Explicit Loops Insertion Formalisation

Quentin Nivon, Gwen Salaün

September 16, 2025

To appear in G, each loop L appearing in the expressions must form a strongly connected component in G. Depending on the composition of the loop, the insertion is performed differently. We can differentiate two major cases: either (i) none of the nodes of the loop belong to G, i.e., $L \cap V = \emptyset$ or (ii) at least one node of the loop belongs to G, i.e., $L \cap V \neq \emptyset$. By definition, the nodes $v \in L \setminus V$ cannot be mutually exclusive of any other node, otherwise they would have been added to G in the previous step, nor sequentially constrained to any other node, otherwise they would already belong to G. Thus, they are not constrained with regards to any other node of G.

In case (i), the approach simply consists in connecting all the $v \in L$ together, to form a strongly connected component. Then, one of these nodes is marked as initial node of the graph.

In case (ii), some $v \in L$ already belong to G, while some others do not. However, the loop nodes belonging to G may be completely disconnected, already connected, or partially connected, thus no assumption can be made on how to connect them. If they do not form a strongly connected component yet, they have to be connected to make this strongly connected component appear in G. This starts by computing all the components of G consisting only of nodes of L.

These components represent disconnected portions of L that one has to connect to make the strongly connected component corresponding to L appear in G. However, this must be done carefully in order not to break any mutual exclusion already handled by G. Indeed, connecting two such components consists in adding new flows to G, which has an impact on its paths, and thus, potentially, on its mutual exclusions. To ensure that L will be added to G if it is possible (i.e., if it does not intrinsically break some mutual exclusions), all permutations of the components are computed. Each such permutation represents a possible order in which the components can be connected to the others to make the loop appear in G. For each such permutation, the n^{th} component is connected to the $n+1^{\text{th}}$ component. This is done by connecting each node of the n^{th} component having a *0-reachability* to the set of nodes of the $n+1^{\text{th}}$ component ensuring an ∞-*reachability*.

Definition 1 (n-reachability of a Node)**.** *Let $G = (V,\ E,\ \Sigma)$ be a BPMN process. $\forall v \in V$, the* n-reachability *of v is the number of nodes that v can reach, i.e.,*

$$n = |\{v' \in V \setminus \{v\} \mid v \xrightarrow{R} v'\}|$$

By convention, if $n = |V| - 1$ (i.e., if v can reach all the nodes of G), v is said to have an ∞-reachability.

This notion can be extended to a set of nodes.

Definition 2 (n-reachability of a Set of Nodes)**.** *Let $G = (V,\ E,\ \Sigma)$ be a BPMN process. $\forall \{v_1, ..., v_m\} \subseteq V$, the* n-reachability *of $\{v_1, ..., v_m\}$ is the number of nodes that $\{v_1, ..., v_m\}$ can reach, i.e.,*

$$n = |\bigcup_{v_i \in \{v_1,...,v_m\}} \{v' \in V \setminus \{v_1, ..., v_m\} \mid v_i \xrightarrow{R} v'\}|$$

By convention, if $n = |V| - m$ (i.e., if the $\{v_1, ..., v_m\}$ can reach all the nodes of G), the set $\{v_1, ..., v_m\}$ is said to have an ∞-reachability.

Such a connection ensures that each node of the n^{th} component can now reach every node of the $n+1^{\text{th}}$ component, and also that the n^{th} and $n+1^{\text{th}}$ components now form a single component. If during the connection phase, the n^{th} component of a permutation cannot be connected to the $n+1^{\text{th}}$ component without breaking some existing mutual exclusions, the permutation is discarded. Once a valid permutation is found, the remaining ones are discarded. If no valid permutation is found, the explicit loop L is not added to G. Finally, if some tasks of the loop did not already belong to G, they are arbitrarily added between two connected components, using the same method than in case (i).

