$

At the last step,  $t = 0$ , the preliminary *backward* value of each initial states of the model could be computed as shown in the following equation.

$$\hat{\beta}_0(q_0, 0) = \pi(q_0) \sum_{S_1} \sum_{\Delta=0}^1 \hat{\beta}_1(S_1) A^0(q_0, \Delta, S_1) \quad (5.143)$$

where  $\forall q_0 \in Q_0$ .

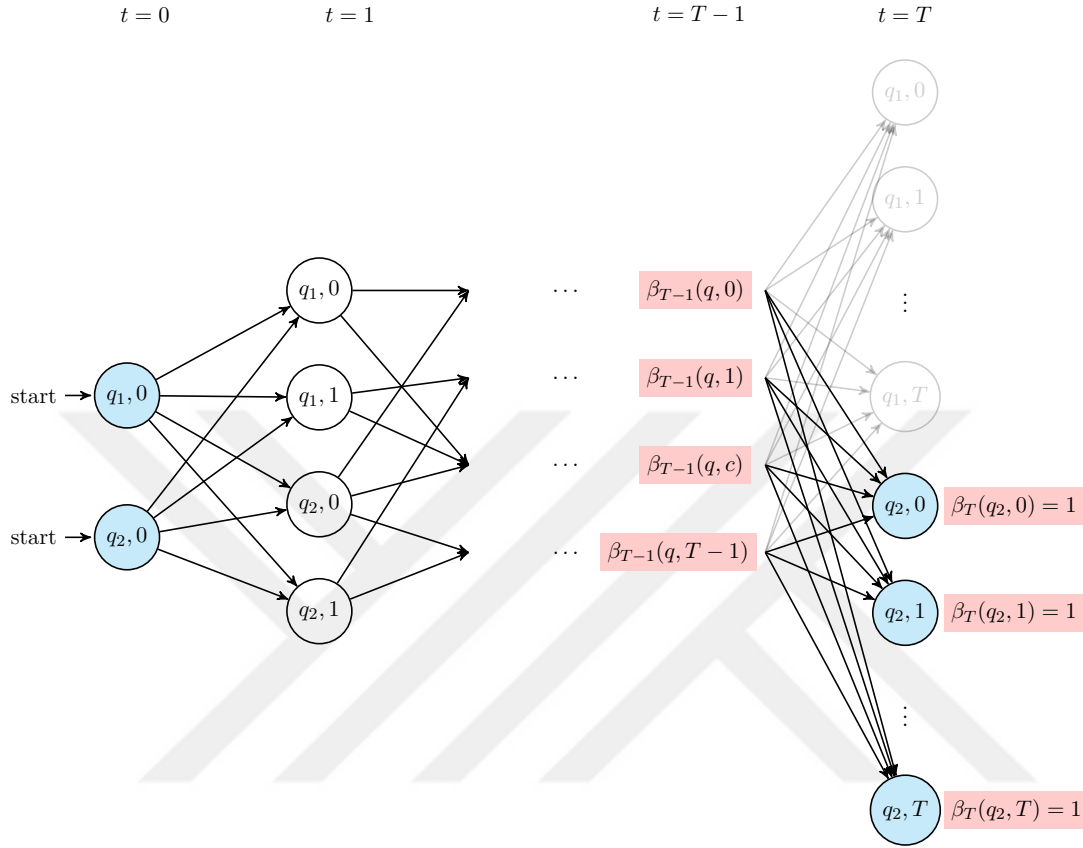


Figure 5.18: Applying initial vector where the probability distribution of multiple initial terminals are hold in, to Model-6.

### Normalize Backward

The second phase of adapted *backward* algorithm computes;

$$\beta_t(q, c) = Pr(o_{t+1:T} | S_t = (q, c), \pi(S_0), S_T \in Q_F) \quad (5.144)$$

The base step of this calculation uses the same knowledge, that is used in previous models, in which only the final terminal with all possible counter values has the normalization value at the last time step  $t = T$  where it is shown in Figure 5.18.

$$\beta_T(q_F, c) = 1 \quad (5.145)$$

$\forall c$  where  $0 \leq c \leq T$ . The recursion part of this calculation has the same proof steps that could be followed as they are done in far previous model Model-1 for the same calculation part of the second phase of adapted *backward* flow. The following formula is used to normalize the *backward* flow;

$$\beta_t(q, c) = \sum_{S_{t+1}} Pr(o_{t+2:T}|S_{t+1}, \pi(S_0), S_T \in Q_F) Pr(S_{t+1}|S_t) Pr(o_{t+1}|S_{t+1}) \frac{Pr(S_T \in Q_F|S_{t+1}, \pi(S_0))}{Pr(S_T \in Q_F|S_t, \pi(S_0))} \quad (5.146)$$

which is equal to;

$$\beta_t(S_t) = \sum_{S_{t+1}} \beta_{t+1}(S_{t+1}) A^{\delta(c)}(S_t, \Delta, S_{t+1}) B(S_{t+1}, o_{t+1}) \frac{\hat{\beta}_{t+1}(S_{t+1})}{\hat{\beta}_t(S_t)} \quad (5.147)$$

where

$$\delta(c) = \begin{cases} 0 & : c = 0 \\ + & : c > 0 \end{cases}$$

### 5.5.3 Applying adapted *Baum-Welch* algorithm to Model-5

The parameter estimation algorithm could be applied to Model-6 by referencing it is defined in Model-1, however, multiple initial terminals and non-zero counter of the final terminal are considered. The algorithm processed on two steps which are *E-step* and *M-step*.

In *E-step*, the multiple initial states and non-zero counter property oblige a small change to the denominator of both  $\gamma$  and  $\xi$  calculations. As it is defined in previous models,  $\gamma$  calculation is;

$$\gamma_t(q, c) = Pr(S_t = (q, c)|O, \pi(S_0), S_T \in Q_F) \quad (5.148)$$

Assuming same proving steps are applied to achieve the formula of  $\gamma$  calculation, then the following equation would be obtained;

$$\gamma_t(S_t) = \frac{\hat{\alpha}_t(S_t) \beta_t(S_t) \hat{\beta}_t(S_t)}{Pr(O, S_T \in Q_F|\pi(S_0))} \quad (5.149)$$

The given observation sequence  $O$  might be expanded as  $o_{1:T}$  due to make the denominator part of the equation 5.149 equal to a corresponding definition of known calculation that is

preliminary *forward* calculation, of course, it considers of the final terminal of the model with whole designated counter values. Thus, the both sum over the initial terminals and feasible counter values must be included to the expressing of the formula as

$$\gamma_t(S_t) = \frac{\hat{\alpha}_t(S_t)\beta_t(S_t)\hat{\beta}_t(S_t)}{\sum_{c'=0}^T \hat{\alpha}_T(q_F, c')} \quad (5.150)$$

The other calculation of *E-step* is  $\xi$  calculation that it could be defined as

$$\xi_t(S_{t-1}, S_t) = Pr(S_{t-1}, S_t | O, S_0, S_T \in Q_F) \quad (5.151)$$

where  $S_{t-1}$  and  $S_t$  represent the current and the next hidden state-counter pairs, respectively. If the same proof steps that are already processed in Model-1 are followed, then the following formula is obtained for  $\xi$  calculation

$$\xi_t((q', c'), (q, c)) = \frac{\hat{\alpha}_{t-1}(q', c')\beta_t(q, c)\hat{\beta}_t(q, c)A^{\delta(c')}(q', \Delta, q)B(q, o_t)}{Pr(O, S_T \in Q_F | S_0)} \quad (5.152)$$

where

$$\delta(c') = \begin{cases} 0 & : c' = 0 \\ + & : c' > 0 \end{cases}$$

Again the denominator of the equation 5.152 needs a small change considering multiple feasible counter values likewise it is done in  $\gamma$  calculation by expanding the given observation sequence  $O$ .

$$\xi_t((q', c'), (q, c)) = \frac{\hat{\alpha}_{t-1}(q', c')\beta_t(q, c)\hat{\beta}_t(q, c)A^{\delta(c')}(q', \Delta, q)B(q, o_t)}{\sum_{c'=0}^T \hat{\alpha}_T(q_F, c')} \quad (5.153)$$

In *M-step*, the calculation of the joint probability distribution is defined as represented in equation 5.38. The multiple counter value on the final terminal has no effect on this joint probability calculation, however, the multiple initial terminal feature has. In Model-2, the equation 5.39 presents that initial vector is included to the joint probability calculation. Thus, the formulas of the estimating parameters have the same form as they are described in Model-2. Because the updating formulas fully consist of  $\gamma$  and  $\xi$  calculations which are already obtained for Model-6.

