Slimming down Petri Boxes: Compact Petri Net Models of Control Flows

Victor Khomenko
School of Computing, Newcastle University, Newcastle upon Tyne, UK

Maciej Koutny
School of Computing, Newcastle University, Newcastle upon Tyne, UK

Alex Yakovlev
School of Engineering, Newcastle University, Merz Court, Newcastle upon Tyne, UK

Abstract

We look at the construction of compact Petri net models corresponding to process algebra expressions supporting sequential, choice, and parallel compositions. If "silent" transitions are disallowed, a construction based on Cartesian product is traditionally used to construct places in the target Petri net, resulting in an exponential explosion in the net size. We demonstrate that this exponential explosion can be avoided, by developing a link between this construction problem and the problem of finding an edge clique cover of a graph that is guaranteed to be complement-reducible (i.e., a cograph). It turns out that the exponential number of places created by the Cartesian product construction can be reduced down to polynomial (quadratic) even in the worst case, and to logarithmic in the best (non-degraded) case. As these results affect the "core" modelling techniques based on Petri nets, eliminating a source of an exponential explosion, we hope they will have applications in Petri net modelling and translations of various formalisms to Petri nets.

2012 ACM Subject Classification Theory of computation → Concurrency

Keywords and phrases Petri net, Petri box, cograph, edge clique cover, control flow, static construction, local construction, interface graph, Burst automata, composition

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2022.8

1 Introduction

Petri nets have a special place among modelling formalisms due to their simplicity of semantics, intuitive graphical notation, and the possibility of capturing behaviours concisely without making subsequent processing (e.g., formal verification or synthesis) undecidable. This has led to the abundance of software tools for Petri nets, and to extensive use of Petri nets both as a modelling formalism and as an intermediate representation to which a model that was initially expressed in a different formalism is translated, e.g., to utilise efficient formal verification techniques and tools. In fact, developing translations from various process algebras and other formalisms to Petri nets has been a hot research topic for the past four decades, see e.g., [2, 5, 9, 12].

The possibility to create concise models is often the key advantage of Petri nets over simpler formalisms like Finite State Machines (FSMs). Indeed, it is generally accepted that one is likely to encounter the exponential *state space explosion* [13] during, e.g., formal verification – this problem is believed to be fundamental (unless P=PSPACE), and mitigating this explosion using heuristics has been a hot research topic for many years. However, encountering an exponential explosion already during the modelling stage would be unfortunate and indicative of problems in modelling techniques or even the formalism itself.

However, as we observed in [8], a naïve translation of even simple control flows to Petri nets may lead to an exponential explosion in the Petri net size. As a motivating example, [8] considers *Burst Automata* (BA) [3] – a formalism with applications in the area of asynchronous

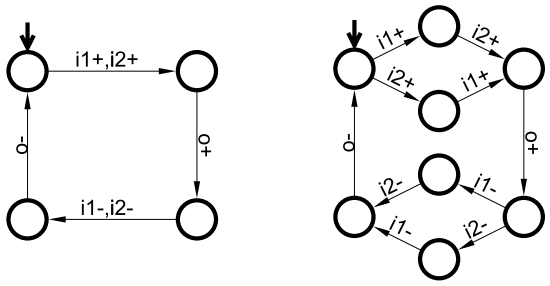

Figure 1 A Burst Automata specification of the C-element and an FSM expressing its interleaving semantics. A C-element waits for both inputs to switch to 1 (actions i_1^+ and i_2^+) before switching its output to 1 (action o^+), and then waits for both inputs to switch to 0 (actions i_1^- and i_2^-) before switching its output to 0 (action o^-). It is assumed that the environment fulfills its part of the contract, i.e. each input switches only once before the output switches.

circuits design. Intuitively, Burst Automata are similar to FSMs, except that their arcs are labelled not by single actions but by sets of actions ("bursts") which fire concurrently. One can define the interleaving semantics of Burst Automata by allowing the actions in a burst to fire in any order, which results in the usual FSM, see the example in Figure 1. When developing a translation between two formalisms, some kind of behavioural equivalence between the models is required, e.g., language equivalence or bisimulation of the corresponding FSMs. As Burst Automata are a very simple FSM-like formalism, it would be reasonable to expect that translating them to Petri nets would be quite simple and efficient.

However, developing a compact translation from Burst Automata to Petri nets is more complicated than one could expect – in particular, efficiently expressing a choice between several bursts of concurrent transitions is not trivial in Petri nets. In [3] a language-preserving linear size translation is proposed, that prefixes each burst with a silent "fork" transition and then uses another "join" transition after the burst to detect completion. Unfortunately, there are situations when this translation is unacceptable. First of all, silent transitions turn a deterministic model into a non-deterministic one, which is often undesirable (e.g., non-determinism cannot be directly implemented physically, say in an asynchronous logic circuit [4]). Second, language equivalence may be too weak (e.g., it does not preserve branching time temporal properties or even deadlocks), and prefixing bursts with silent transitions breaks not only strong but also weak bisimulation, see Figure 2.

To preserve strong bisimulation, the following *Cartesian Product Construction* ($\times$-construction) is traditionally used, see e.g., [1, 2, 5, 14]. To express a choice between several bursts (i.e., sets of concurrent transitions) $B_1, B_2, \ldots, B_n$, this construction would create a set of places corresponding to tuples in $B_1 \times B_2 \times \cdots \times B_n$, so that a place corresponding to a tuple $(b_1, \ldots, b_n)$ is connected to each transition b_i occurring in the tuple. This means that the number of created places is $|B_1| \cdot |B_2| \cdot \ldots \cdot |B_n|$, i.e., the Petri net size is exponential in the number of bursts. For example, [3] developed two translations from Burst Automata to Petri nets based on $\times$-construction, which preserve either weak or strong bisimulation. However, in contrast to the linear size translation of [3] (that does not preserve even weak bisimulation), they may result in an exponentially large Petri net.

In [8] we proposed an alternative to $\times$-construction, that uses at most quadratic (in the total size of all bursts) number of places to express a choice between bursts, thereby reducing the size of Burst Automata to Petri net translation from exponential [3] down to polynomial. Furthermore, in some cases a logarithmic number of places is sufficient, yielding a double-exponential reduction compared with $\times$-construction. The technique was based

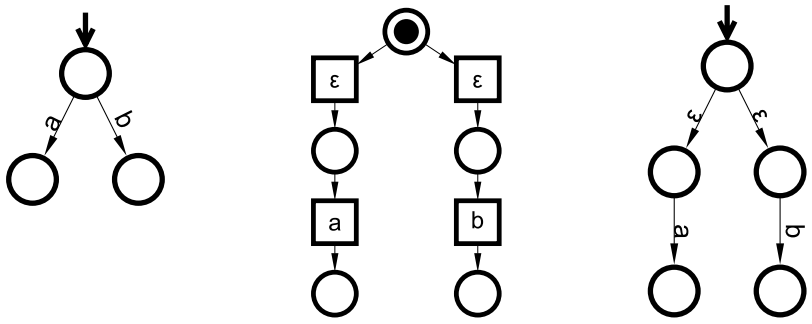

Figure 2 A Burst Automaton with singleton bursts, so coinciding with the FSM expressing its interleaving semantics (left); its Petri net translation prefixing each burst with a silent "fork" transition (middle); the reachability graph (FSM) of this Petri net (right). Note that the two FSMs are language-equivalent but not weakly bisimilar.

on showing the equivalence between the modelling problem of expressing a choice between bursts of concurrent events and the problem of finding an edge clique cover of a complete multipartite graph.

