Analysing Message Sequence Graph Specifications*

Joy Chakraborty
Motorola India Private Limited
C. V. Raman Nagar
Bangalore 560093, India.
j.chakraborty@motorola.com

Deepak D'Souza
Computer Science and Automation
Indian Institute of Science
Bangalore 560012, India.
deepakd@csa.iisc.ernet.in

K. Narayan Kumar
Chennai Mathematical Institute
H1 SIPCOT IT Park
Siruseri 603103, India.
kumar@cmi.ac.in

March 31, 2009

Abstract

We give a detailed construction of a finite-state transition system for a com-connected Message Sequence Graph. Though this result is fairly well-known in the literature there has been no precise description of such a transition system. Several analysis and verification problems concerning MSG specifications can be solved using this transition system. The transition system can be used to construct correct tools for problems like model-checking and detecting implied scenarios in MSG specifications. The transition system we give can also be used for a bounded analysis of general (not necessarily com-connected) MSG specifications.

*Technical Report IISc-CSA-2009-1, Department of Computer Science and Automation, Indian Institute of Science, Bangalore.

1 Introduction

Message Sequence Chart (MSC) based specifications are a popular model of early system design, whose use is particularly widespread in the telecom and software industry. A message sequence chart describes a finite sequence, or more accurately a partially ordered sequence, of message exchanges between agents in the system. These are typically "scenarios" that a system user and developer alike can use to communicate and validate system requirements. Messages may be exchanged "synchronously" as in a handshake protocol, or "asynchronously" with separate send and receive events and a message channel to buffer undelivered messages. Message Sequence Graphs (MSG's), also sometimes referred to as "high-level" MSC's, are an activity diagram-like notation that is often used to describe infinite collections of system behaviour. They are finite graphs whose vertices are labeled by MSC's, each of which represents a single logical unit of interaction. The behaviours specified by an MSG are obtained by taking a path in the MSG beginning at the initial node, and collecting the behaviours given by the "concatenation" of the MSC's associated with the nodes along the path.

Given that MSC-based specifications provide an early encapsulation of system design, from an analysis and verification point of view there are some natural problems that one would like to address. Several of these have been considered in the literature, including detecting race conditions (differences in the "visual" ordering and "execution" ordering), timing conflicts, and confluence or "completability." We would like to focus on the following two problems:

1. The model-checking problem [2]: Here we are given a system description in terms of an MSG, and a property in the form of a finite-state automaton describing say undesirable behaviours. We would like to check that the system does not exhibit any of the undesirable behaviours.

2. Detecting implied scenarios [14, 1]: Given a description of system behaviour in terms of an MSG, there is a natural, distributed, system model induced by the MSG. This system model is "minimal" in that any distributed implementation of the system that exhibits all the behaviours specified by the given MSG, must necessarily exhibit all the behaviours in the system model induced by the MSG. However, the minimal system model may exhibit behaviours that are outside the ones specified by the MSG: these behaviours are called *implied scenarios*. We are interested in identifying such behaviours so that the system designer can be alerted (for example to the fact that the exact behaviour specified by the MSG is not realizable by a distributed implementation).

Message Sequence Chart based specifications have received a fair amount of attention from the Computer Science theory community (see [4, 5] for surveys). In particular the analysis problems mentioned above have been addressed in the following works. Alur and Yannakakis [2] show that the model-checking problem for asynchronous MSG's is undecidable in general. They propose a condition on the MSG, called "boundedness" or "com-connectedness" (essentially that all processes that take part in any loop of the MSG must communicate directly or indirectly with each other in the loop), which is sufficient to ensure that the model-checking problem is decidable. The main task is to show that in such

a case the language of behaviours defined by the MSG is regular, i.e. acceptable by a finite-state transition system. However the details of the construction are not spelt out completely, and there is no proof of correctness given. Independently in [8], Muscholl and Peled also give several undecidability results for asynchronous MSG's, including that checking for race conditions and confluence in asynchronous MSG's is undecidable. They also restrict the class of MSG's considered to com-connected ones (called "loop-connected" there), to obtain decision procedures for the race condition and confluence problems. In this connection they state the result that a com-connected MSG defines a regular language. For a proof they point to an earlier line of work in trace theory which give various sufficient conditions under which a regular language remains regular when closed under "partial commutations". In particular they make use of a result of Clerbout and Latteux [3] which gives such a condition, analogous to com-connectedness. The construction and proof in [3] is in terms of grammars and it is not easy to directly associate a transition system with a given MSG using it.

The software engineering community parallely have developed several tools and methodologies for analysing MSC-based specifications. However some of these works are based on an incorrect understanding of some of the results in the literature, in particular the result claiming regularity of com-connected MSG's. In Section 8 we point out some of these cases, without detracting from the several other contributions made in these papers.

- In [14], Uchitel, Kramer, and Magee claim to solve the problem of detecting implied scenarios for general (not necessarily com-connected) MSG's with synchronous messaging. This is done without explicitly building a transition system for the given MSG. No complete proofs are given in the paper or the cited technical report. This claim cannot be correct as it is not difficult to show that the problem is in fact undecidable (i.e for general synchronous MSG's). We give a proof of this fact in Section 8.

- In [7] Muccini gives a technique for detecting implied scenarios based on identifying "augmented" behaviours in the components of the system model for a given general synchronous MSG. The technique is validated on a few examples, but the paper gives no proofs and admits that the "correctness and completeness are still under analysis."

- The thesis of Uchitel [12] gives the construction of a finite-state transition system, called there the "trace model", for a given com-connected synchronous MSG. This construction is implemented in a tool called LTSA-MSC [11]. However, as we show in Section 7, the trace-model constructed is incomplete: it does *not* accept all behaviours specified by the MSG. As a result the tool also incorrectly flags certain behaviours of the induced system model as implied scenarios.

Our aim in this paper is to give a precise and complete description of a finite-state transition system accepting exactly the behaviours specified by a given com-connected MSG G. We give a precise description of a "reduced" transition system called $\mathcal{T}'_G$ in Section 5 which is guaranteed to be finite-state when the given MSG is com-connected. We give a carefully checked proof of correctness of our construction. Once we have such a transition system, the

analysis problems we mentioned earlier can be solved easily for com-connected MSG's with synchronous messages.

It is also worth pointing out that the transition system $\mathcal{T}'_G$ we describe is sound and complete for *general* MSG's (i.e. not necessarily com-connected) as well – though of course, it may not be finite-state in this case. Our construction also handles both synchronous and asynchronous messaging in the MSG's. Further, it has no ϵ-transitions (i.e. hidden or silent transitions), and has a bounded number of transitions applicable in any state. Thus, this transition system can be used to perform a bounded analysis or model-checking to check properties like "there are no property violations by behaviours of length ≤ 15 in the given MSG model," or that "there are no implied scenarios of length ≤ 15 in the given MSG model."

We thus hope that the construction we give will be a basis on which the software engineering community can build more accurate tools for analysing MSC-based specifications.

2 Message Sequence Charts

We begin with some preliminary notions. For a finite alphabet A, we denote the set of finite words over A by A^*. The empty word is denoted ϵ. For words u and v over A, we denote the concatenation of u followed by v by $u \cdot v$ or simply uv. We write $u \preceq v$ (or $u \preceq_{ue} v$) to denote the fact that u is a prefix (or non-empty prefix respectively) of v. Also, $u \prec v$ denotes that $u \cdot \alpha = v$ for some non-empty α.

A *transition system* is a tuple $\mathcal{T} = (Q, A, q_0, \rightarrow)$ where:

- Q is a set of states
- A is the set of labels or alphabet of the transition system
- q_0 is the initial state
- $\rightarrow \subseteq Q \times A \times Q$ is the labeled transition relation.

We define a *run* from q_1 to q_2 on a word $w \in A^*$ denoted by $q_1 \xrightarrow{w}{}^* q_2$ by induction on length of w as follows: $q_1 \xrightarrow{\epsilon}{}^* q_1$, and $q_1 \xrightarrow{wa}{}^* q_2$ if there exists $q_3 \in Q$ such that $q_1 \xrightarrow{w}{}^* q_3$ and $q_3 \xrightarrow{a} q_2$. The language generated by $\mathcal{T}$, denoted $L(\mathcal{T})$, is defined to be $\{w \in A^* \mid q_0 \xrightarrow{w}{}^* q \text{ for some } q \in Q\}$. For a state $q \in Q$, we represent the language generated by $\mathcal{T}$ starting at q as $L_q(T)$. We define this to be

$$\{w \in A^* \mid q \xrightarrow{w}{}^* r \text{ for some } r \in Q\}.$$

A *message sequence chart* or *MSC* is a tuple

$$M = \langle P, E, C, \lambda, B, \{<_p\}_{p \in P} \rangle$$

where:

- P is a finite set of processes.
- E is a finite set of events.

- C is a finite set of message labels.

  The set of actions of M is defined to be $\Sigma_M = P \times \{!, ?\} \times P \times C$ where $(p, !, q, m)$ signifies process p sending message m to q, while action $(p, ?, q, m)$ signifies p receiving message m from q. All actions of the form $(p, !, q, m)$ are called *send* actions, while actions of the form $(p, ?, q, m)$ are called *receive* actions.

  For a process $p \in P$, the set $\Sigma_p = \{a \in \Sigma_M \mid a = (p, !, q, m) \text{ or } a = (p, ?, q, m)\}$ where $q \in P$ and $m \in C$, is the set of all actions in which p participates.

- $\lambda : E \to \Sigma_M$ is the labeling function which maps events to actions. For a process $p \in P$, the set $E_p = \{e \mid \lambda(e) \in \Sigma_p\}$ are the events in which p participates.

  E is further partitioned into send events $S = \{e \,|\, \lambda(e) = (p, !, m, q),\ p, q \in P,\ m \in C\}$ and receive events $R = \{e \,|\, \lambda(e) = (p, ?, m, q),\ p, q \in P,\ m \in C\}$ respectively.

- $B : S \to R$ is a bijective map which maps each send event to its corresponding receive event. We require that if $\lambda(e) = (p, !, q, m)$ then $\lambda(B(e)) = (q, ?, p, m)$. We refer to B as the "matching receive" map.

- For each $p \in P$, $<_p$ is a strict total order on E_p. In addition, the matching receive map B induces a strict partial order $<_B$ on E, which says that a receive event has to be preceded by the corresponding send event, defined by $e <_B e'$ iff $B(e) = e'$.

  We define $<_M$ as the transitive closure, and $\leq_M$ as the reflexive transitive closure, of $(\bigcup_{p \in P} <_p) \cup <_B$ respectively. It is required that the relation $\leq_M$ must be a partial-order.

A *linearization* of the events in an MSC M as defined above is a sequence of events $w = e_1 e_2 \cdots e_n \in E^*$ containing all the events of E without repetitions, and respecting the partial order $\leq_M$ in the sense that for no $i < j \leq n$ do we have $e_j \leq_M e_i$. We denote the set of linearizations of M by $lin(M)$. We define the event language of M, written $L^e(M)$, to be the set of all prefixes of linearizations of M. Thus, $L^e(M) = \{x \mid x \preceq y,\ y \in lin(M)\}$.

Following [2], we define a *cut* in an MSC M to be a subset c of the events E of M which is closed with respect to the partial order $\leq_M$: i.e. if $e \in c$ and $e' \leq_M e$, then $e' \in c$. Each prefix of a linearization of an MSC corresponds to a sequence of "incremental" cuts as described below:

Lemma 1 *Let M be an MSC with event set E. Then $w \in E^*$ is a prefix of some linearization of M if and only if there exists a sequence of cuts $\{c_x\}_{x \preceq w}$ in M such that*

- $c_\epsilon = \emptyset$
- *For each $x \cdot e \preceq w$, we have $c_{x \cdot e} - c_x = \{e\}$.*

Proof Assume w is a prefix of some linearization of M. Consider $x \preceq w$. Let c_x be the set of all events in x. By our observation about prefix of linearization, the

Figure 1: An example MSC and cut $\{e_1, e_3, e_4\}$ in it shown by shaded events.

set c_x is downward closed with respect to $\leq_M$. So, c_x is a cut in M. Trivially, $\forall x \cdot e \preceq w, c_{x \cdot e} - c_x = \{e\}$ and $c_\epsilon = \emptyset$. This proves the forward direction.

Now, consider the sequence of events $w = e_1 \cdot e_2 \cdots e_n$ such that there exists a sequence of cuts $(c_x)_{x \preceq w}$ such that $c_\epsilon = \emptyset$ and $\forall x \cdot e \preceq w, c_{x \cdot e} - c_x = \{e\}$. We will prove that w is a prefix of linearization of M by showing that the claim is true for all $x \preceq w$.

We claim that $\forall x \preceq w$, cut c_x is the set of events of x. This is trivially true for $x = \epsilon$. Let this be true for some $x \preceq w$. Consider $x \cdot e \preceq w$. Since, c_x is the set of events for x, and $c_{x \cdot e} = c_x \cup \{e\}$ so, $c_{x \cdot e}$ is the set of events for $x \cdot e$. This proves that c_x is the set of events of x. Since cut c_x is downward closed with respect to M so set of events of x is downward closed.

We will now show that for no $i < j \leq n$ do we have $e_j \leq_M e_i$. This is trivially true for $x = e_1$. Let this be true for some $x = e_1 \cdot e_2 \cdots e_j$ where $j < n$. So, for no $i < j$ do we have $e_j \leq_M e_i$. Now consider $x \cdot e_{j+1} \preceq w$. $e_{j+1} \leq_M e_j$ as otherwise, $e_{j+1} \in c_x$ which means e_{j+1} should have appeared in x. This proves our claim.

This proves that both the observations about prefix of linearization is satisfied by w. □

The Figure 1 shows an example MSC on the left, with set of events $E = \{e_1, e_2, e_3, e_4\}$, and $<_p = \{(e_1, e_3)\}$, $<_q = <_r = \emptyset$. Thus the strict partial order $<_M$ is given by $\{(e_1, e_3), (e_1, e_2), (e_3, e_4), (e_1, e_4)\}$. The figure on the right shows a cut $c = \{e_1, e_3, e_4\}$ in the MSC.

We now define the notion of a message sequence graph. A *Message Sequence Graph* (MSG) is a vertex-labeled graph G of the form $\langle V, v_0, \Delta, \mathcal{M}, \mu \rangle$, where V is the set of vertices of the MSG, $v_0 \in V$ is the initial vertex, $\Delta \subseteq (V \times V)$ is the set of directed arcs, $\mathcal{M}$ is the set of MSC's associated with the MSG, and $\mu : V \to \mathcal{M}$ maps each vertex of G to one of the MSCs in $\mathcal{M}$. We assume that the MSC's in $\mathcal{M}$ are all over a common set of processes and labels, none of the MSC's are empty, and also that the events across the MSC's in $\mathcal{M}$ are distinct. The set of events of G is denoted E^G, and is defined to be $\bigcup_{v \in V} E_{\mu(v)}$. We denote the set of events where process p participates by E^G_p, defined in a similar way as for a single MSC. The set of action labels for G is defined to be $\Sigma_G = \bigcup_{v \in V} \Sigma_{\mu(v)}$. Figure 2 shows an example MSG.

A *path* in G is a sequence of vertices $v_1, \ldots, v_k$ $(k \geq 0)$ of G such that $(v_i, v_{i+1}) \in \Delta$ for each $i \in \{1, \ldots, k-1\}$. An *initial path* in G is a path beginning at v_0. We will use the convention that α, β, etc. denote paths in G, and u, v etc. denote vertices in G.

Let π be a non-empty path in G. For each vertex v in G, let each MSC $\mu(v)$ be $\langle P_v, E_v, C, \lambda_v, B_v, \{<^v_p\}_{p \in P} \rangle$. We define the *(weak) concatenation* of

the MSCs in the path π to be the MSC $M_\pi = \langle P, E_\pi, C, \lambda_\pi, B_\pi, \{<_p^\pi\}_{p\in P}\rangle$ where:

- $E_\pi = \bigcup_{\rho v \preceq \pi}(E_v \times \{\rho v\})$.
- For each $\rho v \preceq \pi$, we define $\lambda_\pi(e, \rho v) = \lambda_v(e)$.
- For each $\rho v \preceq \pi$, and for all send events $e \in E_v$, we define $B_\pi(e, \rho v) = (B_v(e), \rho v)$.
- For each $p \in P$, $<_p^\pi$ is given as follows: Let $\rho v \preceq \pi$ and $\rho' v' \preceq \pi$ and $e \in E_v$ and $e' \in E_{v'}$. Then $(e, \rho v) <_p^\pi (e', \rho' v')$ iff e and e' are p-events and either $\rho v \prec \rho' v'$ or $\rho v = \rho' v'$ and $e <_p^v e'$.

The set of linearizations of G, denoted by $lin(G)$, is defined to be

$$\{e_1 \cdots e_n \mid \pi \text{ initial path in } G, (e_1, \rho_1) \cdots (e_n, \rho_n) \in lin(M_\pi)\}.$$

The *event language* of G, denoted $L^e(G)$, is defined to be $\{u \mid u \preceq v \text{ and } v \in lin(G)\}$. The *action language* of G is denoted $L^a(G)$, and defined to be

$$\{\lambda_{v_1}(e_1) \cdots \lambda_{v_n}(e_n) \mid e_1 \cdots e_n \in lin(G), \text{ each } e_i \in E_{v_i}\}.$$

We give one final definition in this section. Consider an MSC $M = \langle P, E, C, \lambda, B, \{<_p\}_{p\in P}\rangle$.

The *communication graph* of M is the directed graph on the set of process P of M, where we have an edge (p, q) iff process p sends a message to process q in M (i.e. there exists an event $e \in E_p$ and an event e' in E_q, with $B(e) = e'$). The MSC M is said to be *com-connected* [2] if the communication graph of M contains at most one non-trivial strongly connected component and all strongly connected components are isolated (i.e. there are no edges between them). We say an MSG G is *com-connected* if for every loop, i.e. a path of the form $u\alpha u$, in G, the communication graph of $M_{u\alpha}$ is com-connected. We illustrate this using the MSG G_1 in Figure 2. The communication graph for the loop $v_0v_1v_0$ is comm-connected, whereas the loop $v_0v_2v_0$ is not com-connected. Thus the MSG G_1 is not com-connected.