## 5.6 Conclusion

In this chapter, various possible HP1CA models were investigated by allowing for different definition of initial and final terminal states. The base of all models is Model-1 that was presented in Chapter 4. Five new models are defined apart from the first model. The adapted version of the fundamental algorithms of the *HMM* is successfully applied to each of the new models like it was as applied to Model-1. The feature that makes these models different from each other is that their initial and final terminals vary.

In Model-1, there are single starting and final terminal where each of them has a zero counter value. However, if the specification of more than one initial terminal is evaluated, then a new *HP1CA* model that is called Model-2 is generated. Therefore, this new model (i.e. Model-2) contains multiple initial terminals which are chosen so that the set of initial states  $Q_0$  and a single termination terminal, of course, all terminals have zero counter value. Each initial terminal has initial distribution probability at  $t = 0$ . These probability values are stored in a vector denoted by  $\pi$  and called the initial vector as used in *HMM*. Thus a new parameter (i.e.  $\pi$ ) was included among the other parameters (i.e.  $B, A^0, A^+$ ) for this model. This new parameter must be estimated as the other are done. This assessment is discussed in sub-section 5.1.3.

Another model (i.e. it is referred to as Model-3) contains the exact opposite of the feature of the Model-2 it means that it has single initial state and multiple final states, each of has a zero counter value.  $Q_F$  represents the set of the final terminals of the model. The model could be terminated while the model reaches to the one of the designated final terminal.

Model-4 provides multiple initial and final points to both sides of the model. This new model is described as the merging of Model-2 and Model-3 where they both have multiple points in their initial and final stage, respectively.  $Q_0$  represents the set of initial terminal and  $Q_F$  is the set of the final terminal. Each set of terminals of course contains zero counter value.

In each previous various models, the models are terminated in the final terminal but the counter compulsory is zero. Model-5 provides gaining a new perspective by removing the obligation of counter value to be zero at the final state. With respect to this knowledge,

the model has a unique single initial terminal and one single final terminal where it might have non-zero counter value. The final terminal which could have multiple counter value is represented as a set where  $Q_F = \{(q_F, c) | c \geq 0\}$ .

The last model is denoted as Model-6 where it has multiple initial terminal and one final terminal which might have non-zero counter value. This model is close, but not certainly, the instruction of a *HMM*. The initial terminal provide multiple starting point as *HMM* does and the final terminal with non-zero counter value also provides multiple final points depending on the number of the feasible counter at the end time scale  $t = T$ .



## Chapter 6

# Experiments

In this section, we explain the adaptation results of all *HP1CA* models and *HMM*. The adaptations are examined in two parts. The first part is only comparing Model-1 of *HP1CA* with *HMM*, and in the second part, the remaining five models are compared with *HMM*. The datasets used in both parts are produced by the models used in the comparison. Each individual model first generates dataset and original parameters, then each one of them examines the corresponding dataset to compare their performance against *HMM*. The biggest reason for this is that existing *HP1CA* models have different initial and final terminals. Our comparison of *HP1CA* models with *HMM* takes place in three different categories. The first category is comparing by the time spent by the calculation models while learning the parameters. Then, we establish the accuracy rate, which is calculated by using the adapted Viterbi algorithm, of the given sequence. The last category of the comparison is related to finding the closest possible model to the probability of the observation sequence that is defined using the original parameters. To make the last comparison, KL divergence is used.

### 6.1 Implementation and results of Model-1

In this section, the implementation steps of Model-1 *HP1CA* is presented, and the results of the same model are compared with the result of a *HMM* by using same dataset, which is generated by Model-1. The dataset contains 1000 observation sequences of different lengths. As emphasised in section 4.2.3, the updating formulas of the parameters consider multiple observation sequences. Thus, the dimension of the output sequence acts as a

substantial part through the estimation of the parameters of the model.

The other substantial parameter is the number of iteration in the learning stage, because the learning algorithm, adapted *Baum-Welch*, is an iterative based dynamic programming algorithm that provides better probability distribution by updating the parameters of the model to local maximum (i.e.  $Pr(\theta^{new}) > Pr(\theta^{old})$ ). The figure 6.1 presents the number of iterations to obtain a local maximum of parameters for this model.

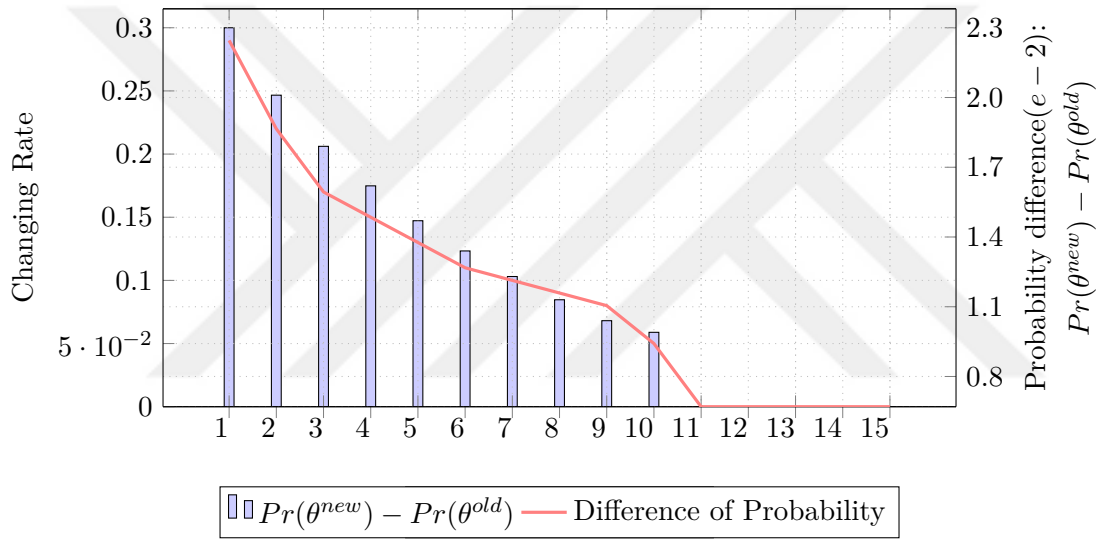


Figure 6.1: The number of iterations for Model-1

The other elements that are important to the running time are the number of hidden variables and the length of the observation sequence. As it is mentioned in Chapter 4, the learning algorithm uses the adapted *forward* and adapted *backward* algorithms in both *E-step* and *M-step*. The learning algorithms has nested loops over the length of the observation sequence  $T$ , feasible maximum counter values  $M$ , and goes twice over the number of hidden variables  $N$  in the iteration step. Therefore, the complexity of the adapted *Baum-Welch* algorithm is  $\mathcal{O}(N^2TM)$ , where the counter value could not exceed the length of the observation sequence(i.e.  $M \leq T$ ).

The time complexity of the original *Baum-Welch* is also quadratic in the number of hidden

states. It is  $\mathcal{O}(N^2T)$ . The lack of the counter size parameter provides the linear computation in the length of the observation sequence. Figure 6.2 shows the learning time scale of the four different models with various number of hidden variables and that process five different datasets that have lengths that range between 10 and 200.