In this paper, we generalise the technique of [8] to arbitrary control flows which are built from atomic actions using choice, concurrency, and sequencing operators. In particular, a polynomial translation of such control flows to Petri nets is possible, that preserves strong bisimulation (in fact, it guarantees the isomorphism of reachability graphs, which is an even stronger equivalence). The developed technique allowed us to further improve the translation of Burst Automata to Petri nets proposed in [8] by handling the sequence operator better, see Figure 3. The developed Petri net translation is compositional – this is ensured by augmenting Petri Box Algebra [2] with the notion of *interface graphs* (superseding the entry and exit places of Petri boxes) in a way that allowed us to import many results from Petri Box Algebra into the new framework.

Since the proposed construction affects the "core" modelling techniques for Petri nets and because the choice, concurrency, and sequencing operators are included in most process algebras and other formalisms for behavioural modelling, we believe it will have many applications. In particular, translations from various formalisms to Petri nets relying on the $\times$-construction can be significantly improved by using the proposed construction instead, eliminating thus a source of an exponential explosion.

The proposed construction is based on the observation that the problem of "gluing" two Petri boxes sequentially is equivalent to finding an *edge clique cover* of a certain *complement-reducible graph (cograph)* where some of the edges are already considered as "covered", with the number of created places corresponding to the number of cliques in the cover. This results in an interesting optimisation problem that is in NP and likely NP-complete. In practice, the optimality is usually not required, and one can use simple approximations which yield useful lower and upper bounds – it is easy to see that at most polynomial (quadratic) number of cliques are always sufficient, which yields a polynomial Petri net.

2 Setting the scene

In this section, we describe a "bare bones" process algebra for expressing control flows, which can be regarded as a core fragment shared by many existing process algebras. We also discuss some basic notions related to Petri nets and clique covers of undirected graphs.

2.1 Models of concurrency

2.1.1 "Bare bones" process algebra

Consider a "bare bones" process algebra, where expressions are constructed from a finite alphabet of actions using operators " $\Box$ " (choice), " $\|$ " concurrency, and " ; " (sequencing). This algebra allows one to model acyclic control flows. It is very simple and, in fact, most existing process algebras build on it, by adding more features and operators (e.g., communication and recursion).

One can then define the semantics of such expressions, e.g., using Finite State Machines, which can be exponential in the size of the expression due to concurrency, e.g., the FSMs for expressions of the form $a_1 \| a_2 \| \ldots \| a_n$ would contain 2^n states.

One might hope that using Petri nets instead of FSMs would cope with the exponential explosion, i.e., that a polynomial translation from this process algebra to Petri nets is possible. However, as observed in [8], this is not trivial. In fact, the traditional $\times$-construction for modelling a choice between several "bursts" of concurrent actions creates an exponential number of places. For example, consider expressions of the form

$$(a_{11} \| \ldots \| a_{1n}) \Box \cdots \Box (a_{m1} \| \ldots \| a_{mn})$$

representing a choice between m "bursts" each containing n concurrent actions. The $\times$-construction creates m^n places to express this in a Petri net, which is exponential in the length of the above expression.

In this paper, we demonstrate that a polynomial bisimulation-preserving translation to Petri nets is indeed possible for any "bare bones" process algebra expressions. In fact, the isomorphism of reachability graphs (which is a stronger equivalence than strong bisimulation) holds for the developed translation.

2.1.2 Petri nets

We focus on *safe* (i.e., at most one token per place) Petri nets, which are often used for modelling control flows. For a safe Petri net, the total number of tokens in its initial marking cannot exceed the number of places, so we can define its size as the total number of places, transitions, and arcs, disregarding the initial marking. Note that the size of a Petri net is dominated by its arcs, except the uninteresting degraded case when there are many isolated nodes.

In this paper, the set of transitions is usually given (e.g., when translating a model from some other formalism to Petri nets, the transitions often correspond to the occurrences of actions in that model), and the objective is to express the intended behaviour using small numbers of places and arcs. Note that having a small number of places is often desirable for formal verification as they correspond to state variables, and having a small number of arcs is desirable as they dominate the Petri net size.

2.2 Graphs

We consider undirected graphs with no parallel edges and no self-loops. For simplicity, a graph $(\{v\}, \emptyset)$ comprising a single vertex v and no edges will be denoted just by v.

2.2.1 Edge clique covers

A *clique* in a graph is a set of vertices which are pairwise connected by edges. A clique is called *maximal* (or max-clique) if it is not a subset of any other clique. In what follows, $maxCL(G)$ is the set of all the max-cliques of an undirected graph G.

A set of cliques in a graph form an *edge clique cover* (ECC) if, for every edge, there is at least one clique that contains both endpoints of this edge. The number of cliques in an ECC is called its *size*. Note that, given an ECC, one can expand each clique in it to some maximal one, without increasing the size of the ECC. The minimum possible size of an ECC of a graph G is the *edge clique cover number* (a.k.a. *intersection number*) of G, and will be denoted $ecc(G)$.

2.2.2 Complete multipartite graphs

A graph is called *multipartite* if its vertices are partitioned into several sets in such a way that there are no edges between vertices in the same part. A multipartite graph is *complete* if for every pair of vertices from different parts there is an edge connecting them. A complete multipartite graph with the parts of sizes $t_1 \leq t_2 \leq \cdots \leq t_n$ will be denoted $K_{t_1,t_2,\ldots,t_n}$.

2.2.3 Cographs

Complement-reducible graphs (cographs) [10] can be recursively defined as follows: (i) a single vertex graph is a cograph; (ii) the complement of a cograph is a cograph; (iii) the disjoint union of cographs is a cograph. Intuitively, for every cograph G with more than one vertex, either G or its complement $\overline{G}$ is not connected. One can easily show that any complete multipartite graph is a cograph.

The *join* operation $G_1 — G_2$ consists of forming the disjoint union $G_1 \uplus G_2$ and then adding an edge between every vertex of G_1 and every vertex of G_2. One can see that one can reformulate the above definition of cographs so that it uses the join instead of the complement. Indeed, it is easy to see that $G_1 — G_2 = \overline{\overline{G_1} \uplus \overline{G_2}}$, i.e., join can be expressed via disjoint union and complementation. Similarly, one can express the complementation via disjoint union and join for cographs by repeatedly applying the following rewriting rules:

$$\begin{array}{rcl} \overline{v} & = & v \\ \overline{\overline{G}} & = & G \\ \overline{G_1 \uplus G_2} & = & \overline{G_1} — \overline{G_2}\,. \end{array}$$

3 Sequential composition and edge clique covers

In this section we informally present the underlying idea of the proposed construction, and illustrate it on a simple example from [8] of modelling a fragment of a BA shown in Figure 3(top-left) as a Petri net. There, a BA state with incoming bursts $\{i_1, i_2\}$ and $\{i_3\}$ and outgoing bursts $\{o_1, o_2\}$, $\{o_3, o_4\}$, and $\{o_5, o_6, o_7\}$ should be modelled as a set of Petri net places, assuming that the actions occurring in bursts are modelled as transitions with the corresponding names. More generally, we consider control flows constructed from atomic actions using sequential, choice, and parallel compositions. Such control flows can be naturally to represented by acyclic safe Petri nets (though there are interesting "unnatural" Petri net representations with cycles, see e.g., Section 4.3). The technique illustrated by the translation of this BA fragment can be applied recursively, and thus naturally generalises to such control flows.