Our aim in this paper is to provide a constructive proof of the following claim:

Theorem 2 ([2, 8]) *Let G be a com-connected MSG. Then $L^e(G)$ is regular (i.e. it can be generated by a finite-state transition system).*

We will prove this claim in Section 6. We first show in the next section how for a general (not necessarily com-connected) MSG G we can associate a transition system $\mathcal{T}_G$ which generates precisely the language $L^e(G)$, but which may have an infinite number of states. In the following section we give a couple of rules by which we can reduce the state space of $\mathcal{T}_G$, while preserving its language, to obtain a transition system $\mathcal{T}'_G$. Finally in Section 6 we show that when the given MSG is com-connected, this reduced transition system $\mathcal{T}'_G$ will indeed have a finite number of states.

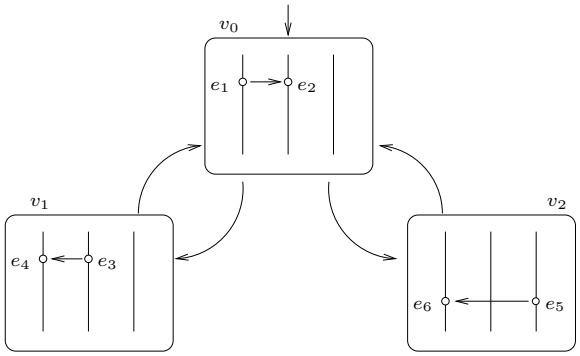


Figure 2: An example MSG G_1

3 Transition system for an MSG

In this section we show how we can associate a transition system $\mathcal{T}_G$ with a given *general* MSG G, which generates exactly the same event language as G. This transition system will, in general, have an infinite number of states.

In the sequel we fix an MSG $G = \langle V, v_0, \Delta, \mathcal{M}, \mu \rangle$.

We begin with some preliminary notions.

We define a *configuration* of G to be a pair of the form (π, c) where π is a path in G and c is a cut in M_π. Configurations will play the role of states in $\mathcal{T}_G$. We can view a configuration (π, c) as a snapshot of the events each process has completed in its process line in the MSC M_π. Each process p can be viewed to be positioned at last event it has performed.

Next we introduce some notation regarding insertion and deletion in paths and cuts in G. For a path $\pi = \alpha\beta$ in G, by $\pi + \alpha/\gamma$ we denote the sequence of nodes $\alpha\gamma\beta$ in G. We note that $\pi + \alpha/\gamma$ may not be a path in G. Similarly, if $\pi = \alpha\beta\gamma$ is a path in G, then we define $\pi - \alpha/\beta$ to mean the sequence of nodes $\alpha\gamma$ in G.

Let (π, c) be a configuration of G, and let $\pi = \alpha\beta$. Then we define the set of events corresponding to c in $\pi + \alpha/\gamma$, denoted $c + \alpha/\gamma$, to be

$$\{(e, \rho) \in c \mid \rho \preceq \alpha\} \cup \{(e, \alpha\gamma\rho) \mid (e, \alpha\rho) \in c \text{ and } \rho \neq \epsilon\}.$$

Once again, it is not necessary that $c + \alpha/\gamma$ is a cut in $M_{\alpha\gamma\beta}$.

Similarly, if (π, c) is a configuration of G, and $\pi = \alpha\beta\gamma$, we define the set of events corresponding to c in $\pi - \alpha/\beta$, denoted $c - \alpha/\beta$ to be

$$\{(e, \rho) \in c \mid \rho \preceq \alpha\} \cup \{(e, \alpha\rho) \mid (e, \alpha\beta\rho) \in c,\ \rho \neq \epsilon\}.$$

Let (π, c) be a configuration of G, with $\pi = \alpha\beta\gamma$. We say that β is *completely traversed* in c if all the events in β are included in c, and all processes which react in β have their maximal event in c located in γ. Formally, we represent this as a predicate $Ex(\pi, \alpha, \beta, c)$ which is true iff the conditions below are true:

- $E_{\alpha\beta} - E_\alpha \subseteq c$, and
- For each $p \in P$, if $(e, \alpha\rho v) \in c$ with $e \in E_p$ and $\rho v \preceq \beta$, then there exists $e' \in E_p$ and $\rho' v' \preceq \gamma$ such that $(e', \alpha\beta\rho' v') \in c$.

Similarly, we say β is *completely unexecuted* in c if none of the events in β are included in c: i.e. $(E_{\alpha\beta} - E_\alpha) \cap c = \emptyset$.

Consider a configuration (π, c) of G. We now want to define a way of cutting out loops in π which are completely unexecuted in c. It is not difficult to see that if we remove an unexecuted loop from a configuration, we get another valid configuration. The following lemma proves this claim.

Lemma 3 *If $(\alpha u \beta u \gamma, c)$ is a configuration in G such that $(E_{\alpha u \beta u} - E_{\alpha u}) = \emptyset$ then $c' = c - \alpha u/\beta u$ is a cut in $M_{\alpha u \gamma}$.*

Proof For every $(e, \rho) \in c$ all $(e', \rho') <_{\alpha u \beta u \gamma} (e, \rho)$ will belong to c. Since $(E_{\alpha u \beta u} - E_{\alpha u}) = \emptyset$ so clearly, c' is downward closed with respect to $<_{\alpha u \gamma}$. $\square$

Given a configuration (π, c) in G, we say $\alpha_1 u_1 \beta_1 u_1 \cdots \alpha_n u_n \beta_n u_n \gamma$ is an *unexecuted loop decomposition* of π with respect to c if

- Each $\beta_i u_i$ is unexecuted
- There is no unexecuted loop in $\alpha_1 u_1 \cdots \alpha_n u_n \gamma$ wrt $(c - \alpha_1 u_1/\beta_1 u_1 - \cdots - \alpha_1 u_1 \cdots \alpha_n u_n/\beta_n u_n)$

Correspondingly, $(\alpha_1 u_1 \cdots \alpha_n u_n \gamma, c - \alpha_1 u_1/\beta_1 u_1 - \cdots - \alpha_1 u_1 \cdots \alpha_n u_n/\beta_n u_n)$ is called an *unexecuted loop-free configuration* of (π, c).

There may be several unexecuted loop decompositions for a given configuration, and hence also several unexecuted loop-free configurations. For example, consider a configuration $(v_1 v_2 v_1 v_3 v_2 v_4, c)$ for some MSG such that no events of v_1, v_2 and v_3 is included in c. Then, we have two unexecuted loop decompositions of $(v_1 v_2 v_1 v_3 v_2 v_4$ with respect to c - $v_1 v_3 v_2 v_4$ and $v_1 v_2 v_4$.

We denote by $\lfloor (\pi, c) \rfloor_{ue}$ the set of all unexecuted loop-free configurations of (π, c).

We also define a (unique) *left-most maximal* unexecuted loop decomposition of a configuration (π, c). We define this to be $\alpha_1 u_1 \beta_1 u_1 \cdots \alpha_n u_n \beta_n u_n \gamma$ with $n \geq 0$ where:

- $\pi = \alpha_1 u_1 \beta_1 u_1 \cdots \alpha_n u_n \beta_n u_n \gamma$
- Each $\beta_i u_i$ is the left-most unexecuted loop in the segment $\alpha_i u_i \beta_i u_i \cdots \beta_n u_n \gamma$. That is, for each $\tau_1 v \tau_2$ such that $\tau_1 v \tau_2 = \alpha_i$, there is no $\tau_3 v \preceq u_i \beta_i \alpha_{u+1} u_{i+1} \cdots \alpha_n u_n \beta_n u_n \gamma$ such that $\tau_2 \tau_3 v$ is completely unexecuted.
- Each $\beta_i u_i$ is maximal: That is, no prefix τu_i of $\alpha_{i+1} u_{i+1} \beta_{i+1} u_{i+1} \cdots \alpha_n u_n \beta_n u_n \gamma$ is completely unexecuted.
- γ does not have any completely unexecuted loops.

Each $\alpha_i u_i$ marks the beginning of the i-th maximal loop which is completely unexecuted in c, starting from the left of π. It is easy to see that this decomposition is unique. In the above example, $v_1 v_3 v_2 v_4$ is the unique left most maximal decomposition.

We define the (unique) *left-most maximal* unexecuted loop configuration induced by (π, c), denoted $\lfloor (\pi, c) \rfloor_{lue}$, to be:

$$(\alpha_1 u_1 \cdots \alpha_n u_n \gamma, (\cdots (c - \alpha_1 u_1/\beta_1 u_1) \cdots) \ - \ \alpha_1 u_1 \cdots \alpha_n u_n/\beta_n u_n).$$

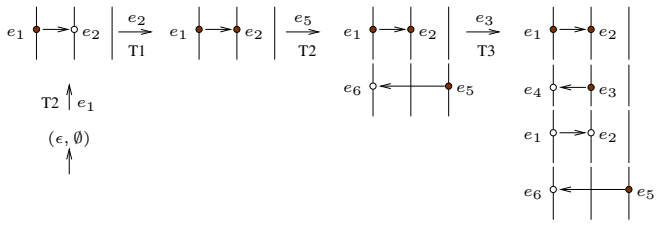


Figure 3: Initial transitions in $\mathcal{T}_G$ for MSG G_1.

We are now in a position to describe a transition system corresponding to the given MSG G. We define $\mathcal{T}_G = (Q, E_G, q_0, \rightarrow)$ where: Q is the set of configurations of G; E_G is the set of events of G as defined in the last section; the initial state is $q_0 = (\epsilon, \emptyset)$; and the transition relation $\rightarrow$ is given by the following rules:

(T1) $(\pi, c) \xrightarrow{e} (\pi, c')$ provided $c' = c \cup \{(e, \rho)\}$ for some $\rho \preceq \pi$ and $(e, \rho) \notin c$.

(T2) $(\pi, c) \xrightarrow{e} (\pi\rho v, c')$ provided $c' = c \cup \{(e, \pi\rho v)\}$ and ρv is loop-free. Also, if $\pi = \epsilon$ then ρv has to be an initial path in G.

(T3) $(\pi_1 u \pi_3, c) \xrightarrow{e} \lfloor (\pi_1 u \pi_2 u \pi_3, c') \rfloor_{lue}$ provided there exists a non-empty and loop-free α and a loop-free β such that $\pi_2 u = \alpha\beta$ and $c + \pi_1 u / \pi_2 u$ is a cut in $M_{\pi_1 u \pi_2 u \pi_3}$ and c' is $(c + \pi_1 u / \pi_2 u) \cup \{(e, \pi_1 u \alpha)\}$.

According to the transition (T1) a process can do an event e in its event set provided it is in the node list π, and does not violate any causality conditions in M_π. This is enforced by requiring the new set of events c' to be a cut in M_π. According to (T2), a process can extend the current node list π by a loop-free path ρv and perform an event e in node v, again provided it does not violate any causality conditions in $M_{\pi\rho v}$. Finally, according to rule (T3), a process p can insert a loop after a prefix $\pi_1 u$ of the current node list π, which comprises a loop-free path α to a node (in which it must then perform an event) followed by a loop-free path back to u. Once again, no causality conditions in the MSC corresponding to the extended path should be violated. After the insertion, we eliminate unexecuted loops in the extended path by taking the induced left-most unexecuted loop-free configuration.

We illustrate these transition rules in Figure 3.

We note that all configurations (π, c) of $\mathcal{T}_G$ reachable from the initial state, satisfy that properties that they always have a process in the last node of π, and also that they are always unexecuted loop free.

4 Correctness of $\mathcal{T}_G$

We now sketch a proof of the correctness of our construction of $\mathcal{T}_G$, by showing that it accepts exactly the language $L^e(G)$. We first show that $\mathcal{T}_G$ is "complete" in the sense that it accepts all event sequences in $L^e(G)$.

Before proceeding with the correctness proof for our construction, we present some simple lemmas which will be useful in our proofs.

Lemma 4 *$(\alpha u\gamma, c)$ is a configuration in G and there is a path $\alpha u\beta u\gamma$ in G such that $\forall(e, \alpha u\rho) \in c$ $\rho \preceq_{ne} \gamma$, if $e \in E_p$ for some $p \in P$ then $E_{\beta u} \bigcap E_p = \emptyset$. Then, $c' = c + \alpha u/\beta u$ is a cut in $M_{\alpha u\beta u\gamma}$.*

Proof Clearly, for any $(e, \rho) \in c$, $(e, \rho + \alpha u/\beta u) \in c'$. As c is downward closed wrt $<_{\alpha u\gamma}$ this means, c is downward closed wrt $<_{\alpha u\beta u\gamma}$. So, c is a cut in $M_{\alpha u\beta u\gamma}$. □

Lemma 5 *Let (π, c) be a configuration in G. For all $\alpha \preceq_{ne} \pi$ such that $\exists(e, \alpha) \in c$ and $\alpha\beta = \pi$, if $(\alpha', c'_\alpha) \in \lfloor(\alpha, c - \alpha/\beta)\rfloor_{ue}$ and $(\beta', c'_\beta) \in \lfloor(\beta, c - \epsilon/\alpha)\rfloor_{ue}$ then $(\alpha'\beta', c'_\alpha \bigcup(c'_\beta + \epsilon/\alpha)) \in \lfloor(\pi, c)\rfloor_{ue}$.*

Proof This easily follows from the fact in configuration (π, c), there is an execution in α. □

We will show that $L^e(G) \subseteq L(\mathcal{T}_G)$. Let $w \in L^e(G)$ with $w = e_1e_2 \cdots e_n$. Then we know by Lemma 1 that there is a sequence of incremental cuts $c_0, c_1, \cdots, c_n$ in M_π for some initial path π in G, such that $c_0 = \emptyset$ and for each $i \in \{0, \ldots, n-1\}$, $c_{i+1} - c_i = \{(e_{i+1}, \rho_{i+1})\}$ for some $\rho_{i+1} \preceq \pi$. For each $i \in \{0, \ldots, n\}$, let θ_i be $\max\{\rho_j \mid j \leq i\}$. We claim that w has a run

$$(\pi_0, d_0) \xrightarrow{e_1} (\pi_1, d_1) \xrightarrow{e_2} \cdots \xrightarrow{e_n} (\pi_n, d_n)$$

in $\mathcal{T}_G$, where $\pi_0 = \epsilon$, $d_0 = \emptyset$, and for all $j \in \{0, \ldots, n\}$, $(\pi_j, d_j) \in \lfloor(\theta_j, c_j)\rfloor_{ue}$.

We do this by showing, using induction on i, that for each $i \in \{0, \ldots, n\}$ we can produce a run

$$(\pi_0, d_0) \xrightarrow{e_1} (\pi_1, d_1) \xrightarrow{e_2} \cdots \xrightarrow{e_i} (\pi_i, d_i)$$

in $\mathcal{T}_G$ such that $(\pi_0, d_0) = (\epsilon, \emptyset)$, and for each $l \in \{1, \ldots, i\}$, $(\pi_l, d_l) \in \lfloor(\theta_l, c_l)\rfloor_{ue}$.

For $i = 0$, we get $\pi_0 = \epsilon$ and $d_0 = \emptyset$ as $c_0 = \emptyset$.

Assume that the hypothesis is true for some $i < n$. Then, $(\pi_0, d_0) = (\epsilon, \emptyset)$ and for each $l \in [1, i](\pi_l, d_l) \in \lfloor(\theta_l, c_l)\rfloor_{ue}$. Let an unexecuted loop decomposition of θ_i wrt c_i be $\alpha_1u_1\beta_1u_1 \cdots \alpha_ju_j\beta_ju_j\gamma$ and correspondingly,

$(\pi_i, d_i) = (\alpha_1u_1 \cdots \alpha_ju_j\gamma, c_i - \alpha_1u_1/\beta_1u_1 - \cdots - \alpha_1u_1 \cdots \alpha_ju_j/\beta_ju_j)$.

We now consider $\{(e_{i+1}, \rho_{i+1})\}$ and show that $(\pi_i, d_i) \xrightarrow{e_{i+1}} (\pi_{i+1}, d_{i+1})$ where $(\pi_{i+1}, d_{i+1}) \in \lfloor(\theta_{i+1}, c_{i+1})\rfloor_{ue}$ following the transition rules for $\mathcal{T}_G$. Following cases need to be analyzed:

- $\theta_i \prec \theta_{i+1}$. This means, $\rho_{i+1} = \theta_{i+1} = \theta_i\rho$ for some non-empty path ρ in G. Let $\alpha'_1u'_1\beta'_1u'_1 \cdots \alpha'_ku'_k\beta'_ku'_k\gamma'$ be an unexecuted loop free decomposition of ρ wrt (e_{i+1}, ρ). We set $(\pi_{i+1}, d_{i+1}) = (\pi_i\tau, d_i \bigcup\{(e_{i+1}, \pi_i\tau)\}$ where $\tau = \alpha'_1u'_1 \cdots \alpha'_ku'_k\gamma'$.

  Since $\exists(e, \theta_i) \in c_{i+1}$ so by Lemma 5, $(\pi_{i+1}, d_{i+1}) \in \lfloor(\theta_{i+1}, c_{i+1})\rfloor_{ue}$.

  By Lemma 3, as (θ_{i+1}, c_{i+1}) is a valid configuration, so, d_{i+1} is a valid cut along $M_{\pi_{i+1}}$. Also, τ is clearly unexecuted loop free. So, by transition rule (T2) we have $(\pi_i, d_i) \xrightarrow{e_{i+1}} (\pi_{i+1}, d_{i+1})$ in $\mathcal{T}_G$.

- $\theta_i = \theta_{i+1}$, $\rho_{i+1} = \alpha_1 u_1 \beta_1 u_1 \cdots \alpha_k u_k \beta_k u_k \rho$ where $\rho \preceq_{ne} \alpha_{k+1} u_{k+1}$ if $k < j$ else $\rho \preceq_{ne} \gamma$. Then we define (π_{i+1}, d_{i+1})

  $= (\alpha_1 u_1 \cdots \alpha_j u_j \gamma, (c_i \bigcup \{(e_{i+1}, \alpha_1 u_1 \cdots \alpha_k u_k \rho)\}) -$
  $\alpha_1 u_1 / \beta_1 u_1 - \cdots \alpha_1 u_1 \cdots \alpha_j u_j / \beta_j u_j)$

  $= (\pi_i, d_i \bigcup \{(e_{i+1}, \alpha_1 u_1 \cdots \alpha_k u_k \rho)\})$ as ρ does not have any unexecuted loop.