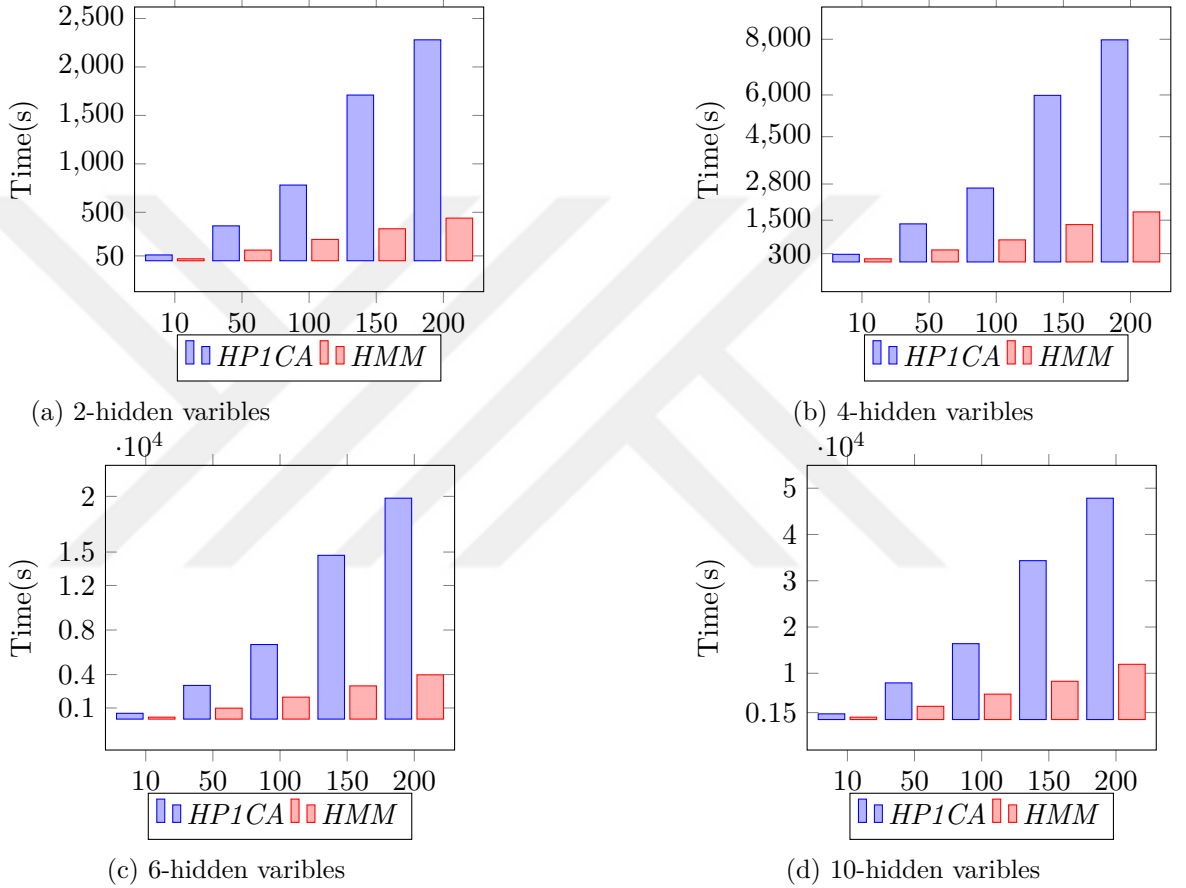


Figure 6.2: The time scales of both models of *HP1CA* and *HMM* over the same datasets

*HMM* overwhelmingly learns quickly against than *HP1CA*. The main reason for this result is that the counter value factor plays an important role, which causes an extra loop. The maximum counter value might be equal to the length of the output sequence (although it is not necessary that large).

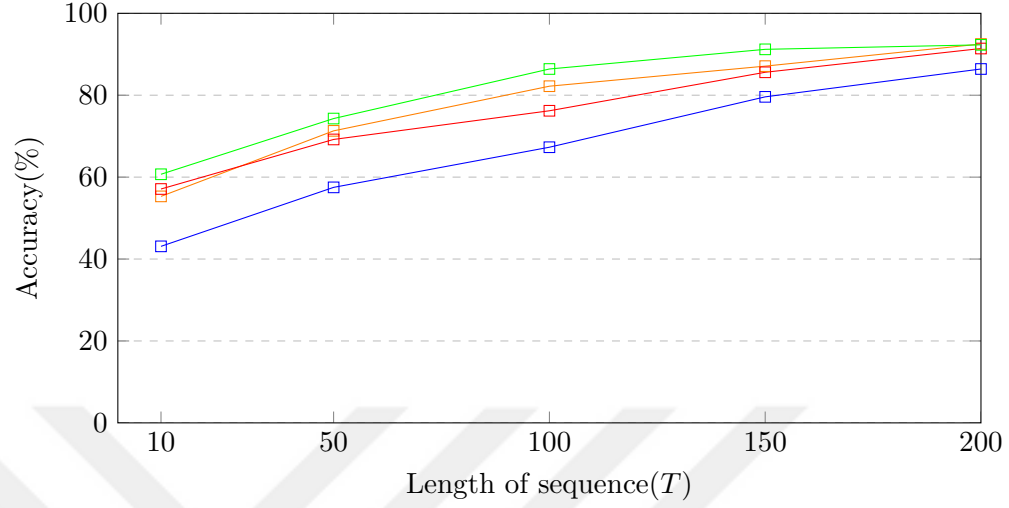
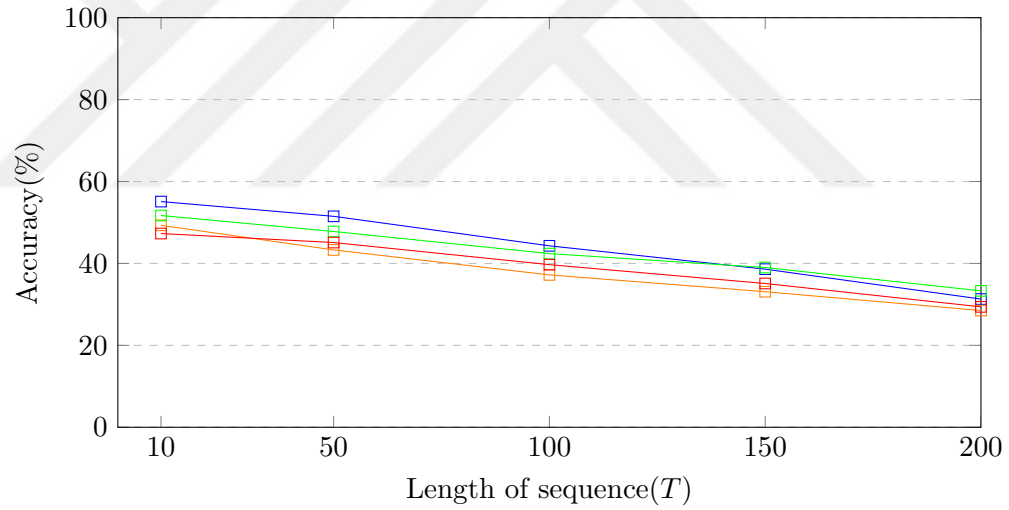
If the results that are shown in Figure-2 are examined, then it is easily seen that the

difference between the running time for *HP1CA* and *HMM* is the counter value factor. For instance, let examine both Figure 6.2a and Figure 6.2b. If only 200 lengths of observation sequence of the datasets are considered, the time taken by *HMM* is approximately equal to 500 seconds, and for the *HP1CA* it is 2000 seconds, respectively. Therefore, *HP1CA* is four times slower than *HMM*.

In that case, the same slower rate approximately is observed when four different hidden variables are used in both models, which process the same length dataset. This slowing rate shows change depending on the maximum counter value. The length of the counter value (i.e.  $M$ ) in different models will reveal the different slowing rate.

The original version of the *Viterbi* algorithm is a kind of dynamic programming algorithm used on *HMM* to determine the best sequence of the hidden variable that might generate the given observation sequence. The adopted version of this algorithm has been already studied in [93] and the time complexity is indicated as the same complexity of a *HP1CA* has. In the test stage, this algorithm provides an alternative way to compare the performance of both *HP1CA* and *HMM*. One of the results to be obtained through the testing step is to compare the accuracy of the best hidden state-counter pair sequence that could produce the given output sequence with the original hidden state-counter pair.

It is worth explaining that  $A^0$  is the only transition matrix that is used during the transition flow on *HMM*, which has no counter value in its structure. Thus, the counter value is always considered to be zero at learning and testing steps on *HMM*. This aspect generally points to the way that the *HMM* has worse accuracy results upon the best hidden variable-counter pair sequence. As it is shown in Figure 6.3b, the *HMM* models have a very low accuracy rate, while the *HP1CA* models have a relatively better accuracy rate. The most obvious reason for this certain difference is the counter factor that varies in a time interval. The dataset that is used in both models to present the accuracy of their performance is generated using a counter, which also varies in this time interval.