In Petri nets modelling control flows, places have the dual role of enforcing both sequencing of transitions and choice between transitions. Indeed, let t_1 and t_2 be two transitions connected to the same place p:

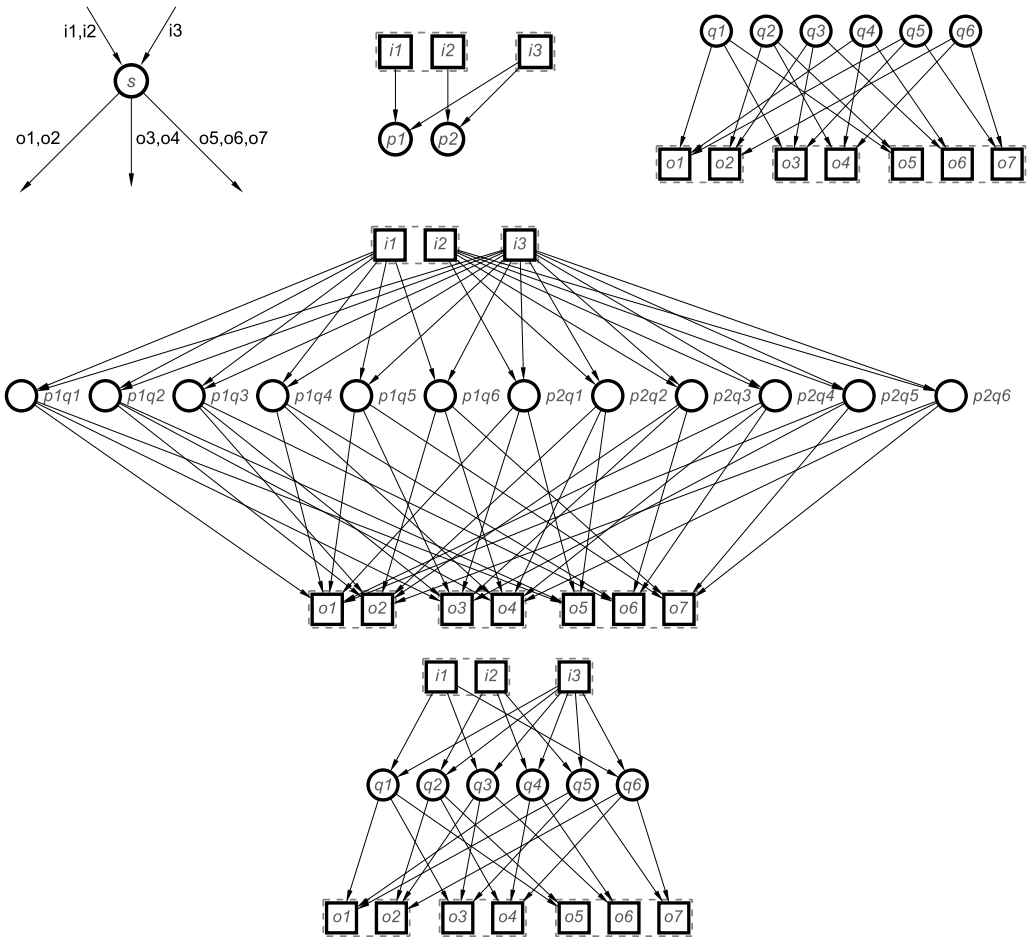

Figure 3 An example of bisimulation-preserving BA to Petri net translation: **(top-left)** a BA state with its incoming and outgoing bursts; **(top-centre)** Petri net translation [8] of incoming bursts – the maximal incoming burst size is two, so two places are created; **(top-right)** Petri net translation [8] of outgoing bursts – $ecc(K_{2,2,3}) = 6$ places are created; **(middle)** the combined Petri net [8] – the 12 places correspond to pairs in $\{p_1, p_2\} \times \{q_1, \ldots, q_6\}$; **(bottom)** the improved construction presented in this paper – the 6 places correspond to an ECC of $(i_1 \uplus i_2)$—i_3—$(o_1 \uplus o_2)$—$(o_3 \uplus o_4)$—$(o_5 \uplus o_6 \uplus o_7)$ with the edges of $(i_1 \uplus i_2)$—i_3 being optional to cover.

- If the connections are of the form $t_1 \to p \to t_2$ then p enforces sequencing of these transitions, as t_2 can fire only after t_1.
- If the connections are of the form $p \to t_1$ and $p \to t_2$ then p enforces the choice between these transitions, as only one of them can fire.
- If the connections are of the form $t_1 \to p$ and $t_2 \to p$ then there is a choice between these transitions (as otherwise the Petri net would be unsafe), but this choice is enforced not by p but by some other place.

Note that t_1 and t_2 are not concurrent if they are connected to the same place.

In our example, there is a choice between the incoming bursts $\{i_1, i_2\}$ and $\{i_3\}$ but it is enforced not by state s of the BA (or the places corresponding to s in the Petri net translation) but elsewhere. There is a choice between outgoing bursts $\{o_1, o_2\}$, $\{o_3, o_4\}$, and $\{o_5, o_6, o_7\}$, and it is enforced by s (and the corresponding places in the Petri net translation).

Moreover, the sequencing between these incoming and outgoing bursts is enforced by s (and the corresponding places in the Petri net translation). Note also that there is sequencing within neither incoming nor outgoing bursts, which leads to the following notion.

A set of transitions T is called *non-sequential* if no two distinct transitions in it are sequential. In other words, there may be choices between some transitions in such a set, and the transitions in any subset of T are concurrent as long as this subset contains no two transitions which are in the choice relationship. In our example, the set of incoming transitions $\{i_1, i_2, i_3\}$ and the set of outgoing transitions $\{o_1, o_2, o_3, o_4, o_5, o_6, o_7\}$ are non-sequential. The behaviour of T can be viewed as a collection of maximal such sets of concurrent transitions. One can represent the choice relation between transitions in T as a graph $G(T)$ (that will be later formalised in the notion of *interface graphs*) where the vertices are the transitions of T and there is an edge between two vertices iff the corresponding transitions are in the choice relationship.

Consider now two disjoint non-empty non-sequential sets of transitions, T_1 and T_2, which are to be composed sequentially. In the example shown in Figure 3(top-left),

$$\begin{array}{rcl} T_1 & = & \{i_1, i_2, i_3\} \\ G(T_1) & = & (i_1 \uplus i_2)\text{—}i_3 \\ T_2 & = & \{o_1, o_2, o_3, o_4, o_5, o_6, o_7\} \\ G(T_2) & = & (o_1 \uplus o_2)\text{—}(o_3 \uplus o_4)\text{—}(o_5 \uplus o_6 \uplus o_7)\ . \end{array}$$

The task is now to add places "between" transitions in T_1 and T_2 and connected to the transitions in T_1 by transition→place arcs, and to transitions in T_2 by place→transition arcs, so that:

- the behaviours of T_1 and T_2 are composed sequentially, i.e., none of the transitions in T_2 can fire until a maximal concurrent set of transitions in T_1 fires;
- the choices between transitions in T_2 are enforced (no need to enforce the choices between transitions in T_1 – this is done elsewhere).

Consider now a place p connected to some set of transitions $C \subseteq T_1 \cup T_2$ (the directions of arcs can be easily inferred). The key observation is that C must be a clique in $G(T_1)$—$G(T_2)$, as otherwise the resulting behaviour will be wrong. Indeed, for the sake of contradiction, suppose $t, t' \in C$ but there is no edge in $G(T_1)$—$G(T_2)$ connecting t and t'. Then these transitions are either both in T_1 or both in T_2, as otherwise there would be an edge between them. If these transitions are both in T_1, they are concurrent and both produce a token on p, resulting in an unsafe Petri net; e.g., in our example i_1 and i_2 are concurrent and including them both into C would result in p being unsafe. If $t, t' \in T_2$ then p creates a choice between them, even though they were meant to be concurrent, thus changing the intended behaviour; e.g., in our example o_1 and o_2 are concurrent, and including them both into C would result in p creating a choice between them.