  Clearly, $(\pi_{i+1}, d_{i+1}) \in \lfloor (\theta_{i+1}, c_{i+1}) \rfloor_{ue}$. By Lemma 3, as (θ_{i+1}, c_{i+1}) is a valid configuration, so, d_{i+1} is a valid cut in $M_{\pi_{i+1}}$. So, by transition rule (T1) we have $(\pi_i, d_i) \xrightarrow{e_{i+1}} (\pi_{i+1}, d_{i+1})$ in $\mathcal{T}_G$.

- $\theta_i = \theta_{i+1}$, $\rho_{i+1} = \alpha_1 u_1 \beta_1 u_1 \cdots \alpha_k u_k \beta_k u_k \rho$ for some $k \leq j$ and $\rho \preceq_{ne} \beta_k u_k$.

  Let τ_1 be $\alpha_1 u_1 \beta_1 u_1 \cdots \alpha_k u_k$ and τ_2 be $\alpha_{k+1} u_{k+1} \beta_{k+1} u_{k+1} \cdots \alpha_j u_j \beta_j u_j \gamma$. Let $\rho\, \eta = \beta_k u_k$. Also, let τ_1' and τ_2' be $\alpha_1 u_1 \cdots \alpha_k u_k$ and $\alpha_{k+1} u_{k+1} \cdots \alpha_j u_j \gamma$ respectively. Let ρ' be some unexecuted loop free decomposition of ρ wrt $\{(e_{i+1}, \rho)\}$ and let η' be some loop free form of η. Clearly both τ_1' and η' ends in u_k.

  By hypothesis, $(\tau_1' \tau_2', d_i)$ is a configuration in G. So, by Lemma 4, $(\tau_1' \rho' \eta_1' \tau_2', d_i + \tau_1' / \rho' \eta')$ is a configuration of G as well.

  We set (π_{i+1}, d_{i+1}) to

  $\lfloor (\tau_1' \rho' \eta_1' \tau_2', d_i + \tau_1' / \rho' \eta') \bigcup \{(e_{i+1}, \tau_1' \rho')\} \rfloor_{lue}$.

  So, following rule (T3) we can have a transition $(\pi_i, d_i) \xrightarrow{e_{i+1}} (\pi_{i+1}, d_{i+1})$ in $\mathcal{T}_G$. Also by Lemma 5, it can be easily shown that

  $(\pi_{i+1}, d_{i+1}) \in \lfloor (\theta_{i+1}, c_{i+1}) \rfloor_{ue}$.

  This case is illustrated in Figure 4.

We now argue the "soundness" of $\mathcal{T}_G$, by showing that if $w \in L(\mathcal{T}_G)$ then $w \in L^e(G)$. Let $w \in L(\mathcal{T}_G)$ with $w = e_1 e_2 \cdots e_n$ such that $(\pi_0, c_0) \xrightarrow{e_1} (\pi_1, c_1) \cdots \xrightarrow{e_n} (\pi_n, c_n)$ is a run in $\mathcal{T}_G$. We note that each π_i must be an initial path in G. We claim that w will produce a sequence of incremental cuts $c_0', c_1', \ldots, c_n'$ in M_{π_n} where $c_0' = \emptyset$, $c_n' = c_n$, and for each $i \in \{0, \ldots, n-1\}$, $c_{i+1}' - c_i' = \{(e_{i+1}, \rho_{i+1})\}$ for some $\rho_{i+1} \preceq \pi_n$. We prove this by induction on length of w.

For $n = 0$, we have $\pi_0 = \epsilon$ and $c_0' = c_0 = \emptyset$ and we have nothing more to prove.

For the induction step, assuming that the claim holds good for w of length n, let us consider the case for $|w| = n + 1$.

Let $w = e_1 e_2 \cdots e_{n+1} \in \mathrm{L}(\mathcal{T}_G)$ which creates a run $(\pi_0, c_0) \xrightarrow{e_1} (\pi_1, c_1) \cdots \xrightarrow{e_{n+1}} (\pi_{n+1}, c_{n+1})$ in $\mathcal{T}_G$. By induction hypothesis, we have a sequence of incremental cuts $d_0, d_1, \cdots d_n$ in M_{π_n} such that $d_0 = \emptyset$, $d_n = c_n$ and for each $i \in [0, n-1]$, $d_{i+1} - d_i = \{(e_{i+1}, \rho_{i+1})\}$ for some $\rho_{i+1} \preceq_{ne} \pi_n$.

Depending on the type of transition $(\pi_n, c_n) \xrightarrow{e_{n+1}} (\pi_{n+1}, c_{n+1})$ in $\mathcal{T}_G$ we have the following cases.

- Case (T1): $(\pi_n, c_n) \xrightarrow{e_{n+1}} (\pi_{n+1}, c_{n+1})$ where $c_{n+1} = c_n \bigcup \{(e_{n+1}, \rho)\}$ for some $\rho \preceq \pi_n$.

  Clearly, $\pi_{n+1} = \pi_n$. For each $i \in [0, n]$ we set c_i' to d_i and $c_{n+1}' = c_{n+1}$. Then, $c_0', \cdots, c_{n+1}'$ is the required sequence of incremental cuts in $M_{\pi_{n+1}}$ satisfying the three conditions.

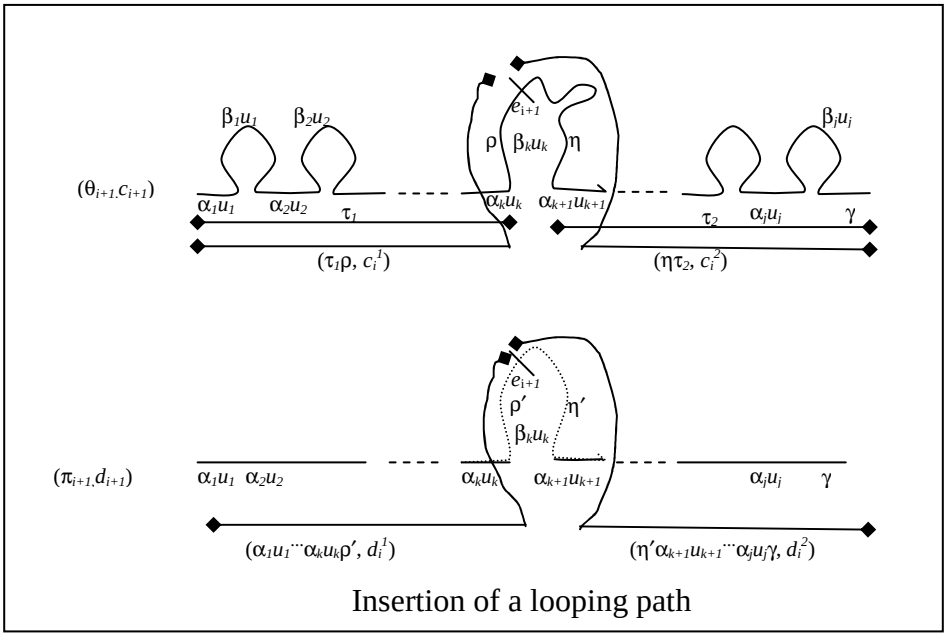


Figure 4: Insertion Of Looping Path - Completeness

- Case (T2): $(\pi_n, c_n) \stackrel{e_{n+1}}{\rightarrow} (\pi_n\rho, c_{n+1})$ where $c_{n+1} = c_n \bigcup \{(e_{n+1}, \pi_n\rho)\}$ for a non-empty and loop free ρ.

  We note that, $\pi_{n+1} = \pi_n\rho$. A cut in M_{π_n} is also a cut in $M_{\pi_n\rho}$. For each $i \in [0, n]$ we set c'_i to d_i and $c'_{n+1} = c_{n+1}$.

  For $i \in [0, n]$, d_i is a cut in M_{π_n}, and so c'_i is a cut in $M_{\pi_{n+1}}$. Then, $c'_0, \cdots, c'_{n+1}$ is the required sequence of incremental cuts in $M_{\pi_{n+1}}$ satisfying the three conditions.

- Case (T3): $(\pi_n, c_n) \stackrel{e_{n+1}}{\rightarrow} (\pi_{n+1}, c_{n+1})$ is an instance of T3. Thus π_n is of the form $\tau_1 u \tau_3$, there exists a path $\tau_2 u$ in G of the form $\alpha\beta$ for some non empty and loop free α and loop free β such that $(c_n + \tau_1 u/\tau_2 u)$ is a cut in $M_{\tau_1 u \tau_2 u \tau_3}$ and $(\pi_{n+1}, c_{n+1}) = \lfloor (\tau_1 u \tau_2 u \tau_3, c') \rfloor_{lue}$ where $c' = (c_n + \tau_1 u/\tau_2 u) \bigcup \{(e_{n+1}, \tau_1 u \alpha)\}$.

  Note that $\tau_1 u$ and τ_3 is unexecuted loop free. Let $\eta_1 = max\{\rho \preceq \tau_1 u | (e, \rho) \in c_n\}$ and let $\tau_1 u = \eta_1 \eta_2$.

  In c', there is no execution in α except in its last node. Since $\tau_1 u$ is unexecuted loop free, so if there is any unexecuted loop formed in $\tau_1 u \alpha$, wrt c', then, it must be in $\eta_2 \alpha$. Let $\alpha_1 u_1 \beta_1 u_1 \cdots \alpha_j u_j \beta_j u_j \gamma$ be left-most maximal unexecuted loop decomposition of $\eta_2 \alpha$ wrt $\{(e_{n+1}, \eta_2 \alpha)\}$.

  β is also loop free and unexcuted. Let $\eta_3 = min\{\rho \preceq \tau_3 | (e, \tau_1 u \rho) \in c_n\}$ and let $\tau_3 = \eta_3 \eta_4$. So, if there is any unexecuted loop formed in $\beta\tau_3$, wrt c'

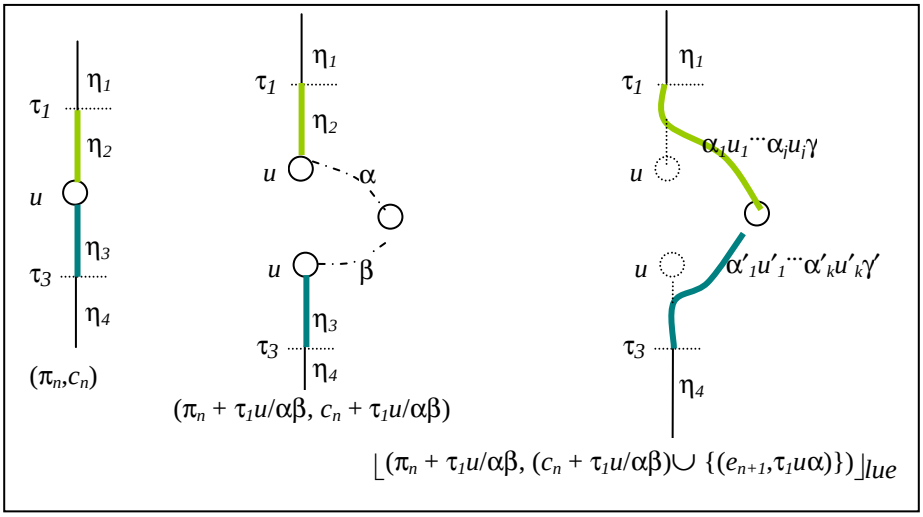


Figure 5: Insertion Of Looping Path - Soundness

, then it must be in $\beta\eta_3$. Let $\alpha'_1u'_1\beta'_1u'_1\cdots\alpha'_ku'_k\beta'_ku'_k\gamma'\eta_4$ be the left-most maximal unexecuted loop decomposition of $\beta\tau_3$ wrt c'.

Then, (π_{n+1}, c_{n+1}) will be

$$(\eta_1\alpha_1u_1\cdots\alpha_ju_j\gamma\alpha'_1u'_1\cdots\alpha'_ku'_k\gamma'\eta_4,$$
$$c' - \eta_1\alpha_1u_1/\beta_1u_1 - \cdots - \eta_1\alpha_1u_1\cdots\alpha_ju_j/\beta_ju_j$$
$$-\eta_1\alpha_1u_1\cdots\alpha_ju_j\gamma\alpha'_1u'_1/\beta'_1u'_1 - \cdots -$$
$$\eta_1\alpha_1u_1\cdots\alpha_ju_j\gamma\alpha'_1u'_1\cdots\alpha'_ku'_k/\beta'_ku'_k).$$

By our hypothesis, $d_n = c_n$. By transition, $(c_n + \tau_1u/\tau_2u)$ is a cut in $M_{\tau_1u\tau_2\pi_3}$ and so, $(d_n + \tau_1u/\tau_2u)$ is a cut in $M_{\tau_1u\tau_2\pi_3}$. So, clearly, $(d_i + \tau_1u/\tau_2u)$ is a cut in $M_{\tau_1u\tau_2\pi_3}$.

For each $i \in [0, n]$ we set c'_i to

$$d_i + \tau_1u/\tau_2u - \eta_1\alpha_1u_1/\beta_1u_1 - \cdots$$
$$-\eta_1\alpha_1u_1\cdots\alpha_ju_j/\beta_ju_j - \eta_1\alpha_1u_1\cdots\alpha_ju_j\gamma\alpha'_1u'_1/\beta'_1u'_1 - \cdots$$
$$-\eta_1\alpha_1u_1\cdots\alpha_ju_j\gamma\alpha'_1u'_1\cdots\alpha'_ku'_k/\beta'_ku'_k$$

As mentioned above, $\forall(e,\tau) \in d_n$, either $\tau \preceq \eta_1$ or $\eta_1\eta_2\eta_3 \preceq \tau$. So, for each $i \in [0, n-1]$, $\forall(e,\tau) \in d_i$, either $\tau \preceq \eta_1$ or $\eta_1\eta_2\eta_3 \preceq \tau$. So, clearly, c'_i is a cut in $M_{\eta_1\alpha_1u_1\cdots\alpha_ju_j\gamma\alpha'_1u'_1\cdots\alpha'_ku'_k\gamma'\eta_4}$ or $M_{\pi_{n+1}}$.

We set c'_n to c_{n+1}. So, now we have got incremental cuts on $c'_0, c'_1, \cdots, c'_{n+1}$ in $M_{\pi_{n+1}}$ satisfying the three clauses.

So, all the three conditions are satisfied. This case is as illustrated in Figure 5.

This proves the correctness of our construction.

5 Reducing $\mathcal{T}_G$

In this section our aim is to show that the state-space of the transition system $\mathcal{T}_G$ can be reduced, by observing that we can remove fully traversed prefixes and loops from configurations without affecting the language generated by the transition system. To do this it will be convenient to make use of the notion of bisimulation and some results concerning it.

Let $\mathcal{T} = (Q, A, q_0, \rightarrow)$ be a transition system. A binary relation $\mathcal{R} \subseteq Q \times Q$ is called a *bisimulation relation* on $\mathcal{T}$ if

- Whenever $(q, r) \in \mathcal{R}$ and $q \xrightarrow{a} q'$ for some $q' \in Q$ and $a \in A$, there exists $r' \in Q$ such that $r \xrightarrow{a} r'$ and $(q', r') \in \mathcal{R}$.
- Whenever $(q, r) \in \mathcal{R}$ and $r \xrightarrow{a} r'$ for some $r' \in Q$ and $a \in A$, there exists $q' \in Q$ such that $q \xrightarrow{a} q'$ and $(q', r') \in \mathcal{R}$.

We say that states q and r in $\mathcal{T}$ are *bisimilar* if there exists a bisimulation relation $\mathcal{R}$ on $\mathcal{T}$ with $(q, r) \in \mathcal{R}$. It is easy to see that bisimilar states satisfy the property that the event sequences in $\mathcal{T}$ starting from any of them are identical.

Proposition 6 *Let $\mathcal{T} = (Q, A, q_0, \rightarrow)$ be a transition system. Let $q, r \in Q$ be two bisimilar states. Then $L_q(\mathcal{T}) = L_r(\mathcal{T})$.* □

We note that the identity relation is always a bisimulation relation on a transition system $\mathcal{T}$. Further, if $\mathcal{R}$ and $\mathcal{S}$ are bisimulation relations on a transition system $\mathcal{T}$ so is the relation $\mathcal{R} \cup \mathcal{S}$. Finally, let us define the reflexive transitive closure of $\mathcal{R}$, denoted $\mathcal{R}^*$, by $(q, r) \in \mathcal{R}^*$ iff there exist states $r_0, r_1, \ldots, r_k$ in Q, with $k \geq 0$, such that $r_0 = q$, $r_k = r$, and $(r_i, r_{i+1}) \in \mathcal{R}$ for each $i \in \{0, \ldots, k-1\}$. Then if $\mathcal{R}$ is a bisimulation relation on a transition system $\mathcal{T}$, so is $\mathcal{R}^*$.

The lemma below shows how we can reduce the state space of a transition system using a bisimulation relation on it.

Lemma 7 *Let $T = (Q, A, q_0, \rightarrow)$ be a transition system, and let $\mathcal{R}$ be bisimulation relation on $\mathcal{T}$. Consider a transition system $\mathcal{T}' = (Q', A, q_0, \Rightarrow)$ where $Q' \subseteq Q$, and $\rightarrow'$ satisfies the following conditions for any $q' \in Q'$:*

- *whenever $q' \xrightarrow{a} r$ in $\mathcal{T}$, there exists a state $r' \in Q'$ such that $q' \overset{a}{\Rightarrow} r'$ in $\mathcal{T}'$, and $(r, r') \in \mathcal{R}$.*
- *whenever $q' \overset{a}{\Rightarrow} r'$ in $\mathcal{T}'$, there exists $r \in Q$ such that $q' \xrightarrow{a} r$ in $\mathcal{T}$ and $(r, r') \in \mathcal{R}$.*

Then $L(\mathcal{T}) = L(\mathcal{T}')$.

