APPENDICES

A PROOF OF PROPOSITION 3.2

Proof. Assuming $s = p_1 ... p_m$, we proceed by induction on two variables, m and n, representing respectively the length of the string s and the number of steps in the derivation $\alpha \Rightarrow^n p_1 ... p_m$.

- Base case (with $n = 0, m = 0$):
  In this case, $\alpha \Rightarrow^0 \varepsilon$ and $s = \alpha = \varepsilon$. By Definition 3.1.1, we also know that $y = x$ since $y \in \mathbb{C}_{G,D}(x, \varepsilon) = \{x\}$. We need to show that
  $$\forall x.\ (x \in \mathbb{C}_{G,D}(x, \varepsilon) \iff \varepsilon \in traces(paths(x, x)))$$
  This is straightforward since $\varepsilon \in traces(paths(x, x))$.
  Notice that when $m = 0$, we have to build derivations from the empty string. So, $m = 0 \implies n = 0$.
- Inductive step on m (with $n = 0$): In this case, we have that $s = \alpha$, so we must prove that
  $$\forall x, y, s.\ (y \in \mathbb{C}_{G,D}(x, s) \iff s \in traces(paths(x, y)))$$
  This follows by mathematical induction on m.
- Inductive step on n (with $m > 0$): We need to demonstrate that
  $$\begin{aligned}\forall x, y, \alpha.\ (y \in \mathbb{C}_{G,D}(x, \alpha) &\iff \exists\, p_1 ... p_m\ .\alpha \Rightarrow^n p_1 ... p_m \\ &\quad \wedge p_1 ... p_m \in traces(paths(x, y)))\end{aligned}$$
  for an arbitrary n.
  Since $n > 0$, we have that $\alpha = \alpha_1\ A\ \alpha_2$, where $A \in N$ and $\alpha_1, \alpha_2 \in (N \cup \Sigma)^*$. By Induction Hypothesis, we have that there exist vertices $v, w \in V$ and indexes k, j where $0 \le k \le j \le m$ such that:
  $$\begin{aligned} v \in \mathbb{C}_{G,D}(x, \alpha_1) &\iff \alpha_1 \Rightarrow^* p_1 ... p_k \\ &\quad \wedge p_1 ... p_k \in traces(paths(x, v)) \\ w \in \mathbb{C}_{G,D}(v, A) &\iff A \Rightarrow^* p_{k+1} ... p_j \\ &\quad \wedge p_{k+1} ... p_j \in traces(paths(v, w)) \\ y \in \mathbb{C}_{G,D}(w, \alpha_2) &\iff \alpha_2 \Rightarrow^* p_{j+1} ... p_m \\ &\quad \wedge p_{j+1} ... p_m \in traces(paths(w, y))\end{aligned}$$
  These hypotheses, together with Definition 3.1.4 allow us to conclude the proof.

□

B PROOF OF PROPOSITION 3.6

Proof (Sketch). We analyze the behaviour of the algorithm at the lines that change the set I of trace items:

(line 2) The set I is initialized to contain the item $[\ A \to \{w^\circ\}\ \alpha_1\ \{\ \}\ ...\ \alpha_n\ \{\ \}\ ]$, for each rule $A \to \alpha_1 ... \alpha_n \in P$. From this construction we can see that for $j = 0$, we have that $w = x$, $C_0 = \{x\} = \{w\}$ and $\alpha_1 ... \alpha_j = \varepsilon$. In this case, it is evident that
$$w \in C_0 \iff w \in \mathbb{C}_{G,D}(w, \varepsilon).$$

(line 10) At this line, new trace items are added into the set I for each rule $\alpha_k \to \beta_1 ... \beta_n$. The creation of new items is in under the same conditions presented at line 2. Again $j = 0$, so we have $w = x$, $C_0 = \{x\} = \{w\}$ and $\beta_1 ... \beta_j = \varepsilon$. In this case, we have
$$w \in C_0 \iff w \in \mathbb{C}_{G,D}(w, \varepsilon).$$

(line 8) A position set C in I is incremented with new vertices y such that $(x, \alpha_k, y) \in D'$. We can distinguish two cases:

- If α_k is a terminal symbol, we add to C_k all vertices y such that exists a α_k-labeled edge from x to y in D':
  $$y \in C_k \iff y \in \mathbb{C}_{G,D}(x, \alpha_k).$$
  This condition holds by Definition 3.1.2.
- If $\alpha_k \in N$ we need to add to C_k all the vertices y such that there is an edge labelled (x, α_k, y) in D'. Notice that this edge was the result of a previous processing, meaning that the algorithm has already discovered a path from x to y such that its trace corresponds to the right-hand side of a production rule of α_k. Thus,
  $$y \in C_k \iff y \in \mathbb{C}_{G,D}(x, \alpha_k).$$
  This condition holds by Definition 3.1.3.