Furthermore, though we consider acyclic control flows, in practice they are often just fragments of a higher level control flow that may contain cycles, which means the constructed acyclic Petri net will be a fragment of a larger Petri net that may contain cycles. This means the constructed Petri net should be "reusable", i.e., executable in a cycle without breaking the safeness of Petri net or introducing deadlocks. We now argue that in such a case C must be a max-clique in $G(T_1)$—$G(T_2)$.

First, we show that reusability implies $C \not\subseteq T_1$ and $C \not\subseteq T_2$. Indeed, if $C \subseteq T_1$ then p has no outgoing arcs – besides being useless as it will never affect the enabledness of any transitions, p also accumulates tokens and an attempt to reuse this Petri net will result in unsafeness. If $C \subseteq T_2$ then p has no incoming arcs and so cannot obtain a new token – even if it contains a token initially, the Petri net fragment cannot be reused as this can introduce a deadlock.

Now, for the sake of contradiction, suppose C is not maximal, i.e., it can be expanded to a larger clique by adding t. If $t \in T_1$ then firing t means none of the transitions in $C \cap T_1 \neq \emptyset$ can fire, as each of them is in choice relationship with t, in which case p does not obtain a token and transitions in $C \cap T_2 \neq \emptyset$ will not be able to fire, which changes the intended behavior. If $t \in T_2$ then firing t means none of the transitions in $C \cap T_2 \neq \emptyset$ can fire, as each of them is in choice relationship with t, in which case the token in p cannot be consumed and the Petri net is not reusable as this would introduce unsafeness. Hence, C is a max-clique in $G(T_1)$—$G(T_2)$; note that this implies $C \not\subseteq T_1$ and $C \not\subseteq T_2$.

We have now established that every constructed place must correspond to some max-clique in $G(T_1)$—$G(T_2)$. The question now is how to select a set of such places sufficient to express the desired behavior, which can be reformulated as a question of selecting a sufficient number of max-cliques in $G(T_1)$—$G(T_2)$. One can observe that this problem is similar to the well known problem of computing an edge clique cover of $G(T_1)$—$G(T_2)$, except that covering the edges in the induced subgraph $G(T_1)$ is optional. (Note that any edge clique cover can be easily extended to one containing only max-cliques, without increasing the number of cliques.) Indeed, suppose (t, t') is an edge of $G(T_1)$—$G(T_2)$:

- If $t, t' \in T_1$ then t and t' are in the choice relationship but this is enforced elsewhere, so it is not necessary (and not possible) to enforce it by adding a place p with the arcs $t \to p$ and $t' \to p$; on the other hand, there is no harm having such a place, e.g., both t and t' can be in the same max-clique that also covers other edges of $G(T_1)$—$G(T_2)$. Hence, it is optional to cover the edge (t, t').
- If $t, t' \in T_2$ then t and t' are in the choice relationship and this must be enforced by having a place p with the arcs $p \to t$ and $p \to t'$. Hence, it is necessary to cover the edge (t, t').
- If $t \in T_1$ and $t' \in T_2$ then t must fire before t' and this must be enforced by having a place p with the arcs $t \to p$ and $p \to t'$. Hence, it is necessary to cover the edge (t, t').

This version of the edge clique cover problem will be called *partial edge clique cover problem* (PECCP). It is a natural extension of the standard edge clique cover problem (ECCP), in fact some state-of-art ECCP algorithms, e.g., [6], start from an empty cover and keep adding cliques one-by-one until all edges are covered; hence, at the intermediate stages of the computation, the problem is equivalent to PECCP, with the edges covered by previously added cliques becoming optional to cover. This means that many existing ECCP algorithms can be adapted to solve PECCP.

Note that PECCP is trivially in NP. Moreover, since ECCP is NP-complete for general graphs and is a special case of PECCP, it follows that PECCP is NP-complete for general graphs. However, for control flows constructed from atomic actions using sequential, choice, and parallel compositions, the graphs on which PECCP is solved are guaranteed to be cographs, which is a very restricted subclass of graphs, with many NP-complete problems (e.g., computing a clique of maximal cardinality) becoming polynomial when restricted to this class. However, to our knowledge, the question whether ECCP or PECCP is NP-complete on cographs is still open. Note also that for modelling control flows the optimality is not required, so fast heuristic algorithms computing small but not necessarily smallest covers would be sufficient – in fact, the trivial ECC covering each edge by a separate clique already avoids the exponential explosion due to the $\times$-construction.

Coming back to our example, our previous method described in [8] creates 12 places. It handles incoming and outgoing bursts separately; for the incoming bursts the number of created places corresponds to the maximal input burst cardinality (2 in our example, see Figure 3(top-centre)), and for the outgoing bursts the places correspond to ECC of $G(T_2)$,

i.e., $(o_1 \uplus o_2)$—$(o_3 \uplus o_4)$—$(o_5 \uplus o_6 \uplus o_7)$ in our example. For BAs $G(T_2)$ is guaranteed to be a multipartite graph, $K_{2,2,3}$ in this case – it has an ECC of size 6, i.e., 6 places are created, see Figure 3(top-right). Then a variant of $\times$-construction is used to enforce the sequencing of incoming and outgoing bursts, which results in $2 \cdot 6 = 12$ places, as shown in Figure 3(middle).

Though the construction in [8] yields a polynomial BA to Petri net translation, the translation presented in this paper significantly improves it. For this example, $G(T_1)$—$G(T_2) = (i_1 \uplus i_2)$—$i_3$—$(o_1 \uplus o_2)$—$(o_3 \uplus o_4)$—$(o_5 \uplus o_6 \uplus o_7)$, with the edges of the induced subgraph $(i_1 \uplus i_2)$—i_3 being optional to cover. Solving PECCP yields a cover with 6 cliques, and the corresponding Petri net fragment with 6 places is shown in Figure 3(bottom).

4 Slimming down Box Algebra

Box Algebra [2] provided a generic process-algebraic syntax together with a compositional translation to a class of Petri nets called *(Petri) boxes*. The algebra has several concrete incarnations, including CCS [11] and TCSP [7]. Here we are only interested in a small fragment of Box Algebra corresponding to the "bare bones' process algebra, in order to focus on the salient aspect of control flow in nets constructed from three fundamental composition operators. In particular, we omit all communication and synchronisation aspects.

Process expressions, or *box expressions*, are derived from the syntax

$$E \ ::= \ a \ \mid \ E\,;E \ \mid \ E \,\Box\, E \ \mid \ E \parallel E \tag{1}$$

where a is an atomic action. Since we deal only with control flows and actions do not have any special semantics (e.g., communication), we can assume that no action occurs more than once in any box expression we consider. Intuitively, a denotes a process which can execute atomic action a and terminate, $E\,;F$ denotes sequential composition of two processes, $E \,\Box\, F$ denotes choice composition, and $E \parallel F$ denotes parallel composition.

The semantics of box expressions is given through a translation into Petri nets, called *boxes*. For the simple syntax (1), each place can be uniquely identified by its input and output transitions, and $\pi_{U,W}$ will denote a place with input transitions U and output transitions W, i.e., there is an arrow from transition t to $\pi_{U,W}$ iff $t \in U$, and there is an arrow from $\pi_{U,W}$ to transition t iff $t \in W$.

A *box* is a pair $N = (P, T)$, where $P(= P_N)$ is a finite set of *places* (local states) and $T(= T_N)$ is a disjoint finite set of *transitions* (actions) such that, for every transition $t \in T$, there is a place $\pi_{U,W} \in P$ with $t \in W$.

