19CSE214

Theory of Computation

Case Study

Group – 9

RNA & DNA Structure Prediction

Team Members

|  |  |  |
| --- | --- | --- |
| CB.EN.U4CSE22204 | Adithyan.S | Non Accepted Strings |
| CB.EN.U4CSE22221 | Hari Prasaath.S.B | NPDA |
| CB.EN.U4CSE22227 | Kabilan.M.D | DPDA |
| CB.EN.U4CSE22246 | Sai Krishna.K | CFG |
| CB.EN.U4CSE22263 | Sai Krishna Digvijay Chowdary | Accpeted Strings |

Problem Statement:

This project aims to create a deterministic pushdown automaton (DPDA) capable of distinguishing and validating DNA and RNA sequences. Leveraging specific base-pairing rules for each molecule, the DPDA will accurately identify and accept sequences that adhere to DNA and RNA structural constraints, facilitating biological data processing and analysis.

Introduction:

Understanding the structure and function of DNA and RNA molecules is fundamental in biological research and applications. However, ensuring the validity of genetic sequences poses a significant challenge due to their complex base-pairing rules. In this case study, we explore the development of a deterministic pushdown automaton (DPDA) to accurately identify and validate DNA and RNA sequences based on their specific structural characteristics. By leveraging computational methods, this approach aims to provide a reliable tool for parsing and analyzing genetic information, contributing to advancements in molecular biology and genetic engineering.

Context Free Grammar:

1. Terminal Symbols - A, T, C, G, U, D, R
2. Non – Terminal Symbols - S, D, R, DS, RS
3. Production Rules
   * Start Rules
     + S -> DDS
     + S -> RRS
   * DNA
     + DS -> λ
     + DS ->ADST
     + DS ->TDSA
     + DS ->CDSG
     + DS ->GDSC
   * RNA
     + RS -> λ
     + RS -> ARSU
     + RS -> URSA
     + RS -> CRSG
     + RS -> GRSC

Explanation:

This CFG works as follows:

* The start symbol S can expand to either a DNA sequence (DDS) or an RNA sequence (RRS).
* D and R are terminal symbols that indicate the start of a DNA or RNA sequence, respectively.
* DS handles the production rules for DNA sequences, ensuring correct pairing (A with T, T with A, C with G, and G with C).
* RS handles the production rules for RNA sequences, ensuring correct pairing (A with U, U with A, C with G, and G with C).

Example Derivations:

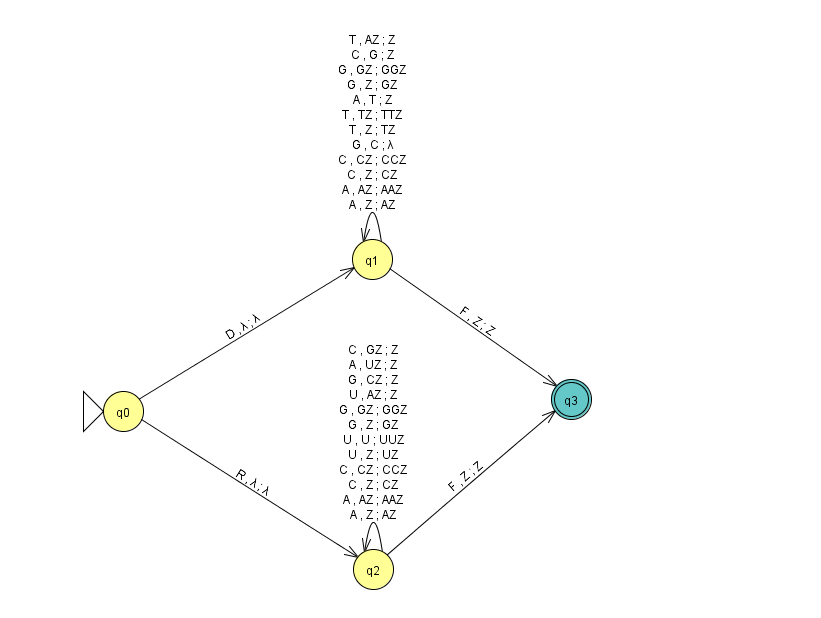
* DNA
  + S
  + ->DDS
  + ->DADST
  + ->DA(CDSG)T
  + ->DA(C λG)T
  + Result – DACTG (One Example)
* RNA
  + S
  + ->RRS
  + ->RARSU
  + ->RA(CRSG)U
  + ->RA(C λG)U
  + Result – RAUCG (One Example)