(a) The accuracy of *HP1CA*(b) The accuracy of *HMM*Figure 6.3: The time scales of both models of *HP1CA* and *HMM* over the same datasets

The third category of the comparison of the models at test step is the probability of the best sequence of hidden state-counter pair of the model. There is one best hidden state-counter pair sequence for each individual observation sequence and a probability of that

sequence. The principle of the processing of the adapted *Viterbi* algorithm proceeds the transition from one hidden state-counter pair to another pair by choosing the existing path which has maximum probability value instead of summing over all existing paths which are reached to the next pair.

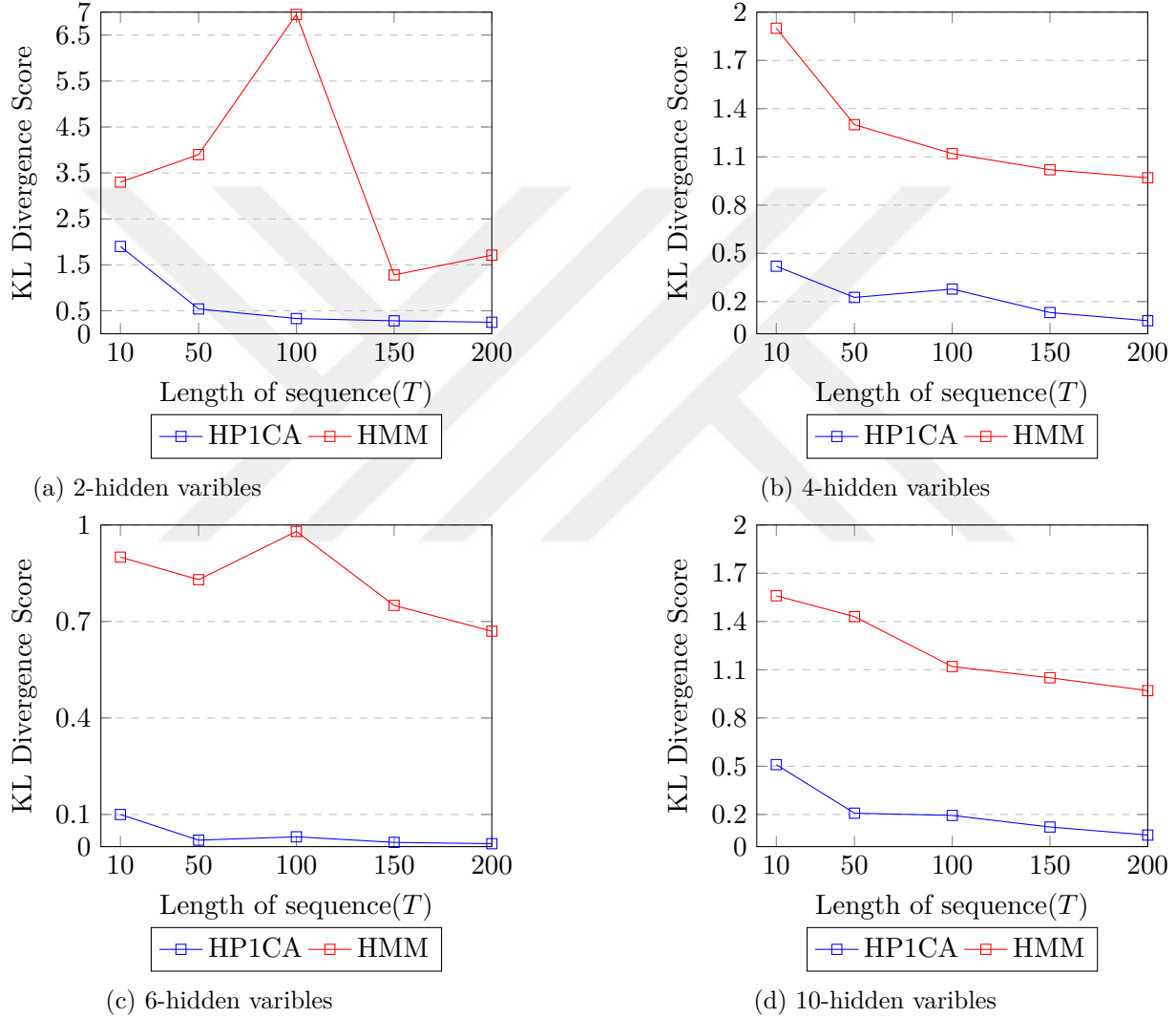


Figure 6.4: The time scales of both models of *HP1CA* and *HMM* over the same datasets

Therefore, the models when they complete the learning step to update their parameters obtain a score by comparing the maximum probability value with original values. This

score is determined by using *Kullback-Leibler*(KL) divergence. This method measures the difference between two probability distributions. Thus, at the end of the testing step, there is an approximated distribution that is obtained from the model and true distribution which is obtained by using the original parameters. If this two distributions perfectly match, then the score is equal to zero otherwise it is equal to a value between 0 and  $\infty$ .

Therefore, four different KL scores are shown in Figure 6.4 for four models that consist of various hidden variables. Each graph represents that KL scores obtained from *HP1CA* models are closer to zero more than KL scores of *HMM*. That is an expected result according to the accuracy of those models.

As a result, the time complexity of *HMM* is expected to be faster than *HP1CA* where it is based on the compared time complexity. The counter parameter  $M$  obviously causes that one extra loop to be added over the nested loops of the length of the observation sequence  $T$ , and twice over the number of the hidden variable  $N$ . Thus, *HP1CA* is  $M$  times slower than *HMM*. However, when the accuracy rate of both models is compared, it is observed that *HP1CA* has an overwhelmingly better accuracy rate than *HMM* has when the models especially process long observation sequences. KL divergence can be used to compare how much the probabilities of a model are close to the original probability values that are obtained by using the original parameters. As expected, KL values of *HP1CA* are much lower than KL scores of *HMM* due to the fact that they are closest to the original model, which presents that a *HP1CA* obtains a closer probability distribution as compared with *HMM*.

## 6.2 Implementation and results of other HP1CA models

In the previous model, the test results of the Model-1 type of *HP1CA* is compared with the test result of *HMM* by using the different number of hidden variables and length of observation sequence. In this section, the performance of the remaining five model types of *HP1CA* will be examined against *HMM* using the 6 hidden states and 200 length observation sequences. The computation time of learning, the accuracy of the models, and KL scores will be compared with *HMM* as done in previous section.

Firstly, the computation time of those five models will be compared with *HMM* in the

learning stage provided that the number of iteration remains the same. As it is shown in Figure 6.5, *HMM* again has less computation time which is expected. The reason of obtaining this result is that the factor of counter value that is mentioned in Section 6.1 for Model-1.

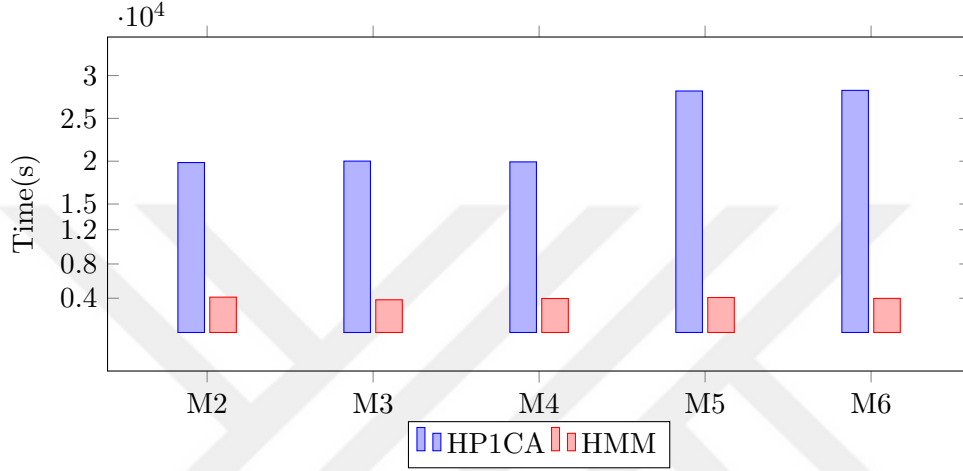


Figure 6.5: The computation times of five model types of HP1CA and HMM

It is worthy to determine that the difference between the processed of those five *HP1CA* models is either the number of initial and final terminals with a zero counter value is more than one or the counter value of the final terminal is non-zero.