The flow relation of N (represented by arrows) is implicit and can be recovered from the sets indexing places: the *input* and *output* places of a transition $t \in T$ are respectively given by $pre_N(t) = \{\pi_{U,W} \in P \mid t \in W\}$ and $post_N(t) = \{\pi_{U,W} \in P \mid t \in U\}$.

Associating boxes with box expressions is done compositionally, through the $\mathsf{box}(\cdot)$ mapping, by combining their entry and exit places to reflect the intended control flow of execution, where the *entry* places N^{e} of a box N have the form $\pi_{\varnothing,W} \in P$, and the *exit* places N^{x} have the form $\pi_{U,\varnothing} \in P$. The *internal* places N^{i} are all the remaining places of N. Intuitively:

- $\mathsf{box}(E \parallel F)$ is obtained simply by placing $\mathsf{box}(E)$ and $\mathsf{box}(F)$ side by side.
- $\mathsf{box}(E \,\Box\, F)$ is obtained by placing $\mathsf{box}(E)$ and $\mathsf{box}(F)$ side by side and then gluing each entry place of $\mathsf{box}(E)$ with each entry place of $\mathsf{box}(F)$, effectively creating the product $\mathsf{box}(E)^{\mathsf{e}} \times \mathsf{box}(F)^{\mathsf{e}}$, and similarly for the exit places.
- $\mathsf{box}(E\,;F)$ is obtained by placing $\mathsf{box}(E)$ above $\mathsf{box}(F)$ and then gluing each exit place of $\mathsf{box}(E)$ with each entry place of $\mathsf{box}(F)$, effectively creating the product $\mathsf{box}(E)^{\mathsf{x}} \times \mathsf{box}(F)^{\mathsf{e}}$.

The above procedure will be referred to as the $\times$-*construction*. Formally, the box corresponding to a given box expression is defined recursively as follows:

$$\begin{array}{lcl} \mathsf{box}(a) & = & (\{\pi_{\varnothing,\{a\}}, \pi_{\{a\},\varnothing}\}, \{a\}) \\ \mathsf{box}(E \parallel F) & = & (P_{\mathsf{box}(E)} \cup P_{\mathsf{box}(F)}, \mathcal{T}) \\ \mathsf{box}(E \,\square\, F) & = & (\mathcal{E} \cup \mathcal{P} \cup \mathcal{X}, \mathcal{T}) \\ \mathsf{box}(E\,;F) & = & (\mathsf{box}(E)^{\mathsf{e}} \cup \mathcal{P} \cup \mathcal{I} \cup \mathsf{box}(F)^{\mathsf{x}}, \mathcal{T}) \end{array}$$

where $\mathcal{T} = T_{\mathsf{box}(E)} \cup T_{\mathsf{box}(F)}$, $\mathcal{P} = \mathsf{box}(E)^{\mathsf{i}} \cup \mathsf{box}(F)^{\mathsf{i}}$, and:

$$\begin{array}{lcll} \mathcal{E} & = & \{\pi_{\varnothing,U\cup W} & \mid \ \pi_{\varnothing,U} \in \mathsf{box}(E)^{\mathsf{e}} \wedge \pi_{\varnothing,W} \in \mathsf{box}(F)^{\mathsf{e}}\} \\ \mathcal{X} & = & \{\pi_{U\cup W,\varnothing} & \mid \ \pi_{U,\varnothing} \in \mathsf{box}(E)^{\mathsf{x}} \wedge \pi_{W,\varnothing} \in \mathsf{box}(F)^{\mathsf{x}}\} \\ \mathcal{I} & = & \{\pi_{U,W} & \mid \ \pi_{U,\varnothing} \in \mathsf{box}(E)^{\mathsf{x}} \wedge \pi_{\varnothing,W} \in \mathsf{box}(F)^{\mathsf{e}}\}. \end{array} \tag{2}$$

Markings (global states) of a box $N = (P, T)$ we consider are sets of places. The default initial marking is N^{e}. A transition $t \in T$ is *enabled* at marking M if $pre_N(t) \leq M$. It then can be *fired* leading to the marking $M' = M - pre_N(t) + post_N(t)$, and we denote this by $M[t\rangle_N M'$. An overall behaviour of N is given by its *reachability graph* defined as a labelled directed graph $RG(N) = (\mathcal{R}, \mathcal{A}, N^{\mathsf{e}})$, where $\mathcal{R}$ are the *reachable markings* of N (i.e., the least set containing N^{e} and such that if $M \in \mathcal{R}$ and $M[t\rangle_N M'$ then $M' \in \mathcal{R}$), and $\mathcal{A}$ contains all labelled arcs (M, t, M') such that $M \in \mathcal{R}$ and $M[t\rangle_N M'$.

To formulate a central semantical result of this paper – a full (local) characterisation of successful slimming down of compositionally defined boxes – we need two more notions. The *local conflict* and *local causality* of N are two relations on transitions given respectively by:

$$\begin{array}{rcl} \mathit{Confl}(N) & = & \{(t,u) \in T \times T \mid pre_N(t) \cap pre_N(u) \neq \varnothing\} \\ \mathit{Caused}(N) & = & \{(t,u) \in T \times T \mid post_N(t) \cap pre_N(u) \neq \varnothing\}. \end{array} \tag{3}$$

Intuitively, these two relations capture the dual role of Petri net places in enforcing both choice and causality between transitions.

▶ **Proposition 1.** *Let E be a box expression derived from the syntax (1), and $N = (P, T_{\mathsf{box}(E)})$ be a box such that $P \subseteq P_{\mathsf{box}(E)}$.*
Then $RG(N)$ and $RG(\mathsf{box}(E))$ are isomorphic reachability graphs if and only if $\mathit{Confl}(N) = \mathit{Confl}(\mathsf{box}(E))$ and $\mathit{Caused}(N) = \mathit{Caused}(\mathsf{box}(E))$.

Proposition 1 means that we can safely delete places from $\mathsf{box}(E)$ iff the local conflict and local causality relations on the transitions are retained. Note that such a property does not hold in general – in either direction – not even for boxes which are safe and acyclic. Indeed, let $N = \mathsf{box}(a; b; c)$, and let N' be N with an added place $\pi_{\{a\},\{c\}}$. Then $RG(N)$ is isomorphic to $RG(N')$, but

$$\mathit{Caused}(N') = \mathit{Caused}(N) \uplus \{(a, c)\} \ .$$

As a complementary example, let $N = \mathsf{box}((a \,\square\, b); c)$, and let N' be N with an added place $\pi_{\{b\},\{c\}}$. Then

$$\mathit{Confl}(N) = \mathit{Confl}(N') \quad \text{and} \quad \mathit{Caused}(N) = \mathit{Caused}(N') \ ,$$

but $RG(N)$ is not isomorphic to $RG(N')$ as in $RG(N')$ one cannot "execute' a followed by c.

Our goal now becomes that of finding criteria for identifying possibly largest sets of such "safe deletions" characterised by Proposition 1, and thus transferring the slimming down problem from semantic to algorithmic setting. What is more, we aim at developing a "static" approach to slimming, i.e., one that does not require looking into the behaviour of boxes.

4.1 Interface graphs

In this section, we introduce and investigate a novel concept in the area of Petri boxes which aims at capturing different ways in which local conflict and local causality can arise. We start by introducing *efficient (quadratic) representation* of the entry and exit places of $\mathsf{box}(E)$ without going through the *expensive (exponential)* process of applying the $\times$-construction.