Proof We first show that $\mathrm{L}(\mathcal{T}) \subseteq \mathrm{L}(\mathcal{T}')$. We prove by induction on length of $w \in \sum^*$ that if $q_0 \xrightarrow{w}{}^* q$ in $\mathcal{T}$ then $\exists q' \in Q'$ s.t. $q_0 \overset{w}{\Longrightarrow}{}^* q'$ in $\mathcal{T}'$ and $(q, q') \in \mathcal{R}^*$.

For $w = \epsilon$, it is trivial as $q_0 \xrightarrow{\epsilon}{}^* q_0$ and $q_0 \overset{\epsilon}{\Longrightarrow}{}^* q_0$ and clearly $(q_0, q_0) \in \mathcal{R}^*$.

Assume that the hypothesis is true for any $|w| = n$. Consider a $w \cdot a \in L(\mathcal{T})$ such that $q_0 \xrightarrow{w}{}^* q \xrightarrow{a} r$. Then,

1. By hypothesis, $\exists q' \in Q'$, such that $q_0 \overset{w}{\Longrightarrow}{}^* q'$ in $\mathcal{T}'$ and $(q, q') \in \mathcal{R}^*$.

2. $q \xrightarrow{a} r$ and $(q, q') \in \mathcal{R}^*$. So, $\exists r'' \in Q$ such that $q' \xrightarrow{a} r''$. Clearly, $(r, r'') \in \mathcal{R}^*$.

3. Since $q' \in Q'$, so, by construction, $\exists r' \in Q'$, such that $q' \stackrel{a}{\Rightarrow} r'$ in $\mathcal{T}'$ and $(r'', r') \in \mathcal{R}^*$.

4. Since $(r, r'') \in \mathcal{R}^*$ and $(r'', r') \in \mathcal{R}^*$ so $(r, r') \in \mathcal{R}^*$. Then in $\mathcal{T}'$, we have $q_0 \stackrel{w}{\Longrightarrow}^* q' \stackrel{a}{\Rightarrow} r'$ such that $(r, r') \in \mathcal{R}^*$.

This proves that $\mathrm{L}(\mathcal{T}) \subseteq \mathrm{L}(\mathcal{T}')$. Now we show $\mathrm{L}(\mathcal{T}') \subseteq \mathrm{L}(\mathcal{T})$. We prove by induction on length of $w \in \sum^*$ that if $q_0 \stackrel{w}{\Longrightarrow}^* q'$ in $\mathcal{T}'$ then $\exists q \in Q$ s.t. $q_0 \xrightarrow{w}^* q$ in $\mathcal{T}$ and $(q, q') \in \mathcal{R}^*$.

For $w = \epsilon$, it is trivial as $q_0 \stackrel{\epsilon}{\Longrightarrow}^* q_0$ and $q_0 \xrightarrow{\epsilon}^* q_0$ and clearly $(q_0, q_0) \in \mathcal{R}^*$. Assume that the hypothesis is true for any $|w| = n$.

Consider a $w \cdot a \in L(\mathcal{T}')$ such that $q_0 \stackrel{w}{\Longrightarrow}^* q' \stackrel{a}{\Rightarrow} r'$. Then,

1. By hypothesis, $\exists q \in Q$ s.t. $q_0 \xrightarrow{w}^* q$ in $\mathcal{T}$ and $(q, q') \in \mathcal{R}^*$.

2. As $q' \stackrel{a}{\Rightarrow} r'$ so by construction, $\exists r'' \in Q$ such that $q' \xrightarrow{a} r''$ in T and $(r'', r') \in \mathcal{R}^*$.

3. As $(q, q') \in \mathcal{R}^*$ and $q' \xrightarrow{a} r''$ so $\exists r \in Q$ such that $q \xrightarrow{a} r$ in $\mathcal{T}$ and $(r, r'') \in \mathcal{R}^*$.

4. As $(r, r'') \in \mathcal{R}^*$ and $(r'', r') \in \mathcal{R}^*$ so $(r, r') \in \mathcal{R}^*$. Then, in $\mathcal{T}$, we have $q_0 \xrightarrow{w}^* q \xrightarrow{a} r$ and $(r, r') \in \mathcal{R}^*$.

This proves that $\mathrm{L}(\mathcal{T}') \subseteq \mathrm{L}(\mathcal{T})$. □

We return now to our transition system $\mathcal{T}_G$ corresponding to the given MSG G.

We give two bisimulation relations in $\mathcal{T}_G$ which are stated informally below:

1. A configuration in $\mathcal{T}_G$ which has a completely traversed prefix in its node list is bisimilar to a corresponding configuration which has the prefix removed.

2. A configuration in $\mathcal{T}_G$ which has a completely traversed loop in its node list is bisimilar to a corresponding configuration which has the loop removed.

Before proving that these are actually bisimulation relations, first we prove few simple lemmas regarding the operation on cuts, which will be used in the bisimulation proofs going forward.

Lemma 8 *If $(\alpha\beta\gamma, c)$ is a configuration in G, then, $(c - \epsilon/\alpha) - \epsilon/\beta = c - \epsilon/\alpha\beta$.*

Proof $(c - \epsilon/\alpha) - \epsilon/\beta = \{(e, \rho) | (e, \alpha\beta\rho) \in c \text{ and } \rho \preceq_{ne} \gamma\} = c - \epsilon/\alpha\beta$. □

Lemma 9 *If $(\alpha\beta_1\beta_2\beta_3\gamma, c)$ is a configuration in G, then, $(c - \epsilon/\alpha) - \beta_1/\beta_2 = (c - \alpha\beta_1/\beta_2) - \epsilon/\alpha$.*

Proof $(c - \epsilon/\alpha) - \beta_1/\beta_2$
$= \{(e,\rho)|(e,\alpha\rho) \in c \text{ and } \rho \preceq_{ne} \beta_1\} \bigcup \{(e,\beta_1\rho)|(e,\alpha\beta_1\beta_2\rho) \in c \text{ and } \rho \preceq_{ne} \beta_3\gamma\}$
Also, $(c - \alpha\beta_1/\beta_2) - \epsilon/\alpha$
$= (\{(e,\rho)|(e,\rho) \in c \text{ and } \rho \preceq \alpha\beta_1 \} \bigcup \{(e,\alpha\beta_1\rho)|(e,\alpha\beta_1\beta_2\rho) \in c \text{ and } \rho \preceq_{ne} \beta_3\gamma\})$ - ϵ/α
$= \{(e,\rho)|(e,\alpha\rho) \in c \text{ and } \rho \preceq_{ne} \beta_1\} \bigcup \{(e,\beta_1\rho)|(e,\alpha\beta_1\beta_2\rho) \in c \text{ and } \rho \preceq_{ne} \beta_3\gamma\}$ □

Lemma 10 *if* $(\alpha\beta_1\gamma_1\beta_2\gamma_2, c)$ *is a configuration in* G*, then,* $(c-\alpha/\beta_1)-\alpha\gamma_1/\beta_2 = (c - \alpha\beta_1\gamma_1/\beta_2) - \alpha/\beta_1$

Proof $(c - \alpha/\beta_1) - \alpha\gamma_1/\beta_2$
$= \{(e,\rho)|(e,\rho) \in c \text{ and } \rho \preceq \alpha\}$
$\bigcup(\{(e,\alpha\rho)|(e,\alpha\beta_1\rho) \in c \text{ and } \rho \preceq_{ne} \gamma_1\}$
$\bigcup\{(e,\alpha\gamma_1\rho)|(e,\alpha\beta_1\gamma_1\beta_2\rho) \in c \text{ and } \rho \preceq_{ne} \gamma_2\})$.
Also, $(c - \alpha\beta_1\gamma_1/\beta_2) - \alpha/\beta_1$
$= (\{(e,\rho)|(e,\rho) \in c \text{ and } \rho \preceq \alpha\beta_1\gamma_1) \bigcup\{(e,\alpha\beta_1\gamma_1\rho)|(e,\alpha\beta_1\gamma_1\beta_2\rho) \in c \text{ and } \rho \preceq_{ne} \gamma_2\})$ - α/β_1
$= \{(e,\rho)|(e,\rho) \in c \text{ and } \rho \preceq \alpha\}$
$\bigcup(\{(e,\alpha\rho)|(e,\alpha\beta_1\rho) \in c \text{ and } \rho \preceq_{ne} \gamma_1\}$
$\bigcup\{(e,\alpha\gamma_1\rho)|(e,\alpha\beta_1\gamma_1\beta_2\rho) \in c \text{ and } \rho \preceq_{ne} \gamma_2\})$. □

Lemma 11 *If* $(\alpha\beta_1\gamma_1\gamma_2, c)$ *is a configuration in* G*, then,* $(c-\alpha/\beta_1)+\alpha\gamma_1/\beta_2 = (c + \alpha\beta_1\gamma_1/\beta_2) - \alpha/\beta_1$.

Proof $(c - \alpha/\beta_1) + \alpha\gamma_1/\beta_2$
$= \{(e,\rho)|(e,\rho) \in c \text{ and } \rho \preceq \alpha\}$
$\bigcup\{(e,\alpha\rho)|(e,\alpha\beta_1\rho) \in c \text{ and } \rho \preceq_{ne} \gamma_1\}$
$\bigcup\{(e,\alpha\gamma_1\beta_2\rho)|(e,\alpha\beta_1\gamma_1\rho) \in c \text{ and } \rho \preceq_{ne} \gamma_2\}$.
Also, $(c + \alpha\beta_1\gamma_1/\beta_2) - \alpha/\beta_1$
$= (\{(e,\rho)|(e,\rho) \in c \text{ and } \rho \preceq \alpha\beta_1\gamma_1\}$
$\bigcup\{(e,\alpha\beta_1\gamma_1\beta_2\rho)|(e,\alpha\beta_1\gamma_1\rho) \in c \text{ and } \rho \preceq_{ne} \gamma_2\}) - \alpha/\beta_1$
$= \{(e,\rho)|(e,\rho) \in c \text{ and } \rho \preceq \alpha\}$
$\bigcup\{(e,\alpha\rho)|(e,\alpha\beta_1\rho) \in c \text{ and } \rho \preceq_{ne} \gamma_1\}$
$\bigcup\{(e,\alpha\gamma_1\beta_2\rho)|(e,\alpha\beta_1\gamma_1\rho) \in c \text{ and } \rho \preceq_{ne} \gamma_2\}$.
□

Lemma 12 *If* $(\alpha\gamma_1\gamma_2, c)$ *is a configuration in* G*, then,* $(c+\alpha/\beta_1)+\alpha\beta_1\gamma_1/\beta_2 = (c + \alpha\gamma_1/\beta_2) + \alpha/\beta_1$.

Proof $(c + \alpha/\beta_1) + \alpha\beta_1\gamma_1/\beta_2$
$= (\{(e,\rho)|(e,\rho) \in c \text{ and } \rho \preceq \alpha\}$
$\bigcup\{(e,\alpha\beta_1\rho)|(e,\alpha\rho) \in c \text{ and } \rho \preceq_{ne} \gamma_1\}$
$\bigcup\{((e,\alpha\beta_1\gamma_1\beta_2\rho)|(e,\alpha\gamma_1\rho) \in c \text{ and } \rho \preceq_{ne} \gamma_2\}$
Also, $(c + \alpha\gamma_1/\beta_2) + \alpha/\beta_1$
$= (\{(e,\rho)|(e,\rho) \in c \text{ and } \rho \preceq \alpha\gamma_1\}$
$\bigcup\{(e,\alpha\gamma_1\beta_2\rho)|(e,\alpha\gamma_1\rho) \in c \text{ and } \rho \preceq_{ne} \gamma_2\}) + \alpha/\beta_1$

$= (\{(e,\rho)|(e,\rho) \in c \text{ and } \rho \preceq \alpha\}$
$\bigcup\{(e,\alpha\beta_1\rho)|(e,\alpha\rho) \in c \text{ and } \rho \preceq_{ne} \gamma_1\}$
$\bigcup\{((e,\alpha\beta_1\gamma_1\beta_2\rho)|(e,\alpha\gamma_1\rho) \in c \text{ and } \rho \preceq_{ne} \gamma_2\}$ □

We now present two lemmas related to removal and addition of some completely traversed prefix to a configuration.

Lemma 13 *Let $(\alpha\beta, c)$ be a configuration in G, such that $Ex(\alpha\beta, \epsilon, \alpha, c)$ is true. Then, (β, c') is also a configuration in G, where $c' = c - \epsilon/\alpha$.*

Proof Assume c' is not a cut in M_β. This means, $\exists(e,\rho) \in c'$ and $\exists(e',\rho')$ in M_β such that $(e',\rho') <_\beta (e,\rho)$ but $(e',\rho') \notin c'$. Then, $(e',\alpha\rho') <_{\alpha\beta} (e,\alpha\rho)$ but $(e',\alpha\rho') \notin c$. This contradicts that c being a cut in $M_{\alpha\beta}$ is downward closed wrt $<_{\alpha\beta}$. So, c' is a cut in M_β. □

Lemma 14 *Let (β, c') be a configuration in G, and let $c = (c' + \epsilon/\alpha) \bigcup E_\alpha$. If $Ex(\alpha\beta, \epsilon, \alpha, c)$ is true then $(\alpha\beta, c)$ is a configuration in G.*

Proof As (β, c') be a configuration in G, all events in E_α is included in c and $Ex(\alpha\beta, \epsilon, \alpha, c)$ is true, clearly, c is downward closed wrt $<_{\alpha\beta}$. □

Consider the relation $\rightsquigarrow_1$ on the states of $\mathcal{T}_G$ defined below, which relates a configuration with the one obtained from it by deleting a fully traversed prefix. By Lemma 13 we get a valid configuration after deleting the prefix and by Lemma 14 we get a valid configuration after adding the prefix.

We define $\rightsquigarrow_1$ as follows: $(\alpha\beta, c) \rightsquigarrow_1 (\beta, c')$ provided

- α is non-empty,
- $Ex(\alpha\beta, \epsilon, \alpha, c)$ holds, and
- $c' = c - \epsilon/\alpha$.

We now show that the relation $\rightsquigarrow_1$ is a bisimulation relation on the states of $\mathcal{T}_G$.

Let us assume $\rightsquigarrow_1$ is a bisimilarity relation. Then, if $(\alpha\beta, c) \rightsquigarrow_1 (\beta, c')$ then:

1. If $(\alpha\beta, c) \xrightarrow{e} (\alpha\beta_1, c_1)$ then $(\beta, c') \xrightarrow{e} (\beta_1, c_1')$ and $(\alpha\beta', c') \rightsquigarrow_1 (\beta_1, c_1')$.
2. If $(\beta, c') \xrightarrow{e} (\beta_1, c_1')$ then $(\alpha\beta, c) \xrightarrow{e} (\alpha\beta_1, c_1)$ and $(\alpha\beta', c') \rightsquigarrow_1 (\beta_1, c_1')$.

We first prove the first statement. $Ex(\alpha\beta, \epsilon, \alpha, c)$ is true wrt c, so, e can not be an event in the prefix α. Also, this means $Ex(\alpha\beta_1, \epsilon, \alpha, c_1)$ holds. Following cases have to be considered:

1. Case (T1): $(\alpha\beta, c) \xrightarrow{e} (\alpha\beta, c_1)$ where $c_1 = c \bigcup \{(e, \alpha\rho)\}$ for some $\rho \preceq_{ne} \beta$. We set c_1' to $c' \bigcup \{(e,\rho)\}$. Then,
   $c_1' = c' \bigcup \{(e,\rho)\} = (c - \epsilon/\alpha) \bigcup \{(e,\rho)\} = (c \bigcup \{(e,\alpha\rho)\}) - \epsilon/\alpha = c_1 - \epsilon/\alpha$.
   Since $c_1 = c \bigcup \{(e,\alpha\rho)\}$ is a cut in $M_{\alpha\beta}$ so by Lemma 13, c_1' is a cut in M_β. $(\beta, c') \xrightarrow{e} (\beta, c_1')$ and as $Ex(\alpha\beta, \epsilon, \alpha, c_1)$ holds so, $(\alpha\beta, c_1) \rightsquigarrow_1 (\beta, c_1')$.

2. Case (T2): $(\alpha\beta, c) \xrightarrow{e} (\alpha\beta\tau, c')$ where $c' = c \bigcup \{(e, \alpha\beta\tau)\}$ for some non empty τ. We set c'_1 to $c' \bigcup \{(e, \beta\tau)\}$. Then,

   $c'_1 = c' \bigcup \{(e, \beta\tau)\} = (c - \epsilon/\alpha) \bigcup \{(e, \beta\tau)\} = (c \bigcup \{(e, \alpha\beta\tau)\}) - \epsilon/\alpha = c_1 - \epsilon/\alpha$.

   So by Lemma 13, c'_1 is a cut in $M_{\beta\tau}$. $(\beta, c') \xrightarrow{e} (\beta\tau, c'_1)$ and as $Ex(\alpha\beta\tau, \epsilon, \alpha, c_1)$ holds, so, we have $(\alpha\beta\tau, c_1) \rightsquigarrow_1 (\beta\tau, c'_1)$.

3. Case (T3): $(\alpha\tau_1 u\tau_3, c) \xrightarrow{e} (\alpha\pi', c_1) = \lfloor (\alpha\tau_1 u\tau_2 u\tau_3, d) \rfloor_{lue}$, $c + \alpha\tau_1 u/\tau_2 u$ is a cut in $M_{\alpha\tau_1 u\tau_2 u\tau_3}$ and $d = (c + \alpha\tau_1 u/\tau_2 u) \bigcup \{(e, \alpha\tau_1 u\rho_1)\}$ for some non-empty and loop free ρ_1 and loop free ρ_2 such that $\tau_2 u = \rho_1\rho_2$.

   $c' + \tau_1 u/\tau_2 u = (c - \epsilon/\alpha) + \tau_1 u/\tau_2 u = (c + \alpha\tau_1 u/\tau_2 u) - \epsilon/\alpha$ by Lemma 11.