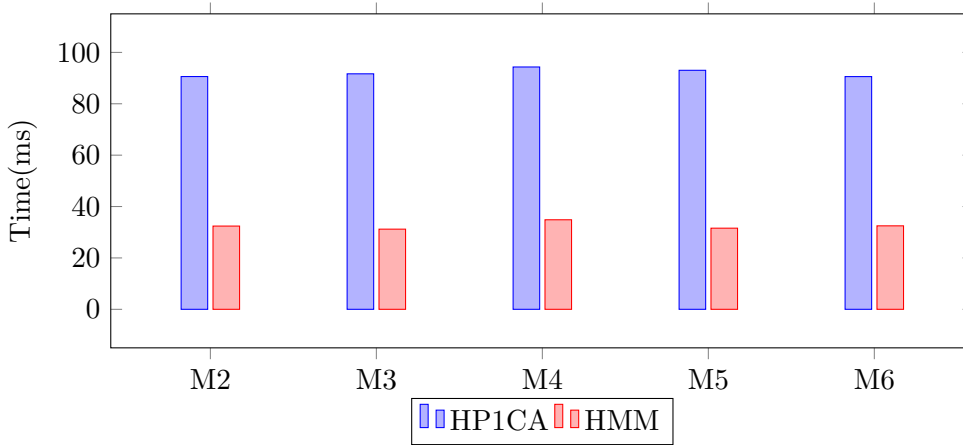
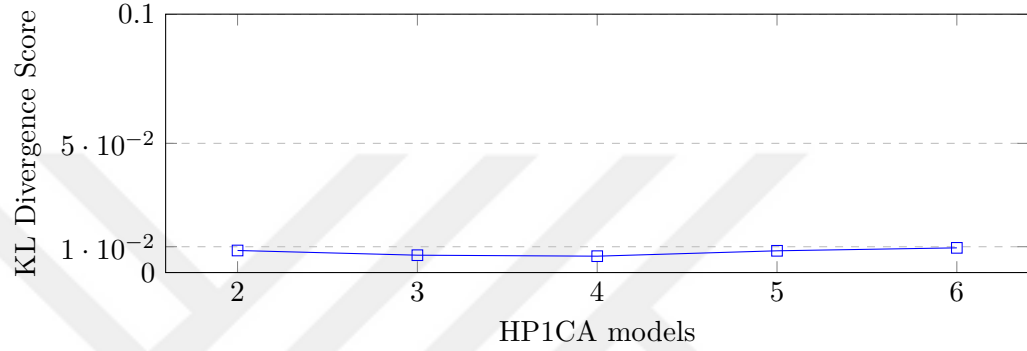
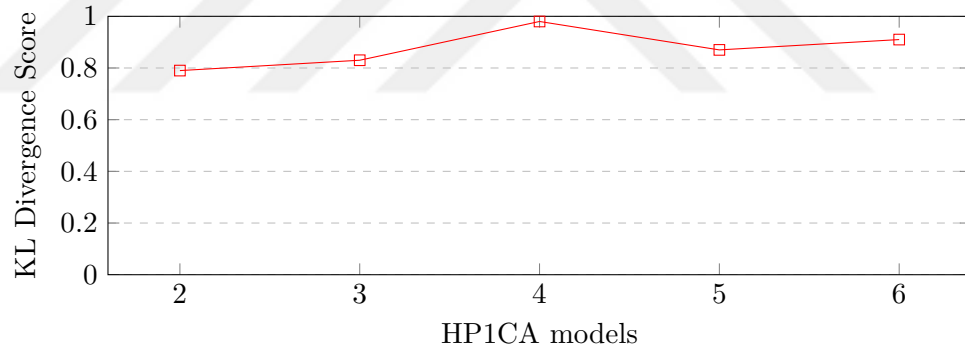


Figure 6.6: The accuracy of five model types of HP1CA and HMM

Therefore, there will be models whose computation times of learning might be close to each other. Model-5 and Model-6 have slightly higher computation time, as seen in Figure 6.5. The most obvious reason is that the counter value of the final terminal used in these models is different from zero and can be any value between 0 and the length of observation sequence (i.e.  $0 \leq M \leq T$ ).



(a) The KL values of HP1CAs



(b) The KL values of HMM

Figure 6.7: The accuracy of five model types of HP1CA and HMM

In the testing part, the investigated five new models of *HP1CA*, where there is a difference only on the definition of their initial and final terminal states, might observe similar accuracy rates to each other. As it is shown in Figure 6.6, the results of the accuracy of those new five models are close to each other but of course, each of them has a better accuracy rate than *HMM* obtained. In addition, it is observed that the Model-4 type has a better rate among those five new *HP1CA* models. Multiple initial and final terminals states that

both have zero counter value are used in Model-4.

As it is known that a *HMM* usually consists of multiple initial states, which is commonly called initial vector, instead of a single initial terminal. In addition, a unique final terminal might sometimes be specified on *HMM*, although typically all states are considered to be final terminal states. According to this context, Model-4, which has multiple initial and final terminals with zero counter values, is structurally closest type to *HMM*.

The observed KL values for each model are shown in Figure 6.7. As expected, KL values of *HP1CA* models show better results by being closest to zero than *HMM*. On the other hand, Model-4 gets the best KL score while the *HMM* gets the worst KL score compared to the original model. We conjecture that in this case, Model-4 is the model that stands out among the new *HP1CA* models and it produces the best results against *HMM*.

In conclusion, in the second part of this comparison, the performance of the new five different *HP1CA* models with respect to the computation time, accuracy and approximation to the original probability value (i.e. KL divergence score) were measured against *HMM*. As a result of this, it is observed that *HP1CA* models were superior to *HMM* in all aspects except for the computation time complexity. Finally, Model-4, which is the closest in its structure to *HMM*, has the best performance among *HP1CA* models according to this results.

## Chapter 7

# Conclusion

We have started with finite automata and augmented it with probabilistic transitions. We have seen that the stochastic version of finite automata corresponds to a Markovian model that is called the Hidden Markov Model (*HMM*). The definition and the link between these state-space models were studied in Section 3.1.1 in Chapter 3.

When we augment finite automata with memory, the resulting transition systems can have infinite number of states. By augmenting finite automata with stack, one obtain push-down automata (*PDA*) where information is added or removed according to the last-in, first-out(LIFO) regime. Prediction to the next action needs a probabilistic feature to abstract the behavior of the machine. For this reason, we examined probabilistic pushdown automata (*PPDA*) which is the stochastic version of *PDA*. The transition systems of that model makes use of probability. When we use a restricted stack, we obtained a counter machine where the machine uses only one stack symbol. Moreover, the probabilistic version of this counter machine which is the subclass of the *PPDA* which is called probabilistic once-counter automata (*P1CA*) constitutes the main contribution model of this thesis. This stochastic counter machine can be obtained by associating the probabilities to the transitions of the deterministic one-counter machine.

We have seen that for each class of those systems there exists a suitable formal language that is generated by the corresponding grammar. For instance, state-space models (e.g. *HMM*, *pfsm*) accepts the regular languages (*RL*) that can be generated using regular grammars. Moreover, context-free languages (*CFL*), which is a superset of *RL*, are accepted by infinite state-space models (e.g. *PDA*). As the counter machine is a subclass of

*PDA*, then the languages that are accepted by *OCA* can be a subclass of *CFL*. However, the time complexity of the learning problem is cubic (i.e.  $\mathcal{O}(n^3)$ ) for *CFL*. To reduce this cost, we defined a new probabilistic counter machine and adopted fundamental algorithms of *HMM* to it.