For each box expression E derived from the syntax (1), let G^{e}_E and G^{x}_E be undirected *interface graphs* defined recursively as follows:

$$\begin{array}{lcllcl} G^{\mathsf{e}}_a & = & a & G^{\mathsf{x}}_a & = & a \\ G^{\mathsf{e}}_{E;F} & = & G^{\mathsf{e}}_E & G^{\mathsf{x}}_{E;F} & = & G^{\mathsf{x}}_F \\ G^{\mathsf{e}}_{E\,\square\,F} & = & G^{\mathsf{e}}_E \text{---} G^{\mathsf{e}}_F & G^{\mathsf{x}}_{E\,\square\,F} & = & G^{\mathsf{x}}_E \text{---} G^{\mathsf{x}}_F \\ G^{\mathsf{e}}_{E\,\|\,F} & = & G^{\mathsf{e}}_E \uplus G^{\mathsf{e}}_F & G^{\mathsf{x}}_{E\,\|\,F} & = & G^{\mathsf{x}}_E \uplus G^{\mathsf{x}}_F \end{array} \tag{4}$$

Note that the vertices of G^{e}_E and G^{x}_E are transitions of $\mathsf{box}(E)$, and that inteface graphs are cographs. For example,

$$\begin{array}{lcl} G^{\mathsf{e}}_{(a\|b)\,\square\,c);(d\|e)} & = & G^{\mathsf{e}}_{(a\|b)\,\square\,c} \\ & = & G^{\mathsf{e}}_{a\|b} \text{---} c \\ & = & (a \uplus b) \text{---} c \\ G^{\mathsf{x}}_{(a\|b)\,\square\,c);(d\|e)} & = & G^{\mathsf{x}}_{d\|e} \\ & = & d \uplus e\ . \end{array}$$

▶ **Proposition 2.** *For box expression E derived from the syntax (1),*

$$\begin{array}{lcl} \mathsf{box}(E)^{\mathsf{e}} & = & \{\pi_{\varnothing,Cl} \mid Cl \in maxCL(G^{\mathsf{e}}_E)\} \\ \mathsf{box}(E)^{\mathsf{x}} & = & \{\pi_{Cl,\varnothing} \mid Cl \in maxCL(G^{\mathsf{x}}_E)\}\ . \end{array}$$

Hence the entry places of $\mathsf{box}(E)$ can be identified with the set of all max-cliques of G^{e}_E, and similarly for the exit places. As a result, by Proposition 1, removing some entry places from $\mathsf{box}(E)$ without changing the overall behaviour is the same as choosing an edge covering of G^{e}_E by max-cliques (and the best result is therefore obtained by taking a minimal edge covering of G^{e}_E by max-cliques).

A similar observation holds for the internal places:

▶ **Proposition 3.** *If $\mathcal{I}$ is a set of internal places as in Eq. (2), derived during the $\times$-construction, then:*

$$\mathcal{I} = \{\pi_{Cl\cap T_{\mathsf{box}(E)},Cl\cap T_{\mathsf{box}(F)}} \mid Cl \in maxCL(G^{\mathsf{x}}_E \text{---} G^{\mathsf{e}}_F)\}\ .$$

4.2 New construction

"Some people want it to happen.
Some wish it would happen.
Others make it happen" Michael Jordan on healthy dieting

We now introduce an alternative to the $\times$-construction, for a box expression H derived from the syntax (1). Formally, a *slimming of H* is any box N derived from $\mathsf{box}(H)$, in the following way:

- The places in $\mathsf{box}(H)^{\mathsf{e}}$ are replaced by $\{\pi_{\varnothing,Cl} \mid Cl \in \mathcal{CL}\}$, where $\mathcal{CL}$ is any minimal edge covering of G^{e}_H by max-cliques.

- Set $\mathcal{I}$ of internal places as in Eq. (2) corresponding to every sub-expression of the form $E;F$ within H is replaced by

$$\{\pi_{Cl\cap T_{\mathsf{box}(E)},Cl\cap T_{\mathsf{box}(F)}} \mid Cl \in \mathcal{CL}\},$$

  where $\mathcal{CL}$ is any minimal edge covering of G^{x}_E—G^{e}_F by max-cliques, with the edges within its subgraph G^{x}_E being optional to cover.
- The places in $\mathsf{box}(H)^{\mathsf{x}}$ are deleted.

Note also that there is no need to generate at all the sets $\mathsf{box}(H)^{\mathsf{e}}$, $\mathsf{box}(H)^{\mathsf{x}}$ and $\mathcal{I}$, and only operate on graphs and their covers to derive the places of N. The proposed (non-deterministic) construction is therefore truly static and local.

▶ **Proposition 4.** *Let E be a box expression derived from the syntax (1) and N be a slimming of E. Then the following hold:*

1. *$RG(N)$ and $RG(\mathsf{box}(E))$ are isomorphic reachability graphs.*
2. *If $N' = (P, T_{\mathsf{box}(E)})$ is a box such that $P \subset P_{\mathsf{box}}(E)$ and $|P| < |P_N|$, then $RG(N')$ and $RG(\mathsf{box}(E))$ are not isomorphic reachability graphs (and not even language equivalent).*

The above is a key result validating the proposed way of deleting places from the composite boxes, and demonstrating its local optimality in the sense that deleting any remaining place will change the behaviour (note that there are other reasonable notions of optimality, see below). As was explained before, computing an optimal slimming is equivalent to solving PECCP on a cograph – to our knowledge it is still an open question whether this problem is NP-complete. Having said that, in practice, one does not need minimal covers and heuristic approximations yield good solutions (even trivial covers yield polynomial boxes contrasting with the exponential $\times$-construction). Note also that Proposition 4 can be lifted to other control flow operators of the Box Algebra, e.g., iteration.

4.3 No global optimality

The construction proposed in this paper achieves *local* optimality in terms of the number of places, as well as a guaranteed polynomial (in the length of the box expression) size of the overall Petri net, as opposed to the original $\times$-construction that is exponential. Furthermore, in cases like

$$(a_1 \parallel b_1) \,\square\, \ldots \,\square\, (a_n \parallel b_n)\ ,$$

the number of created places can be as low as logarithmic – a double-exponential reduction w.r.t. the $\times$-construction.

This naturally raises the question whether the proposed construction is globally optimal, i.e., whether for a given box expression, a safe Petri net with the minimum possible number of places is always constructed. Unfortunately the answer turns out to be negative. In fact, the number of places is not even asymptotically optimal. We demonstrate this using the following very simple example.

Consider the box expression $a_1\,;\,\ldots\,;a_n$, for which the proposed construction generates a Petri net with n places (denoted p_i, for $i = 1,\ldots,n$), such that there are arcs from a_i to p_{i+1}, for $i = 1,\ldots,n-1$, and from p_i to a_i, for $i = 1,\ldots,n$. Moreover, p_1 is initially marked.

Observe that at most one of these places contains a token, and so the reachable markings of this Petri nets can be compactly encoded using only a logarithmic number of places, e.g., using the following construction. For simplicity, we assume that $n = \binom{k}{k/2}$ for some even k, i.e., n is the number of subsets of size $k/2$ of $\{1,\ldots,k\}$. We denote these subsets $P_1,\ldots,P_n$ (the order can be chosen arbitrarily). Note that $k \sim \log_2 n$ as one can check (using, e.g., `wolframalpha.com`) that

$$\lim_{k\to+\infty} \frac{\log_2 n}{k} = \lim_{k\to+\infty} \frac{\log_2 \binom{k}{k/2}}{k} = 1 \,.$$

Consider now a Petri net with the transitions a_i, $i = 1, \ldots, n$, and k places numbered 1 to k. One can now interpret $P_1, \ldots, P_n$ as subsets of the set of places of this Petri net. These places and transitions are connected so that there are arcs from a_i to each place in P_{i+1}, for $i = 1, \ldots, n-1$, and from each place in P_i to a_i, for $i = 1, \ldots, n$. Moreover, the places in P_1 are initially marked. One can easily see that this Petri net has the expected behaviour, but only $k \sim \log_2 n$ rather than n places.

