Nicholas Skinner

CS321

Fall 2019

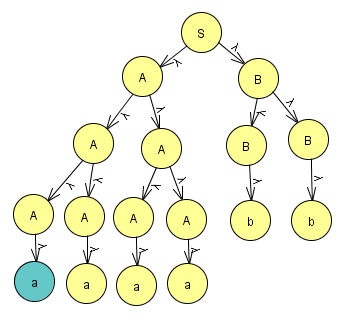
Homework 5

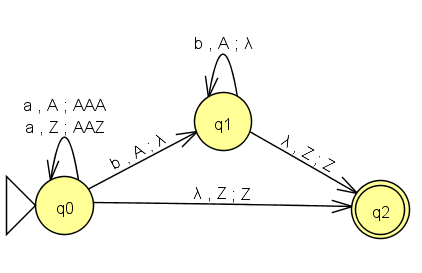
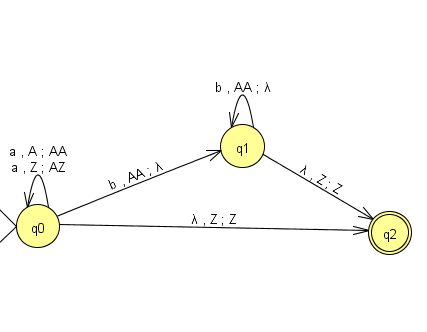
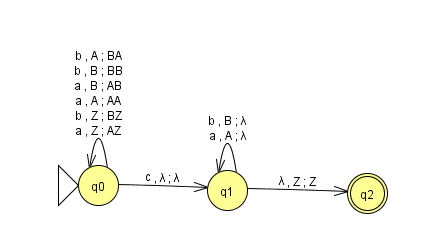
1. **Convert the grammar below to CNF**
   1. **We need to remove the nullable variables, remove the terminals from the variables, then modify any state with more than 2 variables, then we will want to look into single variables and modify them to comply with the form.**
   2. **Base**
      1. **S -> A | ABD | 0BB**
      2. **A -> 0 | BAA**
      3. **B -> BB | 1 | 2 | lambda**
      4. **C -> CD | 0**
      5. **D -> D1 | DD**
   3. **Remove nullables**
      1. **S -> A | ABD | 0BB | AD | 0**
      2. **A -> 0 | BAA | AA**
      3. **B -> 1 | 2**
      4. **C -> CD | 0D -> D1 | DD**
   4. **Remove terminals**
      1. **S -> 0 | BAA | AA | ABD | 0BB | AD | 0**
      2. **A -> 0 | BAA | AA**
      3. **B -> 1 | 2**
      4. **C -> CD | 0**
      5. **D -> D1 | DD**
   5. **Split variables/final state**
      1. **S -> 0 | BE | AA | AF | V1G | AD | 0 A → 0 | BE | AA**
      2. **B -> 1 | 2**
      3. **C -> CD | 0**
      4. **D -> D1 | DD**
      5. **E -> AA**
      6. **F -> BD**
      7. **G -> BB**
      8. **V1 -> 0**
2. **Consider the CNF grammar**
   1. **W1 = babbc**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **I\J** | **1** | **2** | **3** | **4** | **5** |
| **1** | **B,D** | **/** | **B** | **B** | **/** |
| **2** |  | **A** | **S,B** | **S,B** | **S** |
| **3** |  |  | **B,D** | **B,D** | **D,C** |
| **4** |  |  |  | **B,D** | **D,C** |
| **5** |  |  |  |  | **C,D** |

* + 1. **I don’t believe that w1 is in the language, as the top right value returns nothing**
  1. **W2 = aaaabb**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **i\j** | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | **A** | **A** | **A** | **A** | **B,S** | **B,S** |
| **2** |  | **A** | **A** | **A** | **B,S** | **B,S** |
| **3** |  |  | **A** | **A** | **S,B** | **B,S** |
| **4** |  |  |  | **A** | **B,D** | **B,S** |
| **5** |  |  |  |  | **B,D** | **B,D** |
| **6** |  |  |  |  |  | **B,D** |

* + 1. **I believe that this value is contained within the language, as it returns B and S**
    2. ****

1. **A**
   1. **L = { a^n b^2n : n >=0 }**
      1. **For each push, we will put two new A’s onto the stack**
      2. **We will pop one A for each b we receive**
      3. **If we consume the input, and the stack gives Z, then we will go to the final state, this should result in double the number of B’s as A’s**
         1. **M = (Q, Σ, Γ, δ, q0, z, F),**
            1. **Q = {q0, q1, q2}**
            2. **Σ = {a, b, c}**
            3. **Γ= {A, B, Z}**
            4. **q0 = {q0}**
            5. **F = {q2}**
            6. **δ is represented by the graph:**
         2. ****
   2. **L = { w : na(w) = 2nb(w) }**
      1. **For each new a, we will push an A onto the existing stack**
      2. **For each b we receive, we will want to pop two A’s from the stack**
      3. **If we consume the inputs and the stack gives Z, go to the final stack**
         1. **M = (Q, Σ, Γ, δ, q0, z, F),**
            1. **Q = {q0, q1, q2}**
            2. **Σ = {a, b, c}**
            3. **Γ= {A, B, Z}**
            4. **q0 = {q0}**
            5. **F = {q2}**
            6. **δ is represented by the graph:**
         2. ****
   3. **L = { wcw^(R) : w contains {a,b}\* }**
      1. **We will have our initial state that accepts any sequence of A’s and B’s**
      2. **We will use a c to transition from our first state to the next state, where we will now consume any A and B’s in our reverse of w**
      3. **If we receive a z from the base of the stack, we will transition to the final state.**
         1. **M = (Q, Σ, Γ, δ, q0, z, F),**
            1. **Q = {q0, q1, q2}**
            2. **Σ = {a, b, c}**
            3. **Γ= {A, B, Z}**
            4. **q0 = {q0}**
            5. **F = {q2}**
            6. **δ is represented by the graph:**
         2. ****