Our new model, hidden probabilistic one-counter automata (*HP1CA*), is a *P1CA* where the control states and the counter values are invisible. This model generalizes the behavior of *HMM* and *P1CA*. The machine considers non-negative counter which can be incremented or decremented during transitions, and such transitions can be enabled/disabled depending on whether the counter is equal to 0 or not. However, the value of the counter and control states are not observed. The only observed variable is output symbols that are generated by the control state variable of the model. This aspect is adopted from *HMM* in which the only visible variable is observation symbols.

The way the interaction to update the parameters of *HP1CA* is made explicit depends on the adaptation of the learning algorithm for *HMM*. We reduced the cubic time complexity of the estimation algorithm of *CFL* by adapting the *Baum-Welch* algorithm for *HMM* instead. This algorithm is a special case of *Expectation-Maximization (EM)* algorithm used to find the unknown parameters of *HMM*. It consists of two main steps which are *E-step* and *M-step*. We adapted two other fundamental algorithms of *HMM* which are *Forward* algorithm and *Backward* algorithm to calculate the *E-step*. In *M-step*, we defined the formulae to update the parameters of the model by applying *Lagrange Multiplier*. All of the adapted algorithms that are used in *HP1CA* have quadratic cost where there one extra loop that goes over all possible counter values, which is not necessarily proportional to the length of observation sequence because. Normally, transitions that raise the counter value are limited in number. The maximum value of the counter may even be limited by a constant.

After augmenting *HP1CA* with a different possibilities for the definition of the initial or final states, we obtain five more *HP1CA* models. First, an *HP1CA* can either start in (1) a fixed control state with counter zero or (2) have multiple initial control states, and some fixed probability distribution specifies the likelihood of starting in a given control state (with a counter value zero). Second, an *HP1CA* may be required to end with with (A) a unique final control state with counter value zero, (B) a unique final control state

with arbitrary counter value, (C) multiple final control states with zero counter value. The (1-A) model, which has only one starting and final terminal with zero counter value, was studied in Chapter 4. The rest of the models were defined in Chapter 5 along with the adapted algorithms for their analysis.

We demonstrated that for each type of *HP1CA* model gives a better accuracy and less error ratio than *HMM* in the experimental section part. We proposed learning algorithm of the parameters of each model that run in quadratic time in the length of the input (the observation sequence). The same algorithms for *HMM* have the running time smaller by a factor of the maximum counter value encountered during any run of the *HP1CA*.

In this thesis, we have been mainly concerned with a new stochastic one counter machine models that generalize *HMM*. We have chosen our definitions as close as possible to *P1CA* and *HMM* and reduced the time complexity of the resulting model from cubic to quadratic. In the future, other variations of definitions of *HP1CA* can be explored. For instance, in the queuing theory, discrete-time Quasi-Birth-Death Processes (*QBD*) is a countable state Markov chain whose transition matrix has a block structure. According to [41], *QBD* is equivalent, in a precise sense, to *P1CA*. Thus, it would be worthwhile to extend the *QBD* model, just like we extended *P1CA* to *HP1CA*, and study learning algorithms for such an extension.

# Bibliography

- [1] Steven Abney, David McAllester, and Fernando Pereira. Relating probabilistic grammars and automata. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 542–549, College Park, Maryland, USA, June 1999. Association for Computational Linguistics.
- [2] Lakhdar Akroun and Gwen Salaün. Automated verification of automata communicating via fifo and bag buffers. *Form. Methods Syst. Des.*, 52(3):260–276, June 2018.
- [3] Rajeev Alur and D. L. Dill. Automata for modeling real-time systems. In *Proceedings of the Seventeenth International Colloquium on Automata, Languages and Programming*, pages 322–335, Berlin, Heidelberg, 1990. Springer-Verlag.
- [4] Javed Ashraf, Naveed Iqbal, Naveed Sarfraz Khattak, and Ather Mohsin Zaidi. Speaker independent urdu speech recognition using hmm. In Christina J. Hopfe, Yacine Rezgui, Elisabeth Métais, Alun Preece, and Haijiang Li, editors, *Natural Language Processing and Information Systems*, pages 140–148, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
- [5] Domagoj Babić and Zvonimir Rakamarić. Asynchronously communicating visibly pushdown systems. In Dirk Beyer and Michele Boreale, editors, *Formal Techniques for Distributed Systems*, pages 225–241, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
- [6] Soheil Bahrampour, Asok Ray, Soumalya Sarkar, Thyagaraju Damarla, and Nasser M. Nasrabadi. Performance comparison of feature extraction algorithms for target detection and classification. *Pattern Recognition Letters*, 34(16):2126 – 2134, 2013.

- [7] Marco Baioletti, Giuseppe Busanello, and Barbara Vantaggi. Conditional independence structure and its closure: Inferential rules and algorithms. *International Journal of Approximate Reasoning*, 50(7):1097 – 1114, 2009. Special Section on Graphical Models and Information Retrieval.
- [8] Borja Balle, Jorge Castro, and Ricard Gavaldà. Learning probabilistic automata: A study in state distinguishability. *Theoretical Computer Science*, 473:46 – 60, 2013. Special Issue on Algorithmic Learning Theory.
- [9] Leonard E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. In Oved Shisha, editor, *Inequalities III: Proceedings of the Third Symposium on Inequalities*, pages 1–8, University of California, Los Angeles, 1972. Academic Press.
- [10] Leonard E. Baum and J. A. Eagon. An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *Bull. Amer. Math. Soc.*, 73(3):360–363, 05 1967.
- [11] Leonard E. Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *Ann. Math. Statist.*, 37(6):1554–1563, 12 1966.
- [12] Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Ann. Math. Statist.*, 41(1):164–171, 02 1970.
- [13] Moritz Beyreuther, Conny Hammer, Joachim Wassermann, Matthias Ohrnberger, and Tobias Megies. Constructing a Hidden Markov Model based earthquake detector: application to induced seismicity. *Geophysical Journal International*, 189(1):602–610, 04 2012.
- [14] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*, pages 359–361. Springer-Verlag, Berlin, Heidelberg, 2006.
- [15] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*, pages 615–625. Springer-Verlag, Berlin, Heidelberg, 2006.
- [16] Ahmed Bouajjani, Javier Esparza, and Oded Maler. Reachability analysis of push-down automata: Application to model-checking. In Antoni Mazurkiewicz and Józef

- Winkowski, editors, *CONCUR '97: Concurrency Theory*, pages 135–150, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.
- [17] Tomas Brazdil, Vaclav Brozek, and Kousha Etessami. One-counter stochastic games. In *FSTTCS*, 2010.
  - [18] Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. Class-based  $n$ -gram models of natural language. *Computational Linguistics*, 18(4):467–480, 1992.
  - [19] Tomáš Brázdil, Javier Esparza, Stefan Kiefer, and Antonín Kucera. Analyzing probabilistic pushdown automata. *Formal Methods in System Design*, 43, 10 2013.
  - [20] S. Sarkar M. Yasar C. Rao, A. Ray. Review and comparative evaluation of symbolic dynamic filtering for detection of anomaly patterns. 3(101), 2009.
  - [21] Ogawa M. Cai X. Well-structured pushdown systems. In *D’Argenio P.R., Melgratti H. (eds) CONCUR 2013 – Concurrency Theory. CONCUR 2013.*, Berlin, Heidelberg, Germany, 2013. Springer.
  - [22] Olivier Carton, Bruno Guillon, and Fabian Reiter. Counter machines and distributed automata: A story about exchanging space and time. *CoRR*, abs/1804.03582, 2018.
  - [23] Daniel Chambers, Jenny Baglivo, J. Ebel, and Alan Kafka. Earthquake forecasting using hidden markov models. *Pure and Applied Geophysics*, 169:625–639, 11 2014.
  - [24] J.-C. Chappelier, M. Rajman, and Ch-Lausanne. A generalized cyk algorithm for parsing stochastic cfg, 1998.
  - [25] I. Chattopadhyay and A. Ray. Structural transformations of probabilistic finite state machines. *International Journal of Control*, 81(5):820–835, 2008.
  - [26] Rupali S Chavan and Ganesh. S Sable. An overview of speech recognition using hmm. volume 2, page 233 – 238, June 2013.
  - [27] Z. Chen and E. N. Brown. State space model. *Scholarpedia*, 8(3):30868, 2013. revision #189565.
  - [28] Mejda Chihaoui, Wajdi Bellil, Akram Elkefi, and Chokri Ben Amar. Face recognition using hmm-lbp. pages 249–258, 11 2016.