5 Conclusions

In this paper, we observed that the $\times$-construction traditionally used for composing, e.g., Petri boxes, is sub-optimal and causes an exponential explosion in the size of Petri nets that can be avoided. We showed the equivalence between this modelling problem and the problem of finding an ECC of a cograph, where the covering of some edges is considered optional. This allowed us to develop a polynomial Petri net translation of arbitrary control flows built from atomic actions using the sequential, choice, and parallel compositions.

These results affect the "core" modelling techniques based on Petri nets and eliminate a source of exponential explosion when modelling control flows, and in translations from various process algebras and other formalisms to Petri nets.

References

1 Eike Best, Raymond R. Devillers, and Jon G. Hall. The box calculus: a new causal algebra with multi-label communication. In Grzegorz Rozenberg, editor, *Advances in Petri Nets 1992, The DEMON Project*, volume 609 of *Lecture Notes in Computer Science*, pages 21–69. Springer, 1992. doi:10.1007/3-540-55610-9_167.

2 Eike Best, Raymond R. Devillers, and Maciej Koutny. *Petri net algebra*. Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2001.

3 A. Chan, Danil Sokolov, Victor Khomenko, David Lloyd, and Alex Yakovlev. Burst automaton: Framework for speed-independent synthesis using burst-mode specifications. *submitted*, 2021.

4 Jordi Cortadella, Michael Kishinevsky, Alex Kondratyev, Luciano Lavagno, and Alex Yakovlev. *Logic Synthesis for Asynchronous Controllers and Interfaces*. Springer, 2002.

5 Ursula Goltz and Alan Mycroft. On the relationship of CCS and Petri nets. In Jan Paredaens, editor, *Automata, Languages and Programming, 11th Colloquium, Antwerp, Belgium, July 16-20, 1984, Proceedings*, volume 172 of *Lecture Notes in Computer Science*, pages 196–208. Springer, 1984.

6 Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Data reduction and exact algorithms for clique cover. *ACM J. Experimental Algorithmics*, 13, 2009.

7 C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

8 Victor Khomenko, Maciej Koutny, and Alex Yakovlev. Avoiding exponential explosion in Petri net models of control flows. In *Proc. Petri Nets'22*, Lecture Notes in Computer Science. Springer, 2022. accepted paper.

9 Victor Khomenko, Roland Meyer, and Reiner Hüchting. A polynomial translation of pi-calculus FCPs to safe Petri nets. *Log. Methods Comput. Sci.*, 9(3), 2013.

10 H. Lerchs. *On cliques and kernels*. Tech. Report, Dept. of Comp. Sci., Univ. of Toronto, 1971.

11 Robin Milner. *A Calculus of Communicating Systems*. Springer, 1980.

12 Ernst-Rüdiger Olderog. Operational Petri net semantics for CCSP. In Grzegorz Rozenberg, editor, *Advances in Petri Nets 1987, covers the 7th European Workshop on Applications and Theory of Petri Nets, Oxford, UK, June 1986*, volume 266 of *Lecture Notes in Computer Science*, pages 196–223. Springer, 1986.

13 Antti Valmari. The state explosion problem. In Wolfgang Reisig and Grzegorz Rozenberg, editors, *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, the volumes are based on the Advanced Course on Petri Nets, held in Dagstuhl, September 1996*, volume 1491 of *Lecture Notes in Computer Science*, pages 429–528. Springer, 1996.

14 Rob J. van Glabbeek and Ursula Goltz. Refinement of actions in causality based models. In J. W. de Bakker, Willem P. de Roever, and Grzegorz Rozenberg, editors, *Stepwise Refinement of Distributed Systems, Models, Formalisms, Correctness, REX Workshop, Mook, The Netherlands, May 29 - June 2, 1989, Proceedings*, volume 430 of *Lecture Notes in Computer Science*, pages 267–300. Springer, 1989. doi:10.1007/3-540-52559-9_68.

A Additional notions and notations

Let $N = (P, T)$ be a box.

- If a transition $t \in T$ is enabled at marking M, we denote $t \in \mathit{enabled}_N(M)$.
- A marking M' is *reachable from marking* M if there are markings $M_0, \ldots, M_k$ $(k \leq 0)$ and transitions $t_1, \ldots, t_k$ such that $M_0 = M$, $M_k = M'$, and

  $$M_0[t_1\rangle_N M_1 \ldots M_{k-1}[t_k\rangle_N M_k.$$

  Moreover, if $M_0 = N^{\mathsf{e}}$ then $\sigma = t_1 \ldots t_k$ is a *firing sequence* of N (we denote this by $\sigma \in \mathit{fseq}(N)$) and M_k is a *reachable marking* of N (we denote this by $M_k \in \mathit{reach}(N)$).
- Two transitions $t \neq u \in T$ are *in conflict* if there is a place $p = \pi_{U,W} \in P$ such that $t, u \in W$ (we denote this by $(t, u) \in \mathit{cfl}_N(p)$), and t *directly causes* u if there is a place $p = \pi_{U,W} \in P$ such that $t \in U$ and $u \in W$ (we denote this by $(t, u) \in \mathit{csd}_N(p)$). Moreover, for a set of places $P' \subseteq P$:

  $$\mathit{cfl}_N(P') = \bigcup\{\mathit{cfl}_N(p) \mid p \in P'\} \quad \text{and} \quad \mathit{csd}_N(P') = \bigcup\{\mathit{csd}_N(p) \mid p \in P'\}.$$

B Proof of Proposition 1

▶ **Lemma 5.** *Let E be a box expression derived from the syntax (1), $N = \mathsf{box}(E)$, $t \in T_N$, and $(N^{\mathsf{e}} =) M_0[t_1\rangle_N M_1 \ldots M_{k-1}[t_k\rangle_N M_k$.*[1]

1. *N is safe box (i.e., each reachable marking is a set of places).*[2]
2. *N^{x} is a marking reachable from M_k, and $\mathit{enabled}_N(N^{\mathsf{x}}) = \varnothing$.*
3. *$\mathit{post}_N(t_i) \cap \mathit{post}_N(t_j) = \varnothing$ and $t_i \neq t_j$, for all $1 \leq i < j \leq k$.*
4. *If $M_k \cap \mathit{pre}_N(t) \neq \varnothing$ then:*
   - *there is $1 \leq i \leq k$ such that $t \in \mathit{enabled}_N(M_i)$, or*
   - *there is a marking M reachable from M_k such that $t \in \mathit{enabled}_N(M)$.*

Proof. Different parts follow by induction on the structure of a box expression (see also [2]). ◀

[1] Hence $t_1 \ldots t_k$ is a firing sequence of N.
[2] Hence we can use set notation when dealing with the semantics of composite boxes.