   Clearly, $Ex(\alpha\tau_1 u\tau_2 u\tau_3, \epsilon, \alpha, (c + \alpha\tau_1 u/\tau_2 u))$ holds. Since $c + \alpha\tau_1 u/\tau_2 u$ is a cut in $M_{\alpha\tau_1 u\tau_2 u\tau_3}$ so by Lemma 13, $c' + \tau_1 u/\tau_2 u$ is a cut in $M_{\tau_1 u\tau_2 u\tau_3}$.

   We set c'_1 to $(c' + \tau_1 u/\tau_2 u) \bigcup \{(e, \tau_1 u\rho_1)\}$. So,

   $c'_1 = ((c - \epsilon/\alpha) + \tau_1 u/\tau_2 u) \bigcup \{(e, \tau_1 u\rho_1)\}$

   $= ((c + \alpha\tau_1 u/\tau_2 u) - \epsilon/\alpha) \bigcup \{(e, \tau_1 u\rho_1)\}$ by Lemma 11.

   $= ((c + \alpha\tau_1 u/\tau_2 u) \bigcup \{(e, \alpha\tau_1 u\rho_1)\}) - \epsilon/\alpha = c_1 - \epsilon/\alpha$.

   So, by Lemma 13, c'_1 is a cut in $M_{\tau_1 u\tau_2 u\tau_3}$, and as it satisfies the conditions for transition of type T3, $(\tau_1 u\tau_3, c') \rightsquigarrow_1 lue(\tau_1 u\tau_2 u\tau_3, c'_1) = \lfloor (\tau_1 u\tau_2 u\tau_3, d - \epsilon/\alpha) \rfloor_{lue} = (\pi', c_1 - \epsilon/\alpha)$. The $\rightsquigarrow_1$ conditions are also preserved.

Now we show that $(\beta, c') \xrightarrow{e} (\beta_1, c'_1)$ then $(\alpha\beta, c) \xrightarrow{e} (\alpha\beta_1, c_1)$ and $(\alpha\beta', c') \rightsquigarrow_1 (\beta_1, c'_1)$. Since $(\alpha\beta, c) \rightsquigarrow_1 (\beta, c')$ so, $Ex(\alpha\beta, \epsilon, \alpha, c)$ holds. We denote c in terms of c' as $c = (c' + \epsilon/\alpha) \bigcup E_\alpha$. Following cases have to be considered:

1. Case (T1): $(\beta, c') \xrightarrow{e} (\beta, c'_1)$ where $c'_1 = c' \bigcup \{(e, \rho)\}$ for $\rho \preceq_{ne} \beta$: We set c_1 to $c \bigcup \{(e, \alpha\rho)\}$.

   Then, $c_1 = c \bigcup \{(e, \alpha\rho)\}$

   $= ((c' + \epsilon/\alpha) \bigcup E_\alpha) \bigcup \{(e, \alpha\rho)\}$

   $= ((c' \bigcup \{(e, \rho)\}) + \epsilon/\alpha) \bigcup E_\alpha)$

   $= ((c'_1 + \epsilon/\alpha) \bigcup E_\alpha)$

   Clearly, $Ex(\alpha\beta, \epsilon, \alpha, c_1)$ holds. So, by Lemma 14, $(\alpha\beta, c_1)$ is a configuration in G and structurally, $(\alpha\beta, c_1) \rightsquigarrow_1 (\beta, c'_1)$. Also, $(\alpha\beta, c) \xrightarrow{e} (\alpha\beta, c_1)$ following (T1).

2. Case (T2): $(\beta, c') \xrightarrow{e} (\beta\tau, c'_1)$ where $c'_1 = c' \bigcup \{(e, \beta\tau)\}$ for some non empty τ. We set c_1 to $c \bigcup \{(e, \alpha\beta\tau)\}$.

   Then, $c_1 = ((c' + \epsilon/\alpha) \bigcup E_\alpha) \bigcup \{(e, \alpha\beta\tau)\}$

   $= ((c' \bigcup \{(e, \beta\tau)\}) + \epsilon/\alpha) \bigcup E_\alpha)$

   $= ((c'_1 + \epsilon/\alpha) \bigcup E_\alpha)$.

   Clearly, $Ex(\alpha\beta\tau, \epsilon, \alpha, c_1)$ holds. So, by Lemma 14, $(\alpha\beta\tau, c_1)$ is a configuration in G and structurally $(\alpha\beta\tau, c_1) \rightsquigarrow_1 (\beta\tau, c'_1)$ Also, $(\alpha\beta, c) \xrightarrow{e} (\alpha\beta\tau, c_1)$ following (T2).

3. Case (T3): $(\tau_1 u\tau_3, c') \xrightarrow{e} (\pi', c_1') = \lfloor(\tau_1 u\tau_2 u\tau_3, d)\rfloor_{lue}$, $c' + \tau_1 u/\tau_2 u$ is a cut in $M_{\tau_1 u\tau_2 u\tau_3}$ and $d = (c' + \tau_1 u/\tau_2 u) \bigcup \{(e, \tau_1 u\rho_1)\}$ for some non-empty and loop free ρ_1 and loop free ρ_2 such that $\tau_2 u = \rho_1\rho_2$.

   Firstly, $c + \alpha\tau_1 u/\tau_2 u = ((c' + \epsilon/\alpha) \bigcup E_\alpha) + \alpha\tau_1 u/\tau_2 u$

   $= ((c' + \tau_1 u/\tau_2 u) + \epsilon/\alpha) \bigcup E_\alpha)$ by Lemma 12

   Clearly, $Ex(\alpha\tau_1 u\tau_2 u\tau_3, \epsilon, \alpha, c + \alpha\tau_1 u/\tau_2 u)$ holds. So, by Lemma 14, $c + \alpha\tau_1 u/\tau_2 u$ is a cut in $M_{\alpha\tau_1 u\tau_2 u\tau_3}$. Similarly, as $(c' + \tau_1 u/\tau_2 u) \bigcup \{(e, \tau_1 u\rho_1)\}$ is a cut in $M_{\tau_1 u\tau_2 u\tau_3}$, we can show that

   $d_1 = (c + \alpha\tau_1 u/\tau_2 u) \bigcup \{(e, \alpha\tau_1 u\rho_1)\}$ is a cut in $M_{\alpha\tau_1 u\tau_2 u\tau_3}$.

   So, by transition rule (T3), we can do

   $(\alpha\tau_1 u\tau_3, c) \xrightarrow{e} \lfloor(\alpha\tau_1 u\tau_2 u\tau_3, d_1)\rfloor_{lue} = (\alpha\pi', c_1)$.

   It can be noted that α is completely traversed in d_1. So,

   $\lfloor(\alpha\tau_1 u\tau_2 u\tau_3, d1)\rfloor_{lue}$

   $= (\alpha\pi', (c_1' + \epsilon/\alpha) \bigcup E_\alpha)$.

   So, clearly, $(\alpha\pi', c_1) \rightsquigarrow_1 (\pi', c_1')$.

So, we have also proved that if $(\beta, c') \xrightarrow{e} (\beta_1, c_1')$ then $(\alpha\beta, c) \xrightarrow{e} (\alpha\beta_1, c_1)$ and $(\alpha\beta', c') \rightsquigarrow_1 (\beta_1, c_1')$.

This completes the proof that $\rightsquigarrow_1$ is a bisimilar relation on states of $\mathcal{T}_G$.

We proceed to present two lemmas verifying that the downward closure property of cuts is met by deleting or adding a fully traversed loop to the corresponding configuration.

Lemma 15 *Let $(\alpha u\beta u\gamma, c)$ be a configuration in G. Then, $(\alpha u\gamma, c')$ is a configuration in G, where, $c' = c - \alpha/u\beta$.*

Proof Assume c is not a cut in $M_{\alpha u\gamma}$. This means, $\exists(e, \rho) \in c'$ and $\exists(e', \rho')$ along $\alpha u\gamma$ such that $(e', \rho') <_{\alpha u\gamma} (e, \rho)$ but $(e', \rho') \notin c'$. This contradicts that c is a cut in $M_{\alpha u\beta u\gamma}$ as $(e, \rho + \alpha/u\beta) \in c$, $(e', \rho' + \alpha/u\beta) \notin c$ although clearly $(e', \rho' + \alpha/u\beta) <_{\alpha u\beta u\gamma} (e, \rho + \alpha/u\beta)$. So, c' is a cut in $M_{\alpha u\beta}$. □

Lemma 16 *Let $(\alpha u\gamma, c')$ be a configuration in G and let $c = (c' + \alpha/u\beta) \bigcup (E_{\alpha u\beta - E_\alpha})$. If $Ex(\alpha u\beta u\gamma, \alpha, u\beta, c)$ holds then $(\alpha u\beta u\gamma, \alpha, c)$ is a configuration in G.*

Proof Assume that c is not a cut in $M_{\alpha u\beta u\gamma}$. Since c' is a cut in $M_{\alpha u\gamma}$ and $(E_{\alpha u\beta - E_\alpha}) \subseteq c$, so, this means, $\exists(e, \alpha\rho) \in c$ for $\rho \preceq_{ne} u\beta$ and $\exists(e', \rho') <_{\alpha u\beta u\gamma} (e, \rho)$ for $\rho' \preceq_{ne} \alpha$ such that $(e', \rho') \notin c$. Let $e \in E_p$ for some process p. As $Ex(\alpha u\beta u\gamma, \alpha, u\beta, c)$ holds, this means, $\exists e'' \in E_p$ such that $(e'', \alpha u\beta\rho'') \in c$ for some $\rho'' \preceq_{ne} u\gamma$. So, $(e'', \alpha\rho'') \in c'$ but $(e', \rho') \in c'$ although clearly, $(e', \rho') <_{\alpha u\gamma} (e'', \alpha\rho'')$. Then, c is not a cut in $M_{\alpha u\gamma}$ and so this is not possible. □

We introduce another relation $\rightsquigarrow_2$ relating configurations of $\mathcal{T}_G$ based on deletion of completely traversed loops.

We have $(\alpha u\beta u\gamma, c) \rightsquigarrow_2 (\alpha u\gamma, c')$ provided

- $Ex(\alpha u\beta u\gamma, \alpha, u\beta, c)$ holds, and
- $c' = c - \alpha/u\beta$.

We now prove that $\rightsquigarrow_2$ is a bisimilarity relation on the states of $\mathcal{T}_G$. We assume that this is true for two states $(\alpha u\beta u\gamma, c)$ and $(\alpha u\gamma, c')$ such that $(\alpha u\beta u\gamma, c) \rightsquigarrow_2 (\alpha u\gamma, c')$ and show that:

1. If $(\alpha u\beta u\gamma, c) \xrightarrow{e} (\alpha_1, c_1)$ then $(\alpha u\gamma, c') \xrightarrow{e} (\alpha_1', c_1')$ and $(\alpha_1, c_1) \rightsquigarrow_2 (\alpha_1', c_1')$.
2. If $(\alpha u\gamma, c') \xrightarrow{e} (\alpha_1', c_1')$ then $(\alpha u\beta u\gamma, c) \xrightarrow{e} (\alpha_1, c_1)$ and $(\alpha_1, c_1) \rightsquigarrow_2 (\alpha_1', c_1')$.

Before the main proof, we prove another interesting property about fully traversed loop.

Lemma 17 *Let $(\alpha u\beta u\gamma, c)$ be a configuration in G such that $Ex(\alpha u\beta u\gamma, \alpha, u\beta, c)$ holds. Then, there will be some executions in the later node u.*

Proof As $Ex(\alpha u\beta u\gamma, \alpha, u\beta, c)$ this means, whichever process has reacted in $u\beta$ has reacted in $u\gamma$ as well. So, $\exists(e, \alpha u) \in c$ and $\exists(e', \alpha u\beta\rho) \in c$ where $\rho \preceq_{ne} u\gamma$ such that $(e, \alpha u) <_{\alpha u\beta u\gamma} (e', \alpha u\beta\rho)$. If there is no execution in the later node u, then, $(e, \alpha u\beta u) \notin c$ although clearly, $(e, \alpha u\beta u) <_{\alpha u\beta u\gamma} (e', \alpha u\beta\rho)$. This violates that $(\alpha u\beta u\gamma, c)$ be a configuration in G as c is not downward closed wrt $<_{\alpha u\beta u\gamma}$. □

Now, we prove that if $(\alpha u\beta u\gamma, c) \xrightarrow{e} (\alpha_1, c_1)$ then $(\alpha u\gamma, c') \xrightarrow{e} (\alpha_1', c_1')$ and $(\alpha_1, c_1) \rightsquigarrow_2 (\alpha_1', c_1')$. We analyze the following types of transitions:

1. Case (T1): $(\alpha u\beta u\gamma, c) \xrightarrow{e} (\alpha u\beta u\gamma, c_1)$ where $c_1 = c \bigcup \{(e, \rho)\}$ for some $\rho \preceq_{ne} \alpha u\beta u\gamma$: Since $Ex(\alpha u\beta u\gamma, \alpha, u\beta, c)$ holds wrt c so it is not possible that $\alpha \prec \rho \preceq \alpha u\beta$. We set c_1' to $c' \bigcup (e, \rho - \alpha/u\beta)$. Then,
   $$\begin{aligned} c_1' &= (c - \alpha/u\beta) \bigcup \{(e, \rho - \alpha/u\beta)\} \\ &= (c \bigcup \{(e, \rho)\}) - \alpha/u\beta \\ &= c_1 - \alpha/u\beta. \end{aligned}$$
   Since $c_1 = c \bigcup \{(e, \rho)\}$ is a cut in $M_{\alpha u\beta u\gamma}$ so by Lemma 15, c_1' is a cut in $M_{\alpha u\gamma}$. $(\alpha u\gamma, c') \xrightarrow{e} (\alpha u\gamma, c_1')$ and as $Ex(\alpha u\beta u\gamma, \alpha, u\beta, c_1)$ so, $(\alpha u\beta u\gamma, c_1) \rightsquigarrow_2 (\alpha u\gamma, c_1')$.
2. Case (T2): $(\alpha u\beta u\gamma, c) \xrightarrow{e} (\alpha u\beta u\gamma\tau, c_1)$ where $c_1 = c \bigcup \{(e, \alpha u\beta u\gamma\tau)\}$ for non-empty τ: We set c_1' to $c' \bigcup (e, \alpha u\gamma\tau)$. Then,
   $$\begin{aligned} c_1' &= (c - \alpha/u\beta) \bigcup \{(e, \rho - \alpha/u\gamma\tau)\} \\ &= (c \bigcup \{(e, \alpha u\beta u\gamma\tau)\}) - \alpha/u\beta \\ &= c_1 - \alpha/u\beta \end{aligned}$$
   So by Lemma 15, c_1' is a cut in $M_{\alpha u\gamma\tau}$. So, $(\alpha u\gamma, c') \xrightarrow{e} (\alpha u\gamma\tau, c_1')$ and as $Ex(\alpha u\beta u\gamma\tau, \alpha, u\beta, c_1)$ so $(\alpha u\beta u\gamma\tau, c_1) \rightsquigarrow_2 (\alpha u\gamma\tau, c_1')$.
3. Case (T3): $(\alpha u\beta u\gamma, c) \xrightarrow{e} \lfloor (\alpha_1 v\alpha_2 v\alpha_3 u\beta u\gamma, c_1) \rfloor_{lue}$, where $\alpha = \alpha_1 v\alpha_3$, $c + \alpha_1 v/\alpha_2 v$ is a cut in $M_{\alpha_1 v\alpha_2 v\alpha_3 u\beta u\gamma}$ and $c_1 = (c + \alpha_1 v/\alpha_2 v) \bigcup \{(e, \alpha_1 v\rho_1)\}$ for some non-empty and loop free ρ_1 and loop free ρ_2 such that $\alpha_2 v = \rho_1\rho_2$.

We first show $c' + \alpha_1 v/\alpha_2 v$ is a cut in $M_{\alpha_1 v\alpha_2 v\alpha_3 u\gamma}$. We know

$c' + \alpha_1 v/\alpha_2 v = (c - \alpha_1 v\alpha_3/u\beta) + \alpha_1 v/\alpha_2 v$

$= (c + \alpha_1 v/\alpha_2 v) - \alpha_1 v\alpha_2 v\alpha_3/u\beta$ by Lemma:11.

Since $c+\alpha_1 v/\alpha_2 v$ is a cut in $M_{\alpha_1 v\alpha_2 v\alpha_3 u\beta u\gamma}$, so by Lemma 15, $c'+\alpha_1 v/\alpha_2 v$ is a cut in $M_{\alpha_1 v\alpha_2 v\alpha_3 u\gamma}$.

We set c_1' to $(c' + \alpha_1 v/\alpha_2 v) \bigcup \{(e, \alpha_1 v\rho_1)\}$.

Then, $c_1' = ((c + \alpha_1 v/\alpha_2 v) - \alpha_1 v\alpha_2 v\alpha_3/u\beta) \bigcup \{(e, \alpha_1 v\rho 1)\}$

$= ((c + \alpha_1 v/\alpha_2 v) \bigcup \{(e, \alpha_1 v\rho 1)\} - \alpha_1 v\alpha_2 v\alpha_3/u\beta)$ as $\rho_1 \preceq \alpha_2 v$

$= c_1 - \alpha_1 v\alpha_2 v\alpha_3/u\beta$.

As c_1 is a cut in $M_{\alpha_1 v\alpha_2 v\alpha_3 u\beta u\gamma}$, so by Lemma 15, c_1' is a cut in $M_{\alpha_1 v\alpha_2 v\alpha_3 u\gamma}$. So, by transition rule (T3), we can have

$(\alpha_1 v\alpha_3 u\gamma, c') \xrightarrow{e} \lfloor(\alpha_1 v\alpha_2 v\alpha_3 u\gamma, c_1')\rfloor_{lue}$.