Proposition 1 (Validity of the Components Connection)**.** *Let $G = (V,\ E,\ \Sigma)$ be a BPMN process, let $L = (v_1, ..., v_n) \in \texttt{Loops}$ be a loop that should be added to G, and let $\{G_1, ..., G_m\}$ be the set of components of G consisting of tasks of the loop only. We state that $\forall i \in [1...m-1]$, connecting all the $\{v_1, ..., v_o\} \in G_i$ having a 0-reachability to the (smallest) set of $\{v_1, ..., v_p\} \in G_{i+1}$ ensuring an ∞-rechability, and all the $\{v_1, ..., v_q\} \in G_m$ having a 0-reachability to the (smallest) set of $\{v_1, ..., v_r\} \in G_1$ ensuring an ∞-rechability make L become a strongly connected component in G.*

Proof. Let $G = (V,\ E,\ \Sigma)$ be a BPMN process, let $L = (v_1, ..., v_n) \in \texttt{Loops}$ be a loop that should be added to G, and let $\{G_1, ..., G_m\}$ be the set of components of G consisting of nodes of the loop only. Let us separate the proof into two parts.

First, let us show that connecting all the nodes of a component having a 0-reachability to the (smallest) set of nodes having an ∞-reachability makes the component become a strongly connected component. Let $\{v_a, ..., v_m\}$ be the set of nodes of G_1 having a 0-reachability, and let $\{v_n, ..., v_z\}$ be its (smallest) set of nodes ensuring an ∞-reachability. Adding an edge connecting each $v_i \in \{v_a, ..., v_m\}$ to each $v_j \in \{v_n, ..., v_z\}$ ensures that each $v_i \in \{v_a, ..., v_m\}$ now has a ∞-reachability. Moreover, by definition, each $v_k \notin \{v_a, ..., v_m\} \cup \{v_n, ..., v_z\}$ must have at least a 1-reachability (otherwise it would belong to the $\{v_a, ..., v_m\}$). Hence, it must be able to reach at least one $v_i \in \{v_a, ..., v_m\}$. However, we know that each $v_i \in \{v_a, ..., v_m\}$ now has an ∞-reachability. Thus, each $v_k \notin \{v_a, ..., v_m\} \cup \{v_n, ..., v_z\}$ now has an ∞-reachability. As each $v \in V$ now has a an ∞-reachability, G is a strongly connected component.

Then, let us show that connecting two components G_1 and G_2 with the same method creates another component $G_{1,2}$. Let $\{v_{1_a}, ..., v_{1_m}\}$ be the set of nodes of G_1 having a 0-reachability, let $\{v_{1_n}, ..., v_{1_z}\}$ be its (smallest) set of nodes ensuring an ∞-reachability, let $\{v_{2_a}, ..., v_{2_m}\}$ be the set of nodes of G_2 having a 0-reachability, and let $\{v_{2_n}, ..., v_{2_z}\}$ be its (smallest) set of nodes ensuring an ∞-reachability. Adding an edge connecting each $v_i \in \{v_{1_a}, ..., v_{1_m}\}$ to each $v_l \in \{v_{2_n}, ..., v_{2_z}\}$ ensures that $G_{1,2} = (V_1 \cup V_2, E_1 \cup E_2 \cup \{v_i \to v_l \mid v_i \in \{v_{1_a}, ..., v_{1_m}\} \wedge v_l \in \{v_{2_n}, ..., v_{2_z}\}\}, \Sigma_1 \cup \Sigma_2)$ is a component. Moreover, it also ensures that the $\{v_{1_n}, ..., v_{1_z}\}$ have an ∞-reachability on this component. By construction, we also have that the $\{v_{2_a}, ..., v_{2_m}\}$ still have a 0-reachability. Thus, we created a component $G_{1,2}$ that contains both G_1 and G_2, and which has the same (smallest) set of ∞-reachability nodes than G_1, and the same set of 0-reachability nodes than G_2.

We showed that $\forall i \in [1...m-1]$, connecting each G_i to each G_{i+1} creates a component G_f containing all the G_i, and whose (smallest) set of ∞-reachability nodes is the same than G_1, while its set of 0-reachability nodes is the same than G_m. Finally, we showed that connecting this set of 0-reachability nodes to the set of ∞-reachability nodes makes G_f a strongly connected component, which is our goal. $\square$