- [29] J.J.F. Commandeur and S.J. Koopman. *An Introduction to State Space Time Series Analysis*, pages 9–78. Oxford University Press, 2007.
- [30] Heike Huebener Mihaela Mihailescu Ulrich Cubasch Constantin Mares, Ileana Mares and Petre Stanciu. A hidden markov model applied to the daily spring precipitation over the danube basin. *Advances in Meteorology*, 2014, 2014.
- [31] A. Philip Dawid. Conditional independence for statistical operations. *Ann. Statist.*, 8(3):598–617, 05 1980.
- [32] Giorgio Delzanno and Andreas Podelski. Model checking in clp. In W. Rance Cleaveland, editor, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 223–239, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.
- [33] Norbert Dojer, Anna Gambin, Andrzej Mizera, Bartek Wilczyński, and Jerzy Tiuryn. Applying dynamic bayesian networks to perturbed gene expression data. *BMC bioinformatics*, 7:249, 02 2006.
- [34] P. Dupont, F. Denis, and Y. Esposito. Links between probabilistic automata and hidden markov models: probability distributions, learning models and induction algorithms. *Pattern Recognition*, 38(9):1349 – 1371, 2005. Grammatical Inference.
- [35] Jason Eisner. Inside-outside and forward-backward algorithms are just backprop (tutorial paper). In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 1–17, Austin, TX, November 2016. Association for Computational Linguistics.
- [36] Javier Esparza. Verification of systems with an infinite state space. In *Proceedings of the 4th Summer School on Modeling and Verification of Parallel Processes*, MOVEP ’00, pages 183–186, London, UK, UK, 2001. Springer-Verlag.
- [37] Javier Esparza. *Automata theory An algorithmic approach*, pages 17–26. 2017.
- [38] Javier Esparza, David Hansel, Peter Rossmanith, and Stefan Schwoon. Efficient algorithms for model checking pushdown systems. In *Proceedings of the 12th International Conference on Computer Aided Verification*, CAV ’00, pages 232–247, London, UK, UK, 2000. Springer-Verlag.
- [39] Javier Esparza, Antonin Kucera, and Richard Mayr. Model checking probabilistic pushdown automata. In *Proceedings of the 19th Annual IEEE Symposium on Logic*

- in Computer Science*, LICS '04, pages 12–21, Washington, DC, USA, 2004. IEEE Computer Society.
- [40] Javier Esparza, Antonín Kučera, and Richard Mayr. Quantitative analysis of probabilistic pushdown automata: Expectations and variances. In *Proceedings of the Twentieth Annual IEEE Symposium on Logic in Computer Science (LICS 2005)*, pages 117–126. IEEE Computer Society Press, June 2005.
  - [41] Kousha Etessami, Dominik Wojtczak, and Mihalis Yannakakis. Quasi-birth-death processes, tree-like qbds, probabilistic 1-counter automata, and pushdown systems. *Perform. Eval.*, 67(9):837–857, September 2010.
  - [42] Kousha Etessami, Dominik Wojtczak, and Mihalis Yannakakis. Quasi-birth-death processes, tree-like qbds, probabilistic 1-counter automata, and pushdown systems. *Performance Evaluation*, 67:837–857, 09 2010.
  - [43] Kousha Etessami and Mihalis Yannakakis. Recursive markov chains, stochastic grammars, and monotone systems of nonlinear equations. volume 56, pages 340–352, 02 2005.
  - [44] Patrick C. Fischer. Multi-tape and infinite-state automata—a survey. *Commun. ACM*, 8(12):799–805, December 1965.
  - [45] Patrick C. Fischer, Albert R. Meyer, and Arnold L. Rosenberg. Counter machines and counter languages. *Mathematical systems theory*, 2(3):265–283, Sep 1968.
  - [46] Vojtěch Forejt, Petr Jančar, Stefan Kiefer, and James Worrell. Game characterization of probabilistic bisimilarity, and applications to pushdown automata, 2017.
  - [47] Viliam Geffert. Magic numbers in the state hierarchy of finite automata. *Information and Computation*, 205(11):1652 – 1670, 2007.
  - [48] Steven M. German and A. Prasad Sistla. Reasoning about systems with many processes. *J. ACM*, 39(3):675–735, July 1992.
  - [49] A. Gupta and B. Dhingra. Stock market prediction using hidden markov models. In *2012 Students Conference on Engineering and Systems*, pages 1–4, March 2012.

- [50] John E Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages and computation*, pages 225–241. Boston, [MA] : Pearson/Addison Wesley, 3rd edition edition, 2007. Previous ed.: London : Addison-Wesley, 2001.
- [51] John E. Hopcroft and Jeff D. Ullman. *Introduction to Automata Theory, Languages, and Computation*, pages 37–83. Addison-Wesley Publishing Company, 1979.
- [52] John E. Hopcroft and Jeffrey D. Ullman. *Formal Languages and Their Relation to Automata*, pages 26–33. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1969.
- [53] Jianying Hu, Michael Brown, and William Turin. Hmm based on-line handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18:1039–1045, 11 1996.
- [54] Oliver C. Ibe. 3 - introduction to markov processes. In Oliver C. Ibe, editor, *Markov Processes for Stochastic Modeling (Second Edition)*, pages 49 – 57. Elsevier, Oxford, second edition edition, 2013.
- [55] E. T. Jaynes. *Probability theory: The logic of science*, pages 24–30. Cambridge University Press, Cambridge, 2003.
- [56] X. Jin, S. Sarkar, A. Ray, S. Gupta, and T. Damarla. Target detection and classification using seismic and pir sensors. *IEEE Sensors Journal*, 12(6):1709–1718, June 2012.
- [57] Michael I. Jordan. Graphical models. *Statist. Sci.*, 19(1):140–155, 02 2004.
- [58] Jagdish Joshi. Prediction of weather states using hidden markov model. 12 2012.
- [59] Jagdish Joshi, Kumar Tankeshwar, and Sunita Srivastava. Hidden markov model for quantitative prediction of snowfall and analysis of hazardous snowfall events over indian himalaya. *JESS*, 126, 05 2017.
- [60] Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2Nd Edition)*, pages 548 – 551. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, third edition edition, 2019.
- [61] Rudolph Emil Kalman and Others. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

- [62] Samuel Karlin and Howard E. Taylor. *A First Course in Stochastic Processes*, pages 29 – 30. Academic Press, 2012.
- [63] Genshiro Kitagawa. Self-organizing state space model. 1998.
- [64] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*, pages 15–60. The MIT Press, 2009.
- [65] Dimitris Konovessis, Wenkui Cai, and Dracos Vassalos. Development of bayesian network models for risk-based ship design. *Journal of Marine Science and Application*, 12(2):140–151, Jun 2013.
- [66] Antonín Kučera. The complexity of bisimilarity-checking for one-counter processes. *Theor. Comput. Sci.*, 304(1):157–183, July 2003.
- [67] Antonín Kučera and Petr Jančar. Equivalence-checking on infinite-state systems: Techniques and results. *Theory Pract. Log. Program.*, 6(3):227–264, May 2006.
- [68] T. Lin. Model-checking branching-time properties of probabilistic automata and probabilistic one-counter automata, 2015.
- [69] Peter Linz. An introduction to formal languages and automata, 4th edition. pages 47–77, 2006.
- [70] Srabanti Maji and Deepak Garg. Gene finding using hidden markov model. *Journal of Applied Sciences*, 12, 2012.
- [71] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*, pages 315 – 320. MIT Press, Cambridge, MA, USA, 1999.
- [72] Sascha Mergner. *Applications of State Space Models in Finance*, pages 17–43. 2009.
- [73] Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted finite-state transducers in speech recognition. *Computer Speech and Language*, 16(1):69 – 88, 2002.
- [74] M. A. Moni and A. B. M. S. Ali. Hmm based hand gesture recognition: A review on techniques and approaches. In *2009 2nd IEEE International Conference on Computer Science and Information Technology*, pages 433–437, Aug 2009.