Clearly, $Ex(\alpha_1 v\alpha_2 v\alpha_3 u\beta u\gamma, \alpha_1 v\alpha_2 v\alpha_3, u\beta, c_1)$ holds. So, there can't be any unexecuted portions in $u\beta$ portion. Also by Lemma 17, there is some execution in later node u. Let

c_{11} be $\{(e, \rho) \in c | \rho \preceq_{ne} \alpha_1 v\alpha_2 v\alpha_3$ and $\lfloor(\alpha_1 v\alpha_2 v\alpha_3, c_{11})\rfloor_{lue} = (\tau_1, d_1)$.

c_{12} be $\{(e, \alpha_1 v\alpha_2 v\alpha_3\rho) \in c | \rho \preceq_{ne} u\beta\}$

c_{13} be $\{(e, \alpha_1 v\alpha_2 v\alpha_3 u\beta\rho) \in c | \rho \preceq_{ne} u\gamma\}$ and $\lfloor(u\gamma, c_{13})\rfloor_{lue} = (u\tau_2, d_2)$.

So, by applying Lemma 10 we can say

$\lfloor(\alpha_1 v\alpha_2 v\alpha_3 u\beta u\gamma, c_1)\rfloor_{lue} = (\tau_1 u\beta u\tau_2, d_1) \bigcup (c_{12} + \epsilon/\tau_1) \bigcup (d_2 + \epsilon/\tau_1 u\beta)$.

Applying similar arguments, $\lfloor(\alpha_1 v\alpha_2 v\alpha_3 u\gamma, c_1')\rfloor_{lue}$

$= (\tau_1 u\tau_2, d_1 \bigcup (d_2 + \epsilon/\tau_1))$

$= (\tau_1 u\tau_2, (d_1 \bigcup (c_{12} + \epsilon/\tau_1) \bigcup (d_2 + \epsilon/\tau_1 u\beta)) - \tau_1/u\beta)$.

So, $\lfloor(\alpha_1 v\alpha_2 v\alpha_3 u\beta u\gamma, c_1)\rfloor_{lue} \rightsquigarrow_2 \lfloor(\alpha_1 v\alpha_2 v\alpha_3 u\gamma, c_1')\rfloor_{lue}$.

4. Case (T3): $(\alpha u\beta u, c) \xrightarrow{e} \lfloor(\alpha\beta_1 v\beta_2 v\beta_3 u\gamma, c_1)\rfloor_{lue}$, where $u\beta = \beta_1 v\beta_3$, $c + \alpha\beta_1 v/\beta_2 v$ is a cut in $M_{\alpha\beta_1 v\beta_2 v\beta_3 u\gamma}$ and $c_1 = (c+\alpha\beta_1 v/\beta_2 v) \bigcup \{(e, \alpha\beta_1 v\rho_1)\}$ for some non-empty and loop free ρ_1 and loop free ρ_2 such that $\beta_2 v = \rho_1\rho_2$. We show that this is not possible.

   $Ex(\alpha\beta_1 v\beta_3 u\gamma, \alpha, \beta_1 v\beta_3, c)$ holds. So, $\exists p \in P$ and events $e_1, e_2 \in E_p$, such that $(e_1, \alpha\beta_1 v) \in c$ and $(e_2, \alpha\beta_1 v\beta_3\rho) \in c$ where $\rho \preceq_{ne} u\gamma$. $(e_1, \alpha\beta_1 v\beta_2 v) \notin c + \alpha\beta_1 v/\beta_2 v$ although $(e_2, \alpha\beta_1 v\beta_2 v\beta_3\rho) \in c + \alpha\beta_1 v/\beta_2 v$.

   But $(e_1, \alpha\beta_1 v\beta_2 v) <_{\alpha\beta_1 v\beta_2 v\beta_3 u\gamma} (e_2, \alpha\beta_1 v\beta_2 v\beta_3\rho)$. So, $c + \alpha\beta_1 v/\beta_2 v$ can not be a cut in $M_{\alpha\beta_1 v\beta_2 v\beta_3 u\gamma}$.

   So, this transition is not possible.

5. Case (T3): $(\alpha u\beta u\gamma, c) \xrightarrow{e} \lfloor(\alpha u\beta\gamma_1 v\gamma_2 v\gamma_3, c_1)\rfloor_{lue}$ where $u\gamma = \gamma_1 v\gamma_3$, $c + \alpha u\beta\gamma_1 v/\gamma_2 v$ is a cut in $M_{\alpha u\beta\gamma_1 v\gamma_2 v\gamma_3}$ and $c_1 = (c+\alpha u\beta\gamma_1 v/\gamma_2 v) \bigcup \{(e, \alpha u\beta\gamma_1 v\rho_1)\}$ for some non-empty and loop free ρ_1 and loop free ρ_2 such that $\gamma_2 v = \rho_1\rho_2$: Like in case 3, we can prove that $(\alpha u\gamma, c') \xrightarrow{e} \lfloor(\alpha u\gamma_1 v\gamma_2 v\gamma_3, c_1')\rfloor_{lue}$ where $c_1' = c_1 - \alpha/u\beta$ and $\lfloor(\alpha u\beta\gamma_1 v\gamma_2 v\gamma_3, c_1)\rfloor_{lue} \rightsquigarrow_2 \lfloor(\alpha u\gamma_1 v\gamma_2 v\gamma_3, c_1')\rfloor_{lue}$.

We now show that if $(\alpha u\gamma, c') \xrightarrow{e} (\alpha'_1, c'_1)$ then $(\alpha u\beta u\gamma, c) \xrightarrow{e} (\alpha_1, c_1)$ and $(\alpha_1, c_1) \rightsquigarrow_2 (\alpha'_1, c'_1)$. By our assumption, $Ex(\alpha u\beta u\gamma, \alpha, u\beta, c)$ holds and $c = (c'+\alpha/u\beta)\bigcup(E_{\alpha u\beta}-E_\alpha)$. We prove the statement by going through the possible types of transitions:

1. Case (T1): $(\alpha u\gamma, c') \xrightarrow{e} (\alpha u\gamma, c'_1)$ where $c'_1 = c'\bigcup\{(e,\rho)\}$ for some $\rho \preceq_{ne} \alpha u\gamma$. We set $c_1 = c\bigcup\{(e, \rho+\alpha/u\beta)\}$. Then,

   $$\begin{aligned} c_1 &= ((c'+\alpha/u\beta)\bigcup(E_{\alpha u\beta}-E_\alpha))\bigcup\{(e,\rho+\alpha/u\beta)\} \\ &= ((c'\bigcup\{(e,\rho)\})+\alpha/u\beta)\bigcup(E_{\alpha u\beta}-E_\alpha) \\ &= (c'_1+\alpha/u\beta)\bigcup(E_{\alpha u\beta}-E_\alpha). \end{aligned}$$

   Since $Ex(\alpha u\beta u\gamma, \alpha, u\beta, c)$ holds so clearly $Ex(\alpha u\beta u\gamma, \alpha, u\beta, c_1)$ holds too. As c'_1 is a cut in $M_{\alpha u\gamma}$, so by Lemma 16, c_1 is a cut in $M_{\alpha u\beta u\gamma}$. So, structurally $(\alpha u\beta u\gamma, c_1) \rightsquigarrow_2 (\alpha u\gamma, c'_1)$ and by transition rule (T1), $(\alpha u\beta u\gamma, c) \xrightarrow{e} (\alpha u\beta u\gamma, c_1)$.

2. Case (T2):$(\alpha u\gamma, c') \xrightarrow{e} (\alpha u\gamma\tau, c'_1)$ where $c'_1 = c'\bigcup\{(e,\alpha u\gamma\tau)\}$ for non-empty τ: We set c_1 to $c\bigcup\{(e,\alpha u\beta u\gamma\tau)\}$. Then,

   $$\begin{aligned} c_1 &= ((c'+\alpha/u\beta)\bigcup(E_{\alpha u\beta}-E_\alpha))\bigcup\{(e,\alpha u\beta u\gamma\tau)\} \\ &= ((c'\bigcup\{(e,\alpha u\gamma\tau)\})+\alpha/u\beta)\bigcup(E_{\alpha u\beta}-E_\alpha) \\ &= (c'_1+\alpha/u\beta)\bigcup(E_{\alpha u\beta}-E_\alpha) \end{aligned}$$

   Like in last case, we can easily prove that $(\alpha u\beta u\gamma\tau, c_1) \rightsquigarrow_2 (\alpha u\gamma\tau, c'_1)$ and by transition rule (T2) we have $(\alpha u\beta u\gamma, c) \xrightarrow{e} (\alpha u\beta u\gamma\tau, c_1)$.

3. Case (T3): $(\alpha u\gamma, c') \xrightarrow{e} \lfloor(\alpha_1 v\alpha_2 v\alpha_3 u\gamma, c'_1)\rfloor_{lue}$ where $\alpha = \alpha_1 v\alpha_3$, $c' + \alpha_1 v/\alpha_2 v$ is a cut in $M_{\alpha_1 v\alpha_2 v\alpha_3 u\gamma}$ and $c'_1 = (c'+\alpha_1 v/\alpha_2 v)\bigcup\{(e,\alpha_1 v\rho_1)\}$ for some non-empty and loop free ρ_1 and loop free ρ_2 such that $\alpha_2 v = \rho_1\rho_2$:

   Here, $c = (c'+\alpha_1 v\alpha_3/u\beta)\bigcup(E_{\alpha_1 v\alpha_3 u\beta}-E_{\alpha_1 v\alpha_3})$. We first show that $c+\alpha_1 v/\alpha_2 v$ is a cut in $M_{\alpha_1 v\alpha_2 v\alpha_3 u\beta u\gamma}$.

   $$\begin{aligned} &c+\alpha_1 v/\alpha_2 v \\ &\quad= ((c'+\alpha_1 v\alpha_3/u\beta)\bigcup(E_{\alpha_1 v\alpha_3 u\beta}-E_{\alpha_1 v\alpha_3}))+\alpha_1 v/\alpha_2 v \\ &\quad= ((c'+\alpha_1 v\alpha_3/u\beta)+\alpha_1 v/\alpha_2 v)\bigcup(E_{\alpha_1 v\alpha_2 v\alpha_3 u\beta}-E_{\alpha_1 v\alpha_2 v\alpha_3}) \\ &\quad= ((c'+\alpha_1 v/\alpha_2 v)+\alpha_1 v\alpha_2 v\alpha_3/u\beta)\bigcup(E_{\alpha_1 v\alpha_2 v\alpha_3 u\beta}-E_{\alpha_1 v\alpha_2 v\alpha_3}) \text{ By Lemma 12} \end{aligned}$$

   Also, $Ex(\alpha_1 v\alpha_2 v\alpha_3 u\beta u\gamma, \alpha_1 v\alpha_2 v\alpha_3, u\beta, c+\alpha_1 v/\alpha_2 v)$ holds and so, by Lemma 16 $c+\alpha_1 v/\alpha_2 v$ is a cut in $M_{\alpha_1 v\alpha_2 v\alpha_3 u\beta u\gamma}$. We set c_1 to $(c+\alpha_1 v/\alpha_2 v)\bigcup\{(e,\alpha_1 v\rho_1)\}$. Then,

   $$\begin{aligned} c_1 &= (((c'+\alpha_1 v/\alpha_2 v)+\alpha_1 v\alpha_2 v\alpha_3/u\beta) \\ &\quad\bigcup(E_{\alpha_1 v\alpha_2 v\alpha_3 u\beta}-E_{\alpha_1 v\alpha_2 v\alpha_3})) \\ &\quad\bigcup\{(e,\alpha_1 v\rho_1)\} \\ &= (((c'+\alpha_1 v/\alpha_2 v)\bigcup\{(e,\alpha_1 v\rho_1)\})+\alpha_1 v\alpha_2 v\alpha_3/u\beta) \\ &\quad\bigcup(E_{\alpha_1 v\alpha_2 v\alpha_3 u\beta}-E_{\alpha_1 v\alpha_2 v\alpha_3}) \text{ as } \rho_1 \preceq \alpha_2 v \\ &= (c'_1+\alpha_1 v\alpha_2 v\alpha_3/u\beta)\bigcup E_{\alpha_1 v\alpha_2 v\alpha_3}) \text{ as } \rho_1 \preceq \alpha_2 v \end{aligned}$$

   Since c'_1 is a cut in $M_{\alpha_1 v\alpha_2 v\alpha_3 u\gamma}$ and $Ex(\alpha_1 v\alpha_2 v\alpha_3 u\beta u\gamma, \alpha_1 v\alpha_2 v\alpha_3, u\beta, c_1)$ holds so, by Lemma 16 c_1 is a cut along $M_{\alpha_1 v\alpha_2 v\alpha_3 u\beta u\gamma}$. Following

(T3), we have $(\alpha_1 v \alpha_3 u \beta u \gamma, c) \xrightarrow{e} \lfloor \alpha_1 v \alpha_2 v \alpha_3 u \beta u \gamma \rfloor_{lue}$. Applying similar arguments as in case 3 of the forward direction, we can show that $\lfloor \alpha_1 v \alpha_2 v \alpha_3 u \beta u \gamma, c_1 \rfloor_{lue} \rightsquigarrow_2 \lfloor (\alpha_1 v \alpha_2 v \alpha_3 u \gamma, c_1') \rfloor_{lue}$.

4. Case (T3): $(\alpha u \gamma, c') \xrightarrow{e} \lfloor (\alpha \gamma_1 v \gamma_2 v \gamma_3, c_1') \rfloor_{lue}$ where $u\gamma = \gamma_1 v \gamma_3$, $c' + \alpha\gamma_1 v/\gamma_2 v$ is a cut in $M_{\alpha\gamma_1 v\gamma_2 v\gamma_3}$ and $c_1' = (c' + \alpha\gamma_1 v/\gamma_2 v) \bigcup \{(e, \alpha\gamma_1 v \rho_1)\}$ for some non-empty and loop free ρ_1 and loop free ρ_2 such that $\gamma_2 v = \rho_1 \rho_2$:

   Like in case 3, we can show that in this case also
   $(\alpha u \beta u \gamma, c) \xrightarrow{e} \lfloor (\alpha u \beta \gamma_1 v \gamma_2 v \gamma_3, c_1) \rfloor_{lue}$
   where $c_1 = (c + \alpha u \beta \gamma_1 v/\gamma_2 v) \bigcup \{(e, \alpha u \beta \gamma_1 v \rho_1)\}$ and
   $\lfloor (\alpha u \beta \gamma_1 v \gamma_2 v \gamma_3, c_1) \rfloor_{lue} \rightsquigarrow_2 \lfloor (\alpha \gamma_1 v \gamma_2 v \gamma_3, c_1') \rfloor_{lue}$.

This completes the proof that $\rightsquigarrow_2$ is a bisimilarity relation on the states of $\mathcal{T}_G$.

We summarize the bisimilarity discussion in the following lemma:

Lemma 18 *The relations $\rightsquigarrow_1$ and $\rightsquigarrow_2$ are bimulation relations on the transition system $\mathcal{T}_G$.* □

We will now reduce the state space of $\mathcal{T}_G$ using the bisimulation relations defined above.

For a configuration (π, c) of G, we define its *maximal reducible decomposition* to be $\alpha\alpha_1 u_1 \beta_1 u_1 \cdots \alpha_n u_n \beta_n u_n \gamma$ with $n \geq 0$, where

- $\pi = \alpha\alpha_1 u_1 \beta_1 u_1 \cdots \alpha_n u_n \beta_n u_n \gamma$.
- α is the maximal prefix of π which is completely traversed.
- Each $u_i \beta_i$ is the leftmost completely traversed loop in the segment $\alpha_i u_i \beta_i u_i \cdots \alpha_n u_n \beta_n u_n \gamma$.
- Each $u_i \beta_i$ is maximal.
- γ does not have any completely traversed loop.

The maximal reducible decomposition of (π, c) can be seen to be unique.

Let $\alpha\alpha_1 u_1 \beta_1 u_1 \cdots \alpha_n u_n \beta_n u_n \gamma$ be the maximal reducible decomposition of (π, c). Then we define the *reduced form* of (π, c), denoted $\lfloor (\pi, c) \rfloor$, to be the configuration

$$\begin{aligned}(\alpha_1 u_1 \quad & \cdots \alpha_n u_n \gamma, \\ & (\cdots((c - \epsilon/\alpha) - \alpha_1 u_1 \beta_1) - \cdots) - \alpha_1 u_1 \cdots \alpha_n u_n / u_n \beta_n).\end{aligned}$$

Clearly, the reduced form of a configuration does not contain any completely traversed prefix or loops.

We now define the reduced transition system corresponding to the MSG G, denoted $\mathcal{T}_G'$, as follows: $\mathcal{T}_G' = (Q', E_G, q_0, \Rightarrow)$ where

- $Q' = \{\lfloor (\pi, c) \rfloor \mid (\pi, c)$ a configuration of $G\}$.
- E_G is the set of events of G.
- $q_0 = (\epsilon, \emptyset)$.

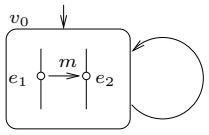


Figure 6: An MSG with an infinite-state $\mathcal{T}'_G$.

- Let $q', r' \in Q'$. Then, $q' \stackrel{e}{\Rightarrow} r'$ iff there exists $r \in Q$ such that $q' \stackrel{e}{\rightarrow} r$ in $\mathcal{T}_G$, and $r' = \lfloor r \rfloor$.

We note that the reduced form of the initial configuration $(\epsilon, \emptyset)$ is itself.

We now want to argue that $\mathcal{T}'_G$ generates the same language as $\mathcal{T}_G$. We have already shown that $\rightsquigarrow_1$ and $\rightsquigarrow_2$ are bisimulation relations on $\mathcal{T}_G$. Hence so is the relation $(\rightsquigarrow_1 \cup \rightsquigarrow_2)^*$. Further, for any configuration (π, c), it is clear that $((\pi, c), \lfloor(\pi, c)\rfloor) \in (\rightsquigarrow_1 \cup \rightsquigarrow_2)^*$. Finally, it is immediate to check that T'_G satisfies the conditions of Lemma 7 with respect to $\mathcal{T}_G$ and the bisimulation relation $\mathcal{R} = (\rightsquigarrow_1 \cup \rightsquigarrow_2)^*$. Hence by Lemma 7 we can conclude that $L(\mathcal{T}'_G) = L(\mathcal{T}_G)$.