We then proceed with the proof proper, using the same notations as in the formulation of Proposition 1. We first observe that $N^{\mathsf{e}} = \mathsf{box}(E)^{\mathsf{e}} \cap P$ as well as $\mathit{cfl}_{\mathsf{box}(E)}(p) = \mathit{cfl}_N(p)$ and $\mathit{csd}_{\mathsf{box}(E)}(p) = \mathit{csd}_N(p)$, for every $p \in P$. Hence $\mathit{cfl}_{\mathsf{box}(E)}(P_{\mathsf{box}(E)}) = \mathit{cfl}_N(P)$ and $\mathit{csd}_{\mathsf{box}(E)}(P_{\mathsf{box}(E)}) = \mathit{csd}_N(P)$. Moreover, if

$$\mathsf{box}(E)^{\mathsf{e}}[t_1\rangle_{\mathsf{box}(E)} M_1 \ldots M_{k-1}[t_k\rangle_{\mathsf{box}(E)} M_k$$

then $N^{\mathsf{e}}[t_1\rangle_N (M_1 \cap P) \ldots (M_{k-1} \cap P)[t_k\rangle_N (M_k \cap P)$ since N is a subnet of $\mathsf{box}(E)$ with the same set of transitions (*).

We then observe that $\mathit{fseq}(N) = \mathit{fseq}(\mathsf{box}(E))$. Indeed, the ($\supseteq$) inclusion follows from (*). If the ($\subseteq$) inclusion does not hold, then there is $t_1 \ldots t_k t \in \mathit{fseq}(N) \setminus \mathit{fseq}(\mathsf{box}(E))$ such that $t_1 \ldots t_k \in \mathit{fseq}(\mathsf{box}(E))$. Thus (also by (*)) there are markings $M_0, \ldots, M_k$ such that

$$\begin{aligned} (\mathsf{box}(E)^{\mathsf{e}} =) & M_0[t_1\rangle_{\mathsf{box}(E)} M_1 \ldots M_{k-1}[t_k\rangle_{\mathsf{box}(E)} M_k \\ (N^{\mathsf{e}} =) & M_0 \cap P[t_1\rangle_N (M_1 \cap P) \ldots (M_{k-1} \cap P)[t_k\rangle_N (M_k \cap P). \end{aligned}$$

We have, $t \in \mathit{enabled}_N(M_k \cap P) \setminus \mathit{enabled}_{\mathsf{box}(E)}(M_k)$. Hence there is $p = \pi_{U,W} \in \mathit{pre}_{\mathsf{box}(E)}(t) \setminus \mathit{pre}_N(t)$ such that $M_k(p) = 0$. On the other hand, since $\mathit{pre}_N(t) \neq \varnothing$ and $t \in \mathit{enabled}_N(M_k \cap P)$, $\mathit{pre}_{\mathsf{box}(E)}(t) \cap M_k \neq \varnothing$. Hence, by Lemma 5(4), one of the following two cases holds:

Case 1: There is $1 \leq i \leq k$ such that $t \in \mathit{enabled}_{\mathsf{box}(E)}(M_i)$ and so $p \in M_i$. Then there is $i < j \leq k$ such that $p \in \mathit{pre}_{\mathsf{box}(E)}(t_j)$. By $\mathit{cfl}_N(P) = \mathit{cfl}_{\mathsf{box}(E)}(P_{\mathsf{box}(E)})$, there is $p' \in P$ such that $M_k(p') = 1$ and $p' \in \mathit{pre}_{\mathsf{box}(E)}(t) \cap \mathit{pre}_{\mathsf{box}(E)}(u)$. However, by Lemma 5(1), this means that p' must have been filled with a token twice along the firing sequence $t_1 \ldots t_k$, contradicting Lemma 5(3).

Case 2: There is a marking M reachable from M_k such that $t \in \mathit{enabled}_{\mathsf{box}(E)}(M)$. This means $p \in M$, and there is $u \in U$ which is executed when reaching M from M_k. Then $(u,t) \in \mathit{csd}_{\mathsf{box}(E)}(p)$, and so, by $\mathit{csd}_{\mathsf{box}(E)}(P) = \mathit{csd}_{\mathsf{box}(E)}(P_{\mathsf{box}(E)})$, there is $p' \in P$ such that $(u,t) \in \mathit{csd}_{\mathsf{box}(E)}(p')$. Thus $p' \in M_k$. As a result, p' must have received a token twice along a firing sequence of $\mathsf{box}(E)$ leading to M, contradicting Lemma 5(3).

Hence $\mathit{fseq}(N) = \mathit{fseq}(\mathsf{box}(E))$ (**). Moreover, by (*), if σ and σ' are firing sequences leading in $\mathsf{box}(E)$ to the same marking, then they also lead to the same marking in N. Suppose then that σ and σ' are firing sequences leading in N to the same marking. Then, by Lemma 5(2), they can be extended by the same σ'' to yield firing sequences $\sigma\sigma''$ and $\sigma'\sigma''$ leading to the marking N^{x}. Thus, by (**), $\sigma\sigma''$ and $\sigma'\sigma''$ lead to some markings M and M' in $\mathsf{box}(E)$. If, for example, $\sigma\sigma'' \neq \mathsf{box}^{\mathsf{x}}$ then, by Lemma 5(2), it can be extended by a nonempty σ''' to lead to $\mathsf{box}(E)^{\mathsf{x}}$. However, contradicting (*), σ''' cannot be fired from N^{x}. Hence, $M = M'$, and so σ and σ' lead to the same marking in $\mathsf{box}(E)$. This and the deterministic nature of $RG(\mathsf{box})$ and $RG(N)$ means that the two reachability graphs are isomorphic.

C Proofs of Propositions 2 and 3

Proposition 2 follows by a straightforward induction on the structure of E. Propositions 3 follows directly from the definitions and Proposition 2.

D Proof of Proposition 4

We first shows that interface graphs are what we need to have in order to characterise local conflict and causalities.

▶ **Lemma 6.** *Let E and F be box expressions derived from the syntax (1) with disjoint sets of actions. Then:*

$$\begin{aligned}
\mathit{Confl}(\mathsf{box}(E\,;F)) &= \mathit{Confl}(\mathsf{box}(E)) \cup \mathit{Confl}(\mathsf{box}(F)) \\
\mathit{Caused}(\mathsf{box}(E\,;F)) &= \mathit{Caused}(\mathsf{box}(E)) \cup \mathit{Confl}(\mathsf{box}(F)) \cup \\
&\quad \{\{t,u\} \in T_{\mathsf{box}(E)} \times T_{\mathsf{box}(F)} \mid \{t,u\} \in \mathit{edge}(G_E^{\mathsf{x}} — G_F^{\mathsf{e}})\} \\
\mathit{Confl}(\mathsf{box}(E \,\square\, F)) &= \mathit{Confl}(\mathsf{box}(E)) \cup \mathit{Confl}(\mathsf{box}(F)) \cup \\
&\quad \{\{t,u\} \in T_{\mathsf{box}(E)} \times T_{\mathsf{box}(F)} \mid \{t,u\} \in \mathit{edge}(G_E^{\mathsf{e}} — G_F^{\mathsf{e}})\} \\
\mathit{Caused}(\mathsf{box}(E \,\square\, F)) &= \mathit{Caused}(\mathsf{box}(E)) \cup \mathit{Caused}(\mathsf{box}(F)) \\
\mathit{Confl}(\mathsf{box}(E \parallel F)) &= \mathit{Confl}(\mathsf{box}(E)) \cup \mathit{Confl}(\mathsf{box}(F)) \\
\mathit{Caused}(\mathsf{box}(E \parallel F)) &= \mathit{Caused}(\mathsf{box}(E)) \cup \mathit{Caused}(\mathsf{box}(F))
\end{aligned}$$

Proof. The result follows directly from the definitions. ◀

We then proceed with the proof proper, using the same notations as in the formulation of Proposition 4. The first part follows directly from Proposition 1(⇐), Propositions 2 and 3, and Lemma 6. The second part follows from the minimality of covers used in the construction of N and Proposition 1(⇒).