DPDA:

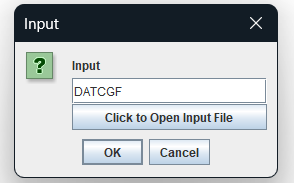
1. States
   1. q0 – Start State
   2. qd – State for processing DNA Sequences
   3. dr – State for processing RNA Sequences
   4. qf – Accepting State
2. Alphabet
   1. Input Symbols – {D, R, A, T, C, G, U}
   2. Stack Symbols - {A, T, U, C, G, Z(Bottom Stack Marker)}
3. Transistion Functions
   1. Initialization:
      1. (q0,D, λ) -> (qd, Z)
      2. (q0,R, λ) -> (qr, Z)
   2. DNA Processing in qd:
      1. (qd,A, Z) -> (qd,AZ)
      2. (qd,A A) -> (qd, AA)
      3. (qd,T, A) -> (qd, λ)
      4. (qd,C, Z) -> (qd, CZ)
      5. (qd,C,C) -> (qd,CC)
      6. (qd,G, C) -> (qd, λ)
      7. (qd,T, Z) -> (qd,TZ)
      8. (qd,T, T) -> (qd, TT)
      9. (qd,A,T) -> (qd, λ)
      10. (qd,G, Z) -> (qd, GZ)
      11. (qd,G, G) -> (qd,GG)
      12. (qd,C, G) -> (qd, λ)
      13. (qr,F,Z) -> (qr, Z)
   3. RNA Processing in qrna:
      1. (qr,A, Z) -> (qr,AZ)
      2. (qr,A A) -> (qr, AA)
      3. (qr,U, A) -> (qr, λ)
      4. (qr,C, Z) -> (qr, CZ)
      5. (qr,C,C) -> (qr,CC)
      6. (qr,G, C) -> (qr, λ)
      7. (qr,U, Z) -> (qr,UZ)
      8. (qr,U, U) -> (qr, UU)
      9. (qr,A,U) -> (qr, λ)
      10. (qr,G, Z) -> (qr, GZ)
      11. (qr,G, G) -> (qr,GG)
      12. (qr,C, G) -> (qr, λ)
      13. (qr,F,Z) -> (qr, Z)

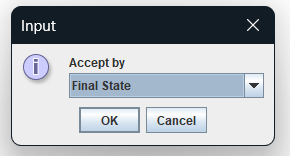
Transition Functions Explanation:

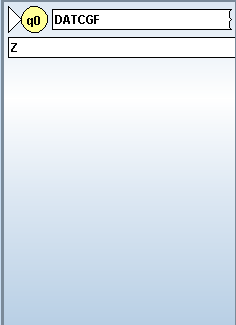
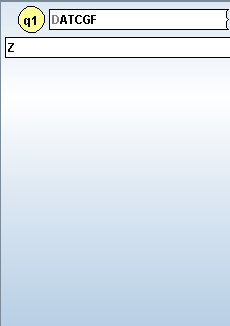
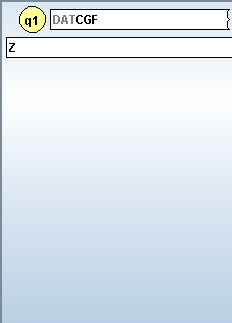
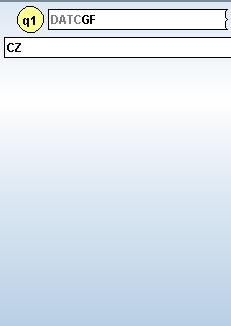
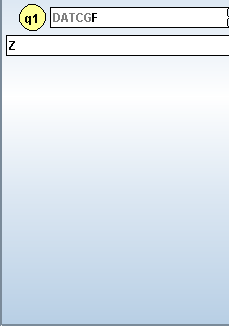
1. Initialization:
   1. Initialize stack with Z
   2. Switch to DNA processing
   3. Switch to RNA processing
2. DNA Processing in qdna:
   1. Push A onto the stack
   2. Push A onto the stack
   3. Pop A from the stack when T is encountered
   4. Push C onto the stack
   5. Push C onto the stack
   6. Pop C from the stack when G is encountered
   7. Push T onto the stack
   8. Push T onto the stack
   9. Pop T from the stack when A is encountered
   10. Push G onto the stack
   11. Push G onto the stack
   12. Pop G from the stack when C is encountered
   13. Accept if stack is empty and we reached the stack bottom marker
3. RNA Processing in qrna:
   1. Push A onto the stack
   2. Push A onto the stack
   3. Pop A from the stack when U is encountered
   4. Push C onto the stack
   5. Push C onto the stack
   6. Pop C from the stack when G is encountered
   7. Push U onto the stack
   8. Push U onto the stack
   9. Pop U from the stack when A is encountered
   10. Push G onto the stack
   11. Push G onto the stack
   12. Pop G from the stack when C is encountered
   13. Accept if stack is empty and we reached the stack bottom marker