We summarise the result of this section in the following theorem:

Theorem 19 *For any MSG G, the reduced transition system $\mathcal{T}'_G$ generates precisely the language $L^e(G)$.*

We note that though $\mathcal{T}'_G$ has fewer states than $\mathcal{T}_G$ in general, it may still have an infinite number of states. Consider the MSG G_2 in Figure 6. The configurations of the form $(v_0^n, \{(e_1, v_0), (e_1, v_0^2), \cdots, (e_1, v_0^n)\})$ where $n > 0$ do not have any unexecuted loops or completely traversed loops. So, for each n, these are distinct states in $\mathcal{T}'_{G_2}$, and thus $\mathcal{T}'_{G_2}$ has an infinite number of states. In the next section we show that this is not possible for a com-connected MSG.

6 Regularity of Com-Connected MSG's

In this section our aim is to supply a proof of Theorem 2. We begin with an observation from [2].

Lemma 20 ([2]) *Let G be a com-connected MSG. Consider a configuration of the form $(\alpha u \beta u \gamma, c)$. Then, either $u\beta$ is completely unexecuted, or $u\beta$ is completely traversed, or there is a process whose last executed event and next unexecuted event are both in $u\beta$.*

Proof Let us assume the contrary. Then among the processes that take part in an event in the nodes in $u\beta$, there must be processes p and q such that *none* of the p events in $u\beta$ are executed, and *all* of the q events in $u\beta$ are executed. As G is com-connected, we must have a path in the communication graph of $M_{u\beta}$ from p to q. Let this path be $p = r_0 \rightarrow r_1 \rightarrow \cdots \rightarrow r_n = q$ with $n \geq 1$. Then, since q has completed all its events in $u\beta$, it must have also received a message from r_{n-1}. Thus r_{n-1} has taken part in $u\beta$, since it must have sent the message received by q. Now either r_{n-1} has another event to take part in in $u\beta$, and we are done; or, it has completed all its events in $u\beta$ and in particular has heard

from r_{n-2}. We repeat this argument till we either find a process r_i which has participated in $u\beta$ and has an unexecuted event in $u\beta$, or reach a contradiction that process p has participated in an event in $u\beta$. This completes the proof of the lemma. □

Let us now consider the reduced transition system $\mathcal{T}'_G$ for a com-connected MSG G. The states in $\mathcal{T}'_G$ are all in reduced form, and hence have no completely unexecuted loops or completely traversed loops. It follows that in any loop in a state (π, c) of $\mathcal{T}'_G$, there must be at least one process positioned in a node in that loop. Thus no node can occur more than $k+1$ times in π, where k is the number of processes. In fact, because of the property of the reachable configurations of $\mathcal{T}_G$ that there is always a process positioned in the last node of the path, the bound of $k+1$ can be reduced to k. This bounds the length of π by mk, where m is the number of nodes in G.

There are $\frac{(m\times k)!}{(k!)^m}$ ways of making strings of size $m \times k$ from m nodes each repeating k times. It is easy to show that this is lower than $m^{m\times k}$. If there are n events per MSC and for each process, then, in each configuration, each processes can have $m \times n \times k$ possible positions. This gives us the upper bound $m^{m\times k} \times (m \times n \times k)^k$ on the number of states in $\mathcal{T}'_G$.

This completes the proof of Theorem 2.

7 Synchronous MSG's

In this section we consider MSG's with synchronous (or "handshake") messages, and show how the transition system for them can be constructed in a similar manner to MSG's with asynchronous messages.

A message sequence chart with *synchronous messages*, or simply a *synchronous MSC* is a tuple

$$M = \langle P, E, C, \lambda, \{<_p\}_{p\in P}\rangle$$

where:

- P is a finite set of processes.
- E is a finite set of events.
- C is a finite set of message labels.

  The set of actions of M is defined to be $\Sigma_M = P \times P \times C$, where (p, q, m) signifies processes p and q synchronously exchanging message m.

  For a process $p \in P$, the set $\Sigma_p = \{a \in \Sigma_M \mid a = (p, q, m) \text{ or } a = (q, p, m),\ q \in P,\ m \in C\}$, is the set of all actions in which p participates.
- $\lambda : E \to \Sigma_M$ is the labeling function which maps events to actions. For a process $p \in P$, the set $E_p = \{e \mid \lambda(e) \in \Sigma_p\}$ are the events in which p participates.
- For each $p \in P$, $<_p$ is a strict total order on E_p.

  We define $<_M$ as the transitive closure, and $\leq_M$ as the reflexive transitive closure, of $\bigcup_{p\in P} <_p$, and require that $\leq_M$ must be a partial-order.

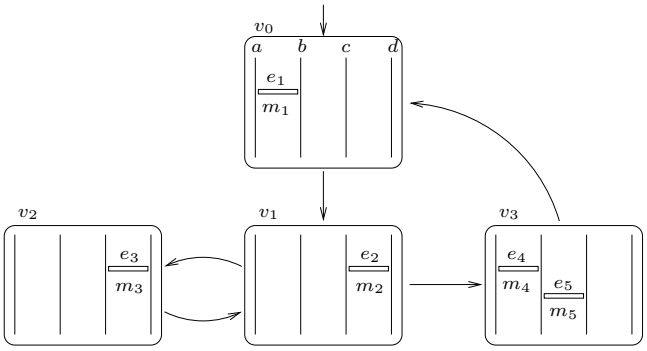


Figure 7: MSG G_4 with synchronous messages.

As for asynchronous MSC's, we define the language of event sequences generated by a synchronous MSC M with event set E, to be the prefixes of linearisations of the partial order $(E, \leq_M)$. A cut in M continues to be a subset of events in E which is closed with respect to $\leq_M$. Analogous to Lemma 1 we can characterise the event sequences of M in terms of sequences of incremental cuts in $(E, \leq_M)$.

The communication graph of a synchronous MSC M is an undirected graph on the set of processes obtained by adding an undirected edge between two processes if they participate in a common event.

A *synchronous* MSG is an MSG $G = (V, v_0, \Delta, \mathcal{M}, \mu)$ as defined earlier for asynchronous MSC's, except that $\mathcal{M}$ is now a set of synchronous MSC's. For a non-empty path π in G, we can define the (weak) concatenation of the synchronous MSC's labelling π, as before to be the (weak) concatenation of the partial orders represented by the MSC's. More formally, for each vertex v in G, let each MSC $\mu(v)$ be $\langle P_v, E_v, C, \lambda_v, \{<_p^v\}_{p \in P}\rangle$. We define the *(weak) concatenation* of the synchronous MSC's in the path π to be the MSC $M_\pi = \langle P, E_\pi, C, \lambda_\pi, \{<_p^\pi\}_{p \in P}\rangle$ where:

- $E_\pi = \bigcup_{\rho v \preceq \pi}(E_v \times \{\rho v\})$.
- For each $\rho v \preceq \pi$, we define $\lambda_\pi(e, \rho v) = \lambda_v(e)$.
- For each $p \in P$, $<_p^\pi$ is given as follows: Let $\rho v \preceq \pi$ and $\rho' v' \preceq \pi$ and $e \in E_v$ and $e' \in E_{v'}$. Then $(e, \rho v) <_p^\pi (e', \rho' v')$ iff e and e' are p-events and either $\rho v \prec \rho' v'$ or $\rho v = \rho' v'$ and $e <_p^v e'$.

The language of event sequences defined by a synchronous MSG G, denoted $L^e(G)$, is defined to be the set of all prefixes of linearizations of M_π for any initial path π in G.

As for asynchronous MSG's, we say a synchronous MSG G is com-connected iff for every loop $u\beta u$ in G, the communication graph of $M_{u\beta}$ is com-connected.

The figure 7 shows a com-connected synchronous MSG G_4.

The construction of the transition systems $\mathcal{T}_G$ and $\mathcal{T}'_G$ for an asynchronous MSG G, is based purely on the partial order induced by a path in the MSG. Hence it also applies for a synchronous MSG G. The proof of correctness of $\mathcal{T}_G$ and $\mathcal{T}'_G$ also make use of properties of the way the partial order induced

by a path in the MSG is defined, and these properties are also satisfied by the definition of this partial order for synchronous MSG's. Hence we can conclude:

Theorem 21 *Let G be a synchronous MSG, and let $\mathcal{T}_G$ and $\mathcal{T}'_G$ be the transition systems defined in Section 3 and 5 respectively. Then both transition systems generate exactly the language $L^e(G)$.*

Further, Lemma 20 can also be seen to apply for synchronous MSG's, and hence we have:

Theorem 22 *Let G be a com-connected synchronous MSG. Then $L^e(G)$ is regular and it can be generated by a finite-state transition system with at most $m^{mk}(mnk)^k$ states.*

Figure 8 is the transition table generated by our algorithm and Figure 9 shows the state diagram of the transition system T'_G for the MSG G_4 of Figure 7.

The MSG of Figure 7 is useful for pointing out the incompleteness of the transition system constructed by Uchitel in [12]. He constructs a transition system, called the "trace model", which is meant to generate the language of event sequences for a given com-connected synchronous MSG. The construction uses a "coordinator" component which keeps track of the current path in the MSG traced out by processes, and ensures that other processes follow this path consistently. The transition rules for the infinite model are similar to ours, except that there is *no rule* for inserting a path within the current node list. The problem arises due to the equivalence relation proposed by him to reduce the infinite model. His equivalence relation removes a loop whenever there are no processes present there. This incorrect reduction essentially leads to the trace model being incomplete.

The trace model generated by the LTSA-MSC tool [11, 13] on the MSG G_4 as input is shown in the figure below.

Source State	Transition		Destination	
	Type	Event	Configuration	State
q_0	T2	e_1	$(v_0, \{(e_1, v_0)\})$	q_1
	T2	e_2	$(v_0v_1, \{(e_2, v_0v_1)\})$	q_2
q_1	T2	e_2	$(v_0v_1, \{(e_1, v_0), (e_2, v_0v_1)\})$	q_3
	T2	e_4	$(v_0v_1v_3, \{(e_1, v_0), (e_4, v_0v_1v_3)\})$ $\rightsquigarrow_1 (v_1v_3, \{(e_4, v_1v_3)\})$	q_4
q_2	T1	e_1	$(v_0v_1, \{(e_1,v_0), (e_2, v_0v_1)\})$	q_3
	T2	e_3	$(v_0v_1v_2, \{(e_2, v_0v_1), (e_3, v_0v_1v_2)\})$	q_5
q_3	T2	e_3	$(v_0v_1v_2, \{(e_1, v_0), (e_2, v_0v_1), (e_3, v_0v_1v_2)\})$	q_6
	T2	e_4	$(v_0v_1v_3, \{(e_1, v_0), (e_2, v_0v_1), (e_4, v_0v_1v_3)\})$ $\rightsquigarrow_1 (v_1v_3, \{(e_2, v_1), (e_4, v_1v_3)\})$	q_7
q_4	T1	e_2	$(v_1v_3, \{(e_2, v_1), (e_4, v_1v_3)\})$	q_7
q_5	T2	e_2	$(v_0v_1v_2v_1, \{(e_2, v_0v_1), (e_3, v_0v_1v_2), (e_2, v_0v_1v_2v_1)\})$ $\rightsquigarrow_2 (v_0v_1, \{(e_2, v_0v_1)\})$	q_2
	T1	e_1	$(v_0v_1v_2, \{(e_1, v_0), (e_2, v_0v_1), (e_3, v_0v_1v_2)\})$	q_6
q_6	T2	e_2	$(v_0v_1v_2v_1, \{(e_1, v_0), (e_2, v_0v_1), (e_3, v_0v_1v_2)$, $(e_2, v_0v_1v_2v_1)\})$ $\rightsquigarrow_2 (v_0v_1, \{(e_1, v_0), (e_2, v_0v_1)\})$	q_3
	T2	e_4	$(v_0v_1v_2v_1v_3, \{(e_1, v_0), (e_2, v_0v_1), (e_3, v_0v_1v_2), (e_4, v_0v_1v_2v_1v_3)\})$ $\rightsquigarrow_1 (v_2v_1v_3, \{(e_3, v_2), (e_4, v_2v_1v_3)\})$	q_8
q_7	T3	e_3	$(v_1v_2v_1v_3, \{(e_2, v_1), (e_3, v_1v_2), (e_4, v_1v_2v_1v_3)\})$ $\rightsquigarrow_1 (v_2v_1v_3, \{(e_3, v_2), (e_4, v_2v_1v_3)\})$	q_9
	T1	e_5	$(v_1v_3, \{(e_2, v_1), (e_4, v_1v_3), (e_5, v_1v_3)\})$	q_{10}
q_8	T1	e_2	$(v_2v_1v_3, \{(e_3, v_2), (e_2, v_2v_1), (e_4, v_2v_1v_3)\})$ $\rightsquigarrow_1 (v_1v_3, \{(e_2, v_1), (e_4, v_1v_3)\})$	q_7
q_9	T1	e_2	$(v_2v_1v_3, \{(e_3, v_2), (e_2, v_2v_1), (e_4, v_2v_1v_3)\})$ $\rightsquigarrow_1 (v_1v_3, \{(e_2, v_1), (e_4, v_1v_3)\})$	q_7
q_{10}	T2	e_1	$(v_1v_3v_0, \{(e_2, v_1), (e_4, v_1v_3), (e_5, v_1v_3), (e_1, v_1v_3v_0)\})$	q_{11}
	T2	e_2	$(v_1v_3v_0v_1, \{(e_2, v_1), (e_4, v_1v_3), (e_5, v_1v_3), (e_2, v_1v_3v_0v_1)\})$ $\rightsquigarrow_1 (v_3v_0v_1, \{(e_4, v_3), (e_5, v_3), (e_2, v_3v_0v_1)\})$	q_{12}
q_{11}	T2	e_4	$(v_1v_3v_0v_1v_3, \{(e_2, v_1), (e_4, v_1v_3), (e_5, v_1v_3), (e_1, v_1v_3v_0), (e_4, v_1v_3v_0\ v_1v_3)\})$	q_{13}
	T2	e_2	$(v_1v_3v_0v_1, \{(e_2, v_1), (e_4, v_1v_3), (e_5, v_1v_3), (e_1, v_1v_3v_0), (e_2, v_1v_3v_0v_1)\})$ $\rightsquigarrow_1 (v_0v_1, \{(e_1, v_0), (e_2, v_0v_1)\})$	q_3
q_{12}	T1	e_1	$(v_3v_0v_1, \{(e_4, v_3), (e_5, v_3), (e_1, v_3v_0), (e_2, v_3v_0v_1)\})$ $\rightsquigarrow_1 (v_0v_1, \{(e_1, v_0), (e_2, v_0v_1)\})$	q_3
q_{13}	T1	e_2	$(v_1v_3v_0v_1v_3, \{(e_2, v_1), (e_4, v_1v_3), (e_5, v_1v_3), (e_1, v_1v_3v_0), (e_2, v_1v_3v_0v_1), (e_4, v_1v_3v_0v_1v_3)\})$ $\rightsquigarrow_1 (v_1v_3, \{(e_2, v_1), (e_4, v_1v_3)\})$	q_7

Figure 8: Transition table $\mathcal{T}'_G$ for MSG G_4.

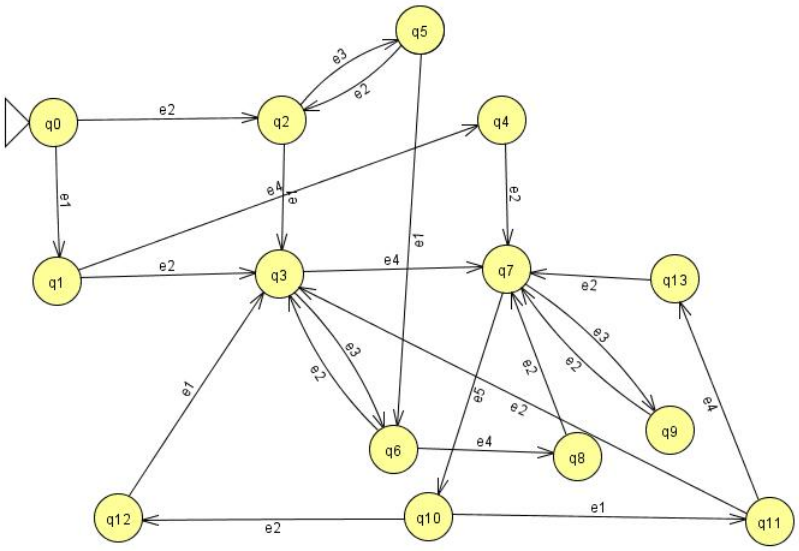
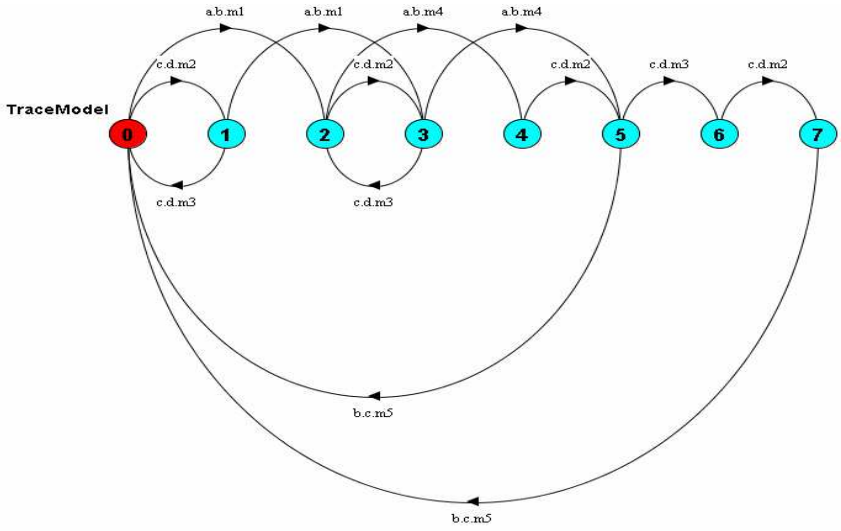
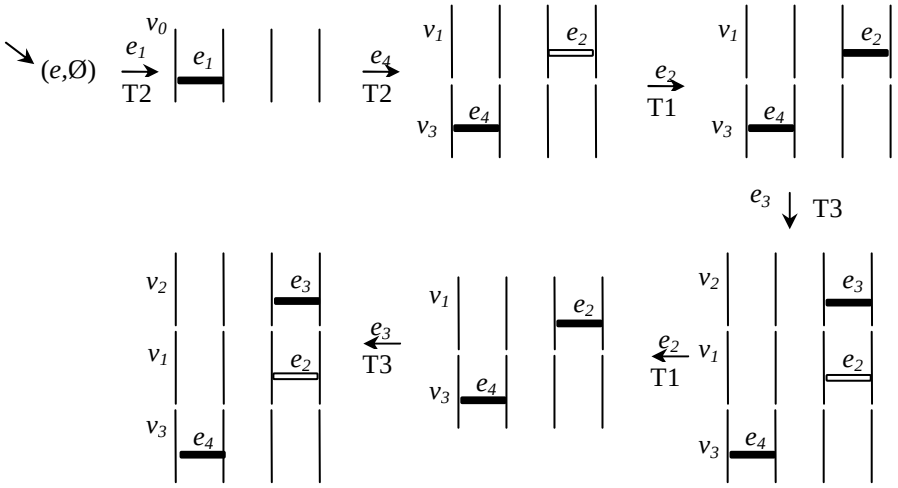


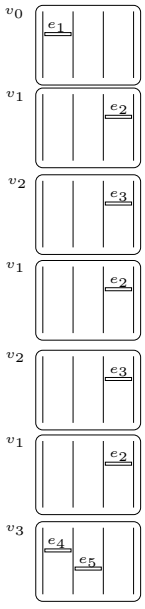
Figure 9: Transition system $\mathcal{T}'_G$ for MSG G_4.