- [75] A. V. Nefian and M. H. Hayes. Face detection and recognition using hidden markov models. In *Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No.98CB36269)*, volume 1, pages 141–145 vol.1, Oct 1998.
- [76] Nguyet Nguyen. Hidden Markov Model for Stock Trading. *International Journal of Financial Studies*, 6(2):1–17, March 2018.
- [77] Ahmet Okutan and Olcay Taner Yıldız. Software defect prediction using bayesian networks. *Empirical Software Engineering*, 19(1):154–181, Feb 2014.
- [78] Zuzanna Parcheta and Carlos-D. Martínez-Hinarejos. Sign language gesture recognition using hmm. In Luís A. Alexandre, José Salvador Sánchez, and João M. F. Rodrigues, editors, *Pattern Recognition and Image Analysis*, pages 419–426, Cham, 2017. Springer International Publishing.
- [79] Emanuel Parzen. *Stochastic Processes*, pages 188–195. Courier Dover Publications, 2015.
- [80] Azaria Paz. Chapter i - stochastic sequential machines. In Azaria Paz, editor, *Introduction to Probabilistic Automata*, pages 1 – 66. Academic Press, 1971.
- [81] Judea Pearl. Probabilistic reasoning in intelligent systems - networks of plausible inference. In *Morgan Kaufmann series in representation and reasoning*, 1988.
- [82] Judea Pearl. *Graphical models for probabilistic and causal reasoning*, pages 70–1. 01 2004.
- [83] Jakob Skou Pedersen and Jotun Hein. Gene finding with a hidden Markov model of genome structure and evolution. *Bioinformatics*, 19(2):219–227, 01 2003.
- [84] Jean-Eric Pin. *Mathematical Foundations of Automata Theory*, pages 42–48. 2019.
- [85] Nir Piterman and Moshe Y. Vardi. Global model-checking of infinite-state systems. In Rajeev Alur and Doron A. Peled, editors, *Computer Aided Verification*, pages 387–400, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.
- [86] Alexander S. Poznyak. 10 - markov processes. In Alexander S. Poznyak, editor, *Advanced Mathematical Tools for Automatic Control Engineers: Stochastic Techniques*, pages 263 – 286. Elsevier, Oxford, 2009.

- [87] Stephen Procter, John Illingworth, and F. Mokhtarian. Cursive handwriting recognition using hidden markov models and a lexicon-driven level building algorithm. *Vision, Image and Signal Processing, IEE Proceedings -*, 147:332 – 339, 09 2000.
- [88] Xinghao Qiao, Shaojun Guo, and Gareth M. James. Functional graphical models. *Journal of the American Statistical Association*, 114(525):211–222, 2019.
- [89] Lawrence R. Rabiner. Readings in speech recognition. chapter A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, pages 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.
- [90] Venkatesh Rajagopalan and Asok Ray. Symbolic time series analysis via wavelet-based partitioning. *Signal Processing*, 86(11):3309 – 3320, 2006. Special Section: Distributed Source Coding.
- [91] Asok Ray. Symbolic dynamic analysis of complex systems for anomaly detection. *Signal Processing*, 84(7):1115 – 1130, 2004.
- [92] Rosenthal Jeffrey S. *A First Look at Rigorous Probability Theory*, pages 7 – 27. World Scientific Publishing Co.Pte.Ltd., second edition edition, 2006.
- [93] Alexander Sakharov and Timothy Sakharov. The viterbi algorithm for subsets of stochastic context-free languages. *Information Processing Letters*, 135:68 – 72, 2018.
- [94] Ferdinando Samaria. *Face recognition using Hidden Markov Models*. PhD thesis, 02 1995.
- [95] Alexander Schliep, Alexander Schönhuth, and Christine Steinhoff. Using hidden markov models to analyze gene expression time course data. *Bioinformatics (Oxford, England)*, 19 Suppl 1:i255–63, 02 2003.
- [96] Regev Schweiger, Yaniv Erlich, and Shai Carmi. Factorialhmm: Fast and exact inference in factorial hidden markov models. *bioRxiv*, 2018.
- [97] Stefan Schwoon. Model-checking pushdown systems. pages 8–94, 01 2002.
- [98] Luc Segoufin. Automata and logics for words and trees over an infinite alphabet. In Zoltán Ésik, editor, *Computer Science Logic*, pages 41–57, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

- [99] Rajen Shah and Jonas Peters. The hardness of conditional independence testing and the generalised covariance measure. 04 2018.
- [100] Matthew Simon. *Pushdown Automata*, chapter 7, pages 299–328.
- [101] Rastislav Šrámek, Broňa Brejová, and Tomáš Vinař. On-line viterbi algorithm for analysis of long biological sequences. In Raffaele Giancarlo and Sridhar Hannenhalli, editors, *Algorithms in Bioinformatics*, pages 240–251, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.
- [102] Mariëlle Stoelinga. An introduction to probabilistic automata. *Bulletin of the EATCS*, 78, 06 2004.
- [103] Liangjun Su and Halbert White. Testing conditional independence via empirical likelihood. *Journal of Econometrics*, 182(1):27 – 44, 2014. Causality, Prediction, and Specification Analysis: Recent Advances and Future Directions.
- [104] Nuri Tasdemir and A. C. Cem Say. Models of pushdown automata with reset. In Giancarlo Mauri and Alberto Leporati, editors, *Developments in Language Theory*, pages 417–428, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
- [105] Wolfgang Thomas. A short introduction to infinite automata. In Werner Kuich, Grzegorz Rozenberg, and Arto Salomaa, editors, *Developments in Language Theory*, pages 130–144, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.
- [106] Salvatore Torre, P. Madhusudan, and Gennaro Parlato. An infinite automaton characterization of double exponential time. In *Proceedings of the 22Nd International Workshop on Computer Science Logic, CSL '08*, pages 33–48, Berlin, Heidelberg, 2008. Springer-Verlag.
- [107] Wen-Guey Tzeng. Learning probabilistic automata and markov chains via queries. *Machine Learning*, 8(2):151–166, Mar 1992.
- [108] Leslie G. Valiant and Michael S. Paterson. Deterministic one-counter automata. *Journal of Computer and System Sciences*, 10(3):340 – 350, 1975.
- [109] Tyler J. VanderWeele and James M. Robins. Directed Acyclic Graphs, Sufficient Causes, and the Properties of Conditioning on a Common Effect. *American Journal of Epidemiology*, 166(9):1096–1104, 08 2007.

- [110] Sicco Verwer, Rémi Eyraud, and Colin de la Higuera. Pautomac: a probabilistic automata and hidden markov models learning competition. *Machine Learning*, 96(1):129–154, Jul 2014.
- [111] Enrique Vidal, Franck Thollard, Colin de la Higuera, Francisco Casacuberta, and Rafael C. Carrasco. Probabilistic finite-state machines - part ii. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:1026–1039, 2005.
- [112] Enrique Vidal, Franck Thollard, Colin de la Higuera, Francisco Casacuberta, and Rafael C. Carrasco. Probabilistic finite-state machines-part i. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(7):1013–1025, July 2005.
- [113] Mengqiu Wang and Christopher D. Manning. Probabilistic finite state machines for regression-based MT evaluation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 984–994, Jeju Island, Korea, July 2012. Association for Computational Linguistics.
- [114] Xia Wang and Yongmiao Hong. Characteristic function based testing for conditional independence: A nonparametric regression approach. *Econometric Theory*, 34(4):815–849, 2018.
- [115] Xiaoyan Wang, Ming Xia, Huiwen Cai, Yong Gao, and Carlo Cattani. Hidden-markov-models-based dynamic hand gesture recognition. *Mathematical Problems in Engineering*, 2012, 2012.
- [116] Zhengxiao Wu. A hidden markov model for earthquake declustering. *Journal of Geophysical Research: Solid Earth*, 115(B3), 2010.
- [117] Yang He, Mou-Yen Chen, and A. Kundu. Off-line handwritten word recognition using hmm with adaptive length viterbi algorithm. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3 - Conference C: Signal Processing (Cat. No.94CH3440-5)*, volume 2, pages 460–462 vol.2, Oct 1994.