AUTOMATION:





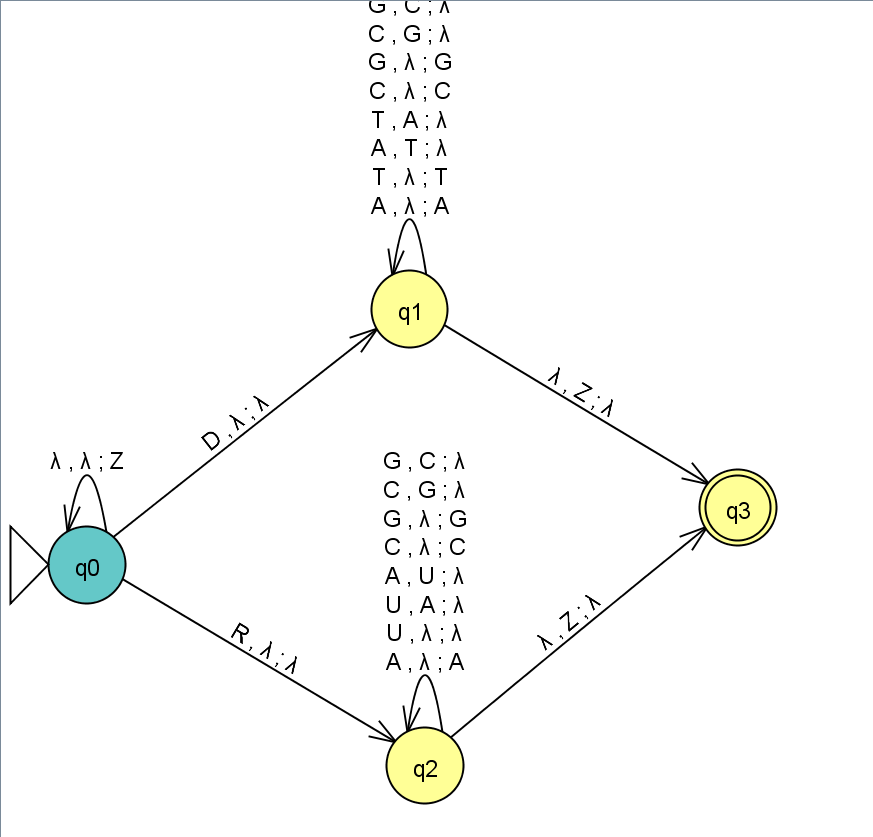
     

NPDA:

1. States
   1. q0 – Start State
   2. qdna – State for processing DNA Sequences
   3. drna – State for processing RNA Sequences
   4. qf – Accepting State
2. Alphabet
   1. Input Symbols – {D, R, A, T, C, G, U}
   2. Stack Symbols - {A, T, U, C, G, Z(Bottom Stack Marker)}
3. Transistion Functions
   1. Initialization:
      1. (q0, λ, λ) -> (q0,Z)
      2. (q0,D, λ) -> (qdna, λ)
      3. (q0,R, λ) -> (qrna, λ)
   2. DNA Processing in qdna:
      1. (qdna,A, λ) -> (qdna,A)
      2. (qdna,T, A) -> (qdna, λ)
      3. (qdna,T, λ) -> (qdna,T)
      4. (qdna,A, T) -> (qdna, λ)
      5. (qdna,C, λ) -> (qdna,C)
      6. (qdna,G, C) -> (qdna, λ)
      7. (qdna,G, λ) -> (qdna,G)
      8. (qdna,C, G) -> (qdna, λ)
      9. (qdna, λ, Z) -> (qf, λ)
   3. RNA Processing in qrna:
      1. (qrna,A, λ) -> (qrna,A)
      2. (qrna,U, A) -> (qrna, λ)
      3. (qrna,U, λ) -> (qrna,U)
      4. (qrna,A, U) -> (qrna, λ)
      5. (qrna,C, λ) -> (qrna,C)
      6. (qrna,G, C) -> (qrna, λ)
      7. (qrna,G, λ) -> (qrna,G)
      8. (qrna,C, G) -> (qrna, λ)
      9. (qrna, λ, Z) -> (qf, λ)

