Closure of DCFL's under complementation

To show that the complement of DCFL is also a DCFL, we can use the approach used in the case of regular languages, but with some modifications. This is because in the case of regular languages, to show that the complement language is regular, we could interchange the final states and non-final states such that if string w is accepted in L, then it is rejected in L'. There are two difficulties with this approach.

1. The original DPDA never moves beyond some point on an input string w.

Reasons: This could happen, on reading the input w, if it reaches an ID where no move is possible or moves are defined with ε, which forces it to make infinite moves without considering the next input.

Problem: In either case, original DPDA does not accept any input with w as a prefix. If a DPDA to accept the complement language is defined by simply changing final and

non-final states, it would accept every string with prefix w without going beyond w. Hence, the resulting DPDA still would accept strings with w as prefix without processing the complete string.

2. After processing the sentence x, the DPDA may take several moves on ε-input.

Reason: The DPDA may be in final states after some of these moves and in non-final states after others.

Problem: By interchanging the final and non final states in DPDA for complement language, it would still accept x.

These difficulties can be overcome by forcing

1. The DPDA to enter an ID only after processing the entire input. This takes care of accepting or rejecting the input only after processing the complete input.
2. Modify the DPDA by adding the second component to the state to record whether a final state of the original DPDA is reached or not. This takes care of rejecting sentence x by DPDA constructed for complement language; it is accepted by original DPDA.

Theorem 3: The complement of a DCFL is a DCFL.

Theorem 3: The complement of a DCFL is a DCFL.

Proof: Let $M = (Q, \Sigma, \tau, \delta, q_0, Z_0, F)$ be a DPDA satisfying the first condition.

Let $M' = (Q', \Sigma, \tau, \delta', q'_0, Z_0, F')$ be a DPDA simulating M where

$Q' = \{[q, k] \mid q \in Q \text{ and } k = 1, 2 \text{ or } 3\}$

$F' = \{[q, 3] \mid q \in Q\}$ and

$$q'_0 = \begin{cases} [q_0, 1] & \text{if } q_0 \text{ is in } F \\ [q_0, 2] & \text{if } q_0 \text{ is not in } F \end{cases}$$

The purpose of k in [q, k] is to record, between true inputs, whether it has entered an accepting state.

- $k = 1 \Rightarrow$ M has entered an accepting state since the last true input.
- $k = 2 \Rightarrow$ M has not entered an accepting state since the last true input.
- M′ simulates the move of M and changes k to 1 or 2 when M reads a true input, k = 1, and irrespective of the new state, whether it is final or non-final.
- If k = 2, M′ first changes k to 3 and then simulates the move of M, changing k to 1 or 2, depending on whether the new state of M is or is not in F.
- For q and p in Q and a in Σ, we define δ′ as follows:
  - If $\delta(q, \varepsilon, Z) = (p, \gamma)$, then for k = 1 or 2, $\delta'([q, k], \varepsilon, Z) = ([p, k'], \gamma)$ where $k' = 1$ if $k = 1$ or if p is in F; otherwise $k' = 2$.
  - If $\delta(q, a, Z) = (p, \gamma)$, for a in Σ, then $\delta'([q, 2], \varepsilon, Z) = ([p, 3], Z)$ and $\delta'([q, 1], a, Z) = \delta'([q, 3], a, Z) = ([p, k], \gamma)$ where k = 1or 2 for p in F, or not in F, respectively.

We can show that L(M′) is the complement of L(M).

Suppose $a_1, a_2, a_3 \ldots a_n$ is in L(M), then M enters a final state on a_n. State of M′ while simulating this string will be 1 before it is possible for M′ to use a true input after a_n. Therefore, M′ does not accept, while a_n was the last true input used.

Suppose $a_1, a_2, a_3 \ldots a_n$ is not in L(M). M′ will some time after reading a_n, have no ε-moves to make and will have to use a true input symbol. The second component of M′

state is 2, since $a_1, a_2, a_3 \ldots a_n$ is not in L(M). M′ will accept before attempting to use a true input symbol.