The event sequence $e_1e_4e_2e_3e_2e_3$ is not allowed by the trace model, while it is accepted by T'_G as shown in the figure below.



The event sequence is legal behaviour of MSG G_4 being prefix of a linearisation of M_π, where π is the path $v_0v_1v_2v_1v_2v_1v_3$, shown in the figure below.



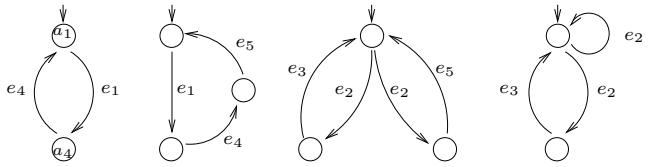


Figure 10: System model induced by MSG G_4.

8 Detecting Implied Scenarios

We now sketch how our construction of $\mathcal{T}'_G$ can be used to detect "implied scenarios" in a com-connected synchronous MSG.

A synchronous MSG G induces a natural "minimal" distributed finite-state system model (called the "architecture model" in [12]). Each process has a component obtained by keeping track of which events it can participate in next. The components for each process in the system model for the MSG G_4 are shown in Figure 10. Thus process a takes part in events e_1 and e_4 and has two states a_1 and a_4 corresponding to these two events. The states a_1 and a_4 can be thought of as the positions in the process line of process a in the MSG G_4, *just before* the events e_1 and e_4. The system model induced by the components, can be viewed as the "synchronised product" of the components for each process, where a pair of components are required to simultaneously execute events that are common to their process lines. The system model, which we denote $\mathcal{S}_G$, can be seen to include *all* the behaviours in $L^e(G)$ for a given synchronous MSG G. The problem is that it may sometimes generate a strict superset of the behaviours in $L^e(G)$, and these behaviours are what are referred to as "implied scenarios." Thus an event sequence $w \in E_G^*$ is called an *implied scenario* if $w \in L(\mathcal{S}_G) - L^e(G)$. We refer the reader to [14, 12] for some illustrative examples of implied scenarios.

We first show that the problem of detecting implied scenarios for *general* MSG's is undecidable. The proof is via a reduction from the Post Correspondence Problem (PCP). Let P be an instance of PCP given by two homomorphisms $g, h : A^* \to B^*$. A solution of P is given by a word w in A^+ such that $g(w) = h(w)$. We make use of the regular language L_P defined in [10] (see also [9]). L_P is defined to be $L_g \cup L_h$, where L_g (similarly L_h) is the complement of $W_g = \{wg(w)\|c^{|g(w)|} \mid w \in A^+\}$. All we need is the fact that L_P satisfies the property that

$$[L_P]_\sim = \begin{cases} (A \cup B)^*\|c^* & \text{if} \quad P \text{ has no solution} \\ \text{a strict subset of } (A \cup B)^*\|c^* & \text{if} \quad P \text{ has a solution} \end{cases}$$

where $[L_P]_\sim$ denotes the trace closure (that is all possible words that can be obtained from words in L_p by commuting letters in $A \cup B$ with the letter c) of the language L_P; and $L\|M$ denotes the "shuffle" of words in L and M defined to be $\{w \mid w \downharpoonright (A \cup B) \in L, w \downharpoonright c \in M\}$. By $w \downharpoonright X$ we mean the word obtained by deleting all occurences of letters outside X from w.

Now let us define an MSG from L_P as follows: Let $\mathcal{A}$ be a finite-state automaton accepting L_P. Without loss of generality we can assume that the states of $\mathcal{A}$ (instead of the transitions) are labelled by letters in $A \cup B \cup \{c\}$.

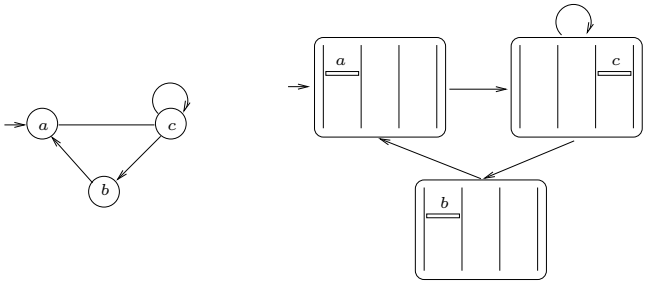


Figure 11: An automaton and the corresponding MSG.

Then we can define an MSG G_P which has the same graph structure as $\mathcal{A}$, and each state v is labelled by the MSC M_a, where a is the label of v, and M_a is the MSC shown below on the left if $a \in (A \cup B)$ or the one on the right if $a = c$.

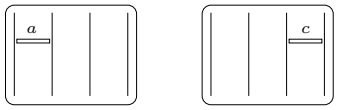


Figure 11 shows the MSG corresponding to the automaton on the left. Here we assume $a, b \in A \cup B$.

It is not difficult to see that the language specified by the MSG G_P is precisely the trace closure of L_P: that is $L^e(G_P) = [L_P]_\sim$. Further, we claim that G_P has implied scenarios iff the PCP problem P has a solution. This is because:

- The language L_P is such that for each $w \in (A \cup B)^*$, and each $c^i \in \{c\}^*$, the strings w and c^i both belong to L_P. Thus, it follows that S_{G_P} will always generate the language $(A \cup B)^* \| c^*$.
- However, the language $L^e(G_P)$ equals $(A \cup B)^* \| c^*$ iff the PCP instance P has no solutions.
- Hence G_P has implied scenarios iff P has a solution.

This completes the reduction.

We note that we have made use of final states in the MSG G_P. However it is not difficult to adapt this reduction to MSG's in which all states are final (which is the definition we have followed in this paper). We can do so as follows.

- We first note that the variant of the PCP problem, called say PCP', in which we ask if there is a solution to the given instance which *begins* with 1 and *ends* with k, and k does not occur anywhere else (where we assume $A = \{1, 2, \ldots, k\}$), is also undecidable. This can be seen from the fact that the classical reduction from the membership problem for Turing Machines to PCP produces such an instance of PCP [6]. Further, clearly without loss of generality, we can assume that the given PCP' instance is such that $g(k)$ and $h(k)$ both end with a '$\dashv$' symbol which is not in $A \cup B$.
- Let L_g (and similarly L_h) be the regular language defined in [10] over $A \cup B \cup \{\dashv\} \cup \{c\}$ corresponding to P as a usual instance of PCP.

- We now restrict L_g by retaining only words that contain 0 or 1 occurrences of $\dashv$. Thus we obtain

$$L'_g = L_g \cap \{w \in (A \cup B \cup \{\dashv\} \cup \{c\})^* \mid w \downarrow \{\dashv\} \in \{\epsilon, \dashv\}\}.$$

  Clearly L'_g is also regular.

- We further define L''_g from L'_g as follows:

$$L''_g = \{w \in L'_g \mid w \downarrow \{\dashv\} = \emptyset\} \cup \{w' \dashv \mid w \in L'_g,\ w \downarrow \{\dashv\} = \{\dashv\},\ w \downarrow (A \cup B \cup \{c\}) = w'\}.$$

  Once again, since L'_g is regular, it follows that L''_g is also regular.

- We now observe that the implied closure of L''_g (and also L''_h) is

$$((A \cup B)^* \cdot \{\dashv, \epsilon\}) \| c^*.$$

  This is because for any $w \in (A \cup B)^* \cdot \{\dashv, \epsilon\}$, and any $c^i \in c^*$, $w \in L''_g$ and $c^i \in L''_g$.

- We note that L''_g and L''_h satisfy

$$[L''_g \cup L''_h]_\sim = \begin{cases} ((A \cup B)^* \cdot \{\dashv, \epsilon\}) \| c^* & \text{if} \quad P \text{ has no solution} \\ \text{a strict subset of } ((A \cup B)^* \cdot \{\dashv, \epsilon\}) \| c^* & \text{if} \quad P \text{ has a solution} \end{cases}$$

  As a matter of fact, if there is a solution then $(AUB)^* \cdot \dashv \| c^*$ is *not* contained in $[L''_g \cup L''_h]_\sim$.

- Finally, Finally, since prefixes of words in L''_g do not help in constructing words in $(AUB)^* \cdot \dashv \| c^*$, it follows that if we replace L''_g by its prefix-closure $pref(L''_g)$, and similarly L''_h by $pref(L''_h)$, we will obtain a prefix-closed regular language, namely $pref(L''_g) \cup pref(L''_h)$ which satisfies all the properties above: namely, that its implied closure is equal to its trace closure iff the given PCP' instance has no solution. This completes the reduction.

Theorem 23 *The problem of detecting implied scenarios for general synchronous MSG's is undecidable.* □

However, when an MSG specification G is given to be com-connected, our finite-state transition system $\mathcal{T}'_G$ we can give a procedure for detecting implied scenarios in G. We simply generate the system model $\mathcal{S}_G$ for G, and the transition system $\mathcal{T}'_G$ for G, and check if $L(\mathcal{S}_G) \subseteq L(\mathcal{T}'_G)$. Note that both $\mathcal{S}_G$ and $\mathcal{T}'_G$ are finite-state transition systems, and hence using standard automata-theoretic techniques we can check this property and report implied scenarios if there are any.

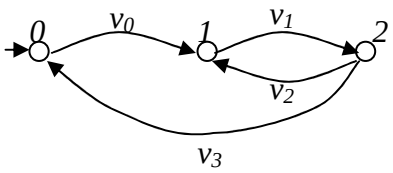
Using our finite-state transition system $\mathcal{T}'_G$ we can give a procedure for detecting implied scenarios in a com-connected synchronous MSG specification G. We simply generate the system model (event) $\mathcal{S}_G$ for G, and the transition system $\mathcal{T}'_G$ for G, and check if $L(\mathcal{S}_G) \subseteq L(\mathcal{T}'_G)$. Note that both $\mathcal{S}_G$ and $\mathcal{T}'_G$ are finite-state transition systems, and hence using standard automata-theoretic

techniques we can check this property and report implied scenarios if there are any.

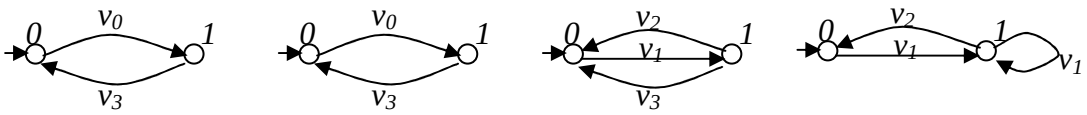
Uchitel uses the same procedure in his thesis [12] but due to the fact that his trace model is incomplete, his detection algorithm is not sound. The MSG G_4 can be seen to have no implied scenarios. However the LTSA-MSC tool reports $e_1e_4e_2e_3e_2e_3$ as one of the several implied scenarios for this MSG.

[14] proposes detection of implied scenarios comparing the system model with the maximal traces of the composition of individual process's view of the MSG. The approach proposed in this paper claims to catch the implied scenarios in MSGs with synchronous messaging, even if the MSG is not bounded. We show that the proposed heuristic approach doesn't work properly with the same exmaple MSG G_4.

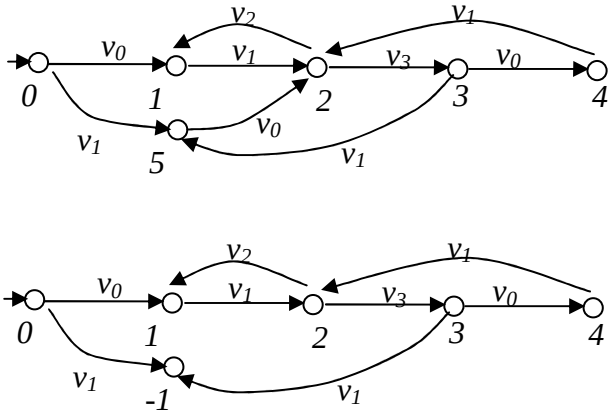
The approach first constructs the MSG as a labeled transition system with each transition labelled with the MSG node being executed. The automaton for G_4 is drawn below.



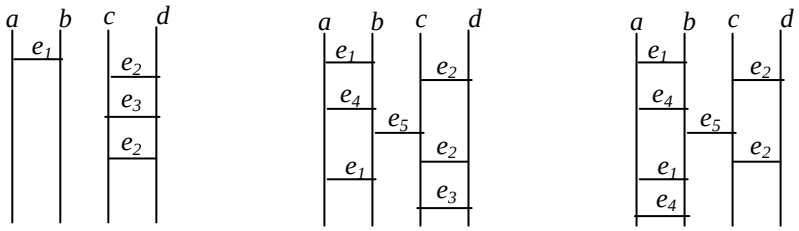
Then each process's local view of the MSG is constructed based on the nodes where each process participates, as shown in the figure below.



The product of the local views of each process generates the composed MSG view. This composed MSG view is compared with the MSG to mark only the allowed transitions. Both the views for G_4 is shown below.



The approach then proceeds to construct the 'safety language' which comprises of the linearizations of MSCs hit till a node is revisited followed by the first message in the next possible MSC as per the MSG. The safety languages generated for G_4 are mentioned in the figure below.



It can be easily seen that the event sequence $e_1e_4e_2e_3e_2e_3$ along $M_{v_0v_1v_2v_1v_2v_1v_3}$ is not prefix of any of the safety strings and so will be wrongly pointed out as implied scenario by this algorithm as well.

9 Conclusion

In this paper we have given a precise construction of a transition system for a given MSG specification, which accepts exactly the set of behaviours specified by the MSG. When the given MSG specification is com-connected, the transition system we give is guaranteed to be finite-state and can thus be used a basis for building sound and complete tools for analysing properties of these specifications. Our transition system is also suitable for analysing MSG specifications in a bounded fashion, even when the given MSG is not com-connected. We hope that work in this paper will be useful to academicians and practitioners

interested in building accurate tools to analyse Message Sequence Chart based specifications.

References

[1] R. Alur, K. Etessami, and M. Yannakakis. Inference of message sequence charts. *IEEE Trans. Software Eng.*, 29(7):623–633, 2003.

[2] R. Alur and M. Yannakakis. Model checking of message sequence charts. In J. C. M. Baeten and S. Mauw, editors, *CONCUR*, volume 1664 of *Lecture Notes in Computer Science*, pages 114–129. Springer, 1999.

[3] M. Clerbout and M. Latteux. Semi-commutations. *Inf. Comput.*, 73(1):59–74, 1987.

[4] B. Genest, A. Muscholl, and D. Peled. Message sequence charts. In J. Desel, W. Reisig, and G. Rozenberg, editors, *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 537–558. Springer, 2003.

[5] J. G. Henriksen, M. Mukund, K. N. Kumar, M. A. Sohoni, and P. S. Thiagarajan. A theory of regular msc languages. *Inf. Comput.*, 202(1):1–38, 2005.

[6] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, Massachusetts, 1979.

[7] H. Muccini. Detecting implied scenarios analyzing non-local branching choices. In M. Pezzè, editor, *FASE*, volume 2621 of *Lecture Notes in Computer Science*, pages 372–386. Springer, 2003.

[8] A. Muscholl and D. Peled. Message sequence graphs and decision problems on mazurkiewicz traces. In M. Kutylowski, L. Pacholski, and T. Wierzbicki, editors, *MFCS*, volume 1672 of *Lecture Notes in Computer Science*, pages 81–91. Springer, 1999.

[9] A. Muscholl and H. Petersen. A note on the commutative closure of star-free languages. *Inf. Process. Lett.*, 57(2):71–74, 1996.

[10] J. Sakarovitch. The "last" decision problem for rational trace languages. In I. Simon, editor, *LATIN*, volume 583 of *Lecture Notes in Computer Science*, pages 460–473. Springer, 1992.

[11] S. Uchitel. LTSA-MSC tool. *http://www.doc.ic.ac.uk/ su2/Synthesis/*, 2001.

[12] S. Uchitel. *Incremental Elaboration of Scenario Based Specifications and Behavior Models Using Implied Scenarios.* PhD thesis, Imperial College, 2003.

[13] S. Uchitel, R. Chatley, J. Kramer, and J. Magee. LTSA-MSC: Tool support for behaviour model elaboration using implied scenarios. In H. Garavel and J. Hatcliff, editors, *TACAS*, volume 2619 of *Lecture Notes in Computer Science*, pages 597–601. Springer, 2003.

[14] S. Uchitel, J. Kramer, and J. Magee. Detecting implied scenarios in message sequence chart specifications. In *ESEC / SIGSOFT FSE*, pages 74–82, 2001.