Transition Functions Explanation:

1. Initialization:
   1. Initialize stack with Z
   2. Switch to DNA processing
   3. Switch to RNA processing
2. DNA Processing in qdna:
   1. Push A onto the stack
   2. Pop A from the stack when T is encountered
   3. Push T onto the stack
   4. Pop T from the stack when A is encountered
   5. Push C onto the stack
   6. Pop C from the stack when G is encountered
   7. Push G onto the stack
   8. Pop G from the stack when C is encountered
   9. Accept if stack is empty and we reached the stack bottom marker
3. RNA Processing in qrna:
   1. Push A onto the stack
   2. Pop A from the stack when U is encountered
   3. Push U onto the stack
   4. Pop U from the stack when A is encountered
   5. Push C onto the stack
   6. Pop C from the stack when G is encountered
   7. Push G onto the stack
   8. Pop G from the stack when C is encountered
   9. Accept if stack is empty and we reached the stack bottom marker



Accepted Strings:

**DNA Examples:**

1. "DATCG"  
   **Explanation**:
   * Start: 𝑞0*q*0​, stack: []
   * Read 'D': 𝑞𝑑*qd*​, stack: [Z]
   * Read 'A': 𝑞𝑑*qd*​, stack: [A, Z]
   * Read 'G': 𝑞𝑑*qd*​, stack: [Z] (popped 'A')
   * Read 'T': 𝑞𝑑*qd*​, stack: [T, Z]
   * Read 'C': 𝑞𝑑*qd*​, stack: [C, T, Z]
   * End of input, stack is not empty, still in 𝑞𝑑*qd*​
2. "DTACG"
3. "DCATG"
4. "DGATC"
5. "DCTAG"

**RNA Examples:**

1. "RAUGC"
   * Start: 𝑞0*q*0​, stack: []
   * Read 'R': 𝑞𝑟*qr*​, stack: [Z]
   * Read 'A': 𝑞𝑟*qr*​, stack: [A, Z]
   * Read 'U': 𝑞𝑟*qr*​, stack: [Z] (popped 'A')
   * Read 'G': 𝑞𝑟*qr*​, stack: [G, Z]
   * Read 'C': 𝑞𝑟*qr*​, stack: [Z] (popped 'G')
   * End of input, transition to 𝑞𝑓*qf*​
2. "RACGU"
3. "RUACG"
4. "RGAUC"
5. "RUCGA”

Non Accepted Strings:

**DNA Examples:**

1. "DATG"  
   **Explanation**:
   * Start: 𝑞0*q*0​, stack: []
   * Read 'D': 𝑞𝑑*qd*​, stack: [Z]
   * Read '

continue

ChatGPT

Certainly! Here are the rest of the non-accepted strings:

**Non-Accepted Strings (continued)**

**DNA Examples:**

1. "DATG"  
   **Explanation**:
   * Start: 𝑞0*q*0​, stack: []
   * Read 'D': 𝑞𝑑*qd*​, stack: [Z]
   * Read 'A': 𝑞𝑑*qd*​, stack: [A, Z]
   * Read 'T': 𝑞𝑑*qd*​, stack: [Z] (popped 'A')
   * Read 'G': 𝑞𝑑*qd*​, stack: [G, Z]
   * End of input, stack is not empty, still in 𝑞𝑑*qd*​
2. "DACG":
3. "DAGT":
4. "DGCTA":
5. "DCTGA":

**RNA Examples:**

1. "RAUCG" **Explanation**:
   * Start: 𝑞0*q*0​, stack: []
   * Read 'R': 𝑞𝑟*qr*​, stack: [Z]
   * Read 'A': 𝑞𝑟*qr*​, stack: [A, Z]
   * Read 'U': 𝑞𝑟*qr*​, stack: [Z] (popped 'A')
   * Read 'C': 𝑞𝑟*qr*​, stack: [C, Z]
   * Read 'G': 𝑞𝑟*qr*​, stack: [Z] (popped 'C')
   * End of input, stack is not empty, still in 𝑞𝑟*qr*​
2. "RAUG":​
3. "RUCGA":
4. "RACG":
5. "RGUA"