(line 14) We deal with those vertices x appearing at the last position set of a trace item $[\ A \to \{w^\bullet\} ... \{x^\circ, ...\}\ ]$ built from a production rule $A \to \gamma$. Items with this configuration indicate the existence of a path from w to x in D' such that its trace is the string γ. Our algorithm adds a new A-labeled edge from w to x (line 12), thus using the production rule. Thus, for every item $i = [\ B \to ... \{w^\bullet, ...\}\ A\ C_j ...\ ]$ built from a production rule $B \to \gamma_1\ A\ \gamma_2$, we can verify that:
$$x \in C_j \iff x \in \mathbb{C}_{G,D}(w, A).$$
This condition holds by Definitions 3.1.3 and 3.1.4.

□

C PROOF OF PROPOSITION 3.8

Proof. The maximum size that D' and I may reach is:

D': The algorithm increments the graph D' with non-terminal-labeled edges, so it uses at most:
$$|D'| \quad = \quad |V| \cdot |N \cup \Sigma| \cdot |V| \tag{7}$$
what is $\mathcal{O}(|V|^2 \cdot |N \cup \Sigma|)$.

I: The set I contains generalized items, which are annotated production rules with a single vertex at the start of the right-hand side. So we have at most:
$$|I| \quad = \quad |V| \cdot |P| \tag{8}$$
For each trace item, the number of position set sets depends on the size of the right-hand side of a production rule. Assuming that k denotes the greatest size of the right-hand side of the rules in P, each trace item may have k position sets of size at most $|V|$ (notice that the first position set on each trace item is always a singleton).
In this context, the worst case in space complexity for I is:
$$|V| \cdot |P| \cdot k \cdot |V|.$$
what is $\mathcal{O}(|V|^2 \cdot |P| \cdot k)$.

We can now estimate the worst-case space complexity as:

$$\mathcal{O}(|V|^2 \cdot (|N \cup \Sigma| + |P| \cdot k)) \tag{9}$$

□

D PROOF OF PROPOSITION 3.9

Proof (Sketch). The main loop iterates until there are no more unmarked vertices x°. The maximum number of unmarked vertices is given by $|I| \cdot k \cdot |V|$, where k is the maximum number of possible position sets for rules of the grammar (the greatest size of a right-hand side of the rules in P, plus one). So, as $|I| = |V| \cdot |P|$, we have at most $|V|^2 \cdot |P| \cdot k$ possible vertices x°.

For each iteration, the form of the trace item i guides the operation to be performed. The tests at lines 6 and 11 have constant cost.

There are two cases to be considered inside the **switch** command:

- The evaluation of the condition at line 7 requires searching over the set of trace items I. The cost of this operation is constant (supposing that we use a matrix representation). Line 8 is the case where the algorithm advances one step on a path by looking for edges $(x, \alpha, y) \in D'$. As there are at most $|V|$ possible destination vertexes, the algorithm performs at most $|V|$ operations in this case.
  At line 10, the algorithm adds new trace items to I in order to start a new derivation. This line ensures that the algorithm only creates at most one trace item for each production rule in P for a fixed vertex x. So, in this case, the algorithm performs at most $|P|$ constant time operations.
  In this way, the overall cost of the case spanning from line 6 to 10 is bounded by $\max(|V|, |P|)$.
- The second case of the *switch* command adds non-terminal labelled edges to the graph. The creation of such edges is performed at line 12, in constant time.
  The appearance of a new edge triggers the update of position sets by the iteration at line 13. We have at most $|V| \cdot |P| \cdot k$ position sets. Assuming, again, a matrix representation, locating each set C in a trace item, requires constant time. Thus, line 14 will be executed $|V| \cdot |P| \cdot k$ times in the worst case.
  In this way, the overall cost of the case spanning from line 11 to 14 is bounded by $|V| \cdot |P| \cdot k$.

This shows that the worst-case time complexity of our algorithm is $\mathcal{O}(|V|^3 \cdot |P|^2 \cdot k^2)$.

□