

Modelling Resource Dependencies

EPFL IC IINFCOM RiSD Technical Report
N° EPFL-REPORT-218599
<http://infoscience.epfl.ch/record/218599>

Simon Bliudze, Simalatsar Alena and Zolotukhina Alina

June 30, 2016

Abstract: The major research in the resource management literature focuses primarily on two complementary sub-problems: 1) specification languages for formulating resource requests and 2) constraint problems modelling allocation and scheduling. Both directions assume the knowledge of the underlying platform architecture and the dependencies it induces on the usage of the various resources.

In this paper we bridge this gap, by introducing Constraint-Flow Nets (cfNets). A cfNet is defined by a set of resources and dependencies between them, each dependency having an associated constraint. The model is inspired by Petri nets, with resources corresponding to places and dependencies—to transitions. Given an architecture of dependent resources, an initial resource request is propagated through the dependencies. The generated constraints are then conjuncted into the global allocation constraint.

We study the notion of conflicts in cfNets and prove that for *conflict-free* cfNets the global allocation constraint can be constructed unambiguously. Furthermore, we provide an efficient algorithm for conflict detection.

```
@TechReport{BSZ16-cfNets,  
  author      = {Bliudze, Simon and  
                Simalatsar, Alena and  
                Zolotukhina, Alina},  
  title       = {Modelling Resource Dependencies},  
  institution = {EPFL IC IINFCOM RiSD},  
  month       = jun,  
  year        = 2016,  
  number      = {EPFL-REPORT-218599},  
  note        = {Available at: \texttt{http://infoscience.epfl.ch/record/218599}}  
}
```

Contents

1	Introduction	2
2	Modelling resource dependencies	3
2.1	Constraint-Flow Nets	3
2.2	Constraints	6
2.3	Examples of cfNets with inhibitor arcs	9
2.4	Allocation constraint problem	12
3	Conflicting dependencies	13
3.1	Conflicting transitions	13
3.2	Conflict detection	17
3.2.1	Boolean encoding of transition enabledness	18
3.2.2	Marking reachability	20
3.2.3	Encoding refinement	25
3.3	Priority	28
4	Case-study: Kalray architecture with cfNets	28
5	Conclusion	30

1 Introduction

Providing resource management is of key importance to many different areas, from embedded systems domain to distributed resource management in large-scale systems or in a cloud.

In the literature, two main complementary sub-problems are investigated: specification languages for formulating resource requests [6, 11, 18, 26] and resource management architectures [8, 9, 15, 17, 20]. The former provides application developers with the means to specify application resource requirements, whereas the latter is using the request information to build a constraint problem, which is then solved by a satisfiability modulo theories (SMT) [2, 22] or a constraint solver [25] to find a satisfactory resource allocation. However, for non-trivial architectures, this approach presents a substantial gap. Indeed, on one hand, the resource manager assumes that an application completely specifies all its resource requirements. On the other hand, specification languages provide request primitives formulated in terms of \langle required amount/resource type \rangle pairs, e.g. “5Mb of memory” or “1 thread”. Ignoring the physical nature of the resources and the dependencies among them makes it impossible for applications to define sufficiently complete resource requests. Furthermore, we argue that such completeness is not desirable. In order to avoid strong platform dependencies, applications should have the possibility to operate on a more abstract level. For a simple example, consider a multicore Network-on-Chip (NoC) platform (e.g. [14]), where each core has a dedicated local memory, but can also access that of the other cores through the NoC. Depending on the location of the requested memory and under the assumptions above, application developers must also explicitly request access to the NoC. Another example is provided by modular platforms, where resources, such as memory, channels or threads, can be created dynamically: applications should be allowed to specify requests for a certain type of resources without having the knowledge of their structure. While some advanced compilation tools, e.g. [13, 22, 23] provide ad-hoc solutions for specific target platforms, the objective of the work presented in this report is to bridge this gap in a generic manner, sufficient to describe resource dependencies for a wide class of platforms.

We consider an environment with a global set of resources \mathcal{R} and an entity (application) that makes a request for a subset of these resources. In general, the information contained in the request is not sufficient to find a satisfactory resource allocation, due to potential dependencies among the resources (in the above example, remote memory access requires the use of the NoC). To model such dependencies we introduce the notion of *Constraint-Flow Nets* (cfNets), inspired by Petri nets with inhibitor arcs. Inhibitor arcs are used to limit dependency applications (e.g. there is no need to repeat a request for a given resource, if it has already been requested). In order to specify relations between the amounts of the resources requested by the application and the necessary amounts of the resources introduced by dependencies, we associate *constraint schemata* to all transitions of a cfNet. These constraint schemata are then used to build the global constraint problem associated to the initial resource request. We prove that such global constraint problems can be unambiguously built for *conflict-free* cfNets. Furthermore, we provide a technique for detection of conflicts and their resolution by introducing priority relations among the conflicting transitions. Hence, given a cfNet with a *priority model*, the global constraint can always be built unambiguously.

The idea of using Petri nets for resource management is not novel. A number of works explore Resource Allocation Systems (RAS) in the context of Flexible Manufacturing Systems (FMS) [7] and introduce different subclasses of Petri nets to account for various allocation requirements [12, 24]. In [19] the authors investigate how the methods used in FMS can be extended to software applications with concurrent processes competing for shared resources. They propose a new subclass of Petri nets, PC²R, where the resources of different types, their availability and control flow of each process are represented as a unique system. A small change in the process flow or resources set will require a change of this model. In contrast, we are applying the separation of

concerns principle by providing distinct models for systems of interdependent resources with their available capacities and for resource requests from abstract applications.

Numerous works study constraints and dependencies, such as temporal, causality and resource constraints, in various contexts by considering the underlying dependency graphs, e.g. [1, 3, 16, 21]. Due to their syntactic structure inspired from Petri nets, cfNets generalise these approaches. Furthermore, they are specifically tailored to provide a natural way of incorporating constraint schemata, that allow expressing quantitative dependencies among amounts of allocated resources. To the best of our knowledge, such combinations of constraint schemata with an underlying graph structure have not been studied in existing literature.

The report is structured as follows. Section 2 introduces cfNets and their semantics in terms of execution and in terms of constraints modelling resource dependencies. In Section 3, we study the notion of conflict in the cfNets, and provide an algorithm for conflict detection. Section 4 presents the case study of the Kalray architecture. Section 5 concludes the report and discusses future work.

2 Modelling resource dependencies

2.1 Constraint-Flow Nets

In this section, we introduce Constraint-Flow Nets (cfNets), which we use to model resource dependencies. Syntactically, cfNets are Petri nets with inhibitor arcs. The semantics of cfNets can be compared to that of Coloured Petri nets with inhibitor arcs and capacities (each place has capacity 1 with respect to each token colour). The colour of a token in a cfNet depends on the transition that has produced this token. The main difference between cfNets and Petri nets is the following: *firing a transition does not remove tokens from its pre-places*. Therefore, the capacity restriction effectively prevents any transition from being fired more than once.

Definition 2.1. Consider a tuple $\mathcal{N} = (\mathcal{R}, T, F, I)$, where

- \mathcal{R} is a finite set of *places* (resources);
- T is a finite set of *transitions* (dependencies);
- $F \subseteq (\mathcal{R} \times T) \cup (T \times \mathcal{R})$ is a set of *arcs*; and
- $I \subseteq \mathcal{R} \times T^* \times T$, with $T^* \stackrel{\text{def}}{=} T \cup \{*\}$, for some fresh symbol $*$ $\notin T$, is a set of *inhibitor arcs*.

For $t \in T$, we denote

- by $R^-(t) \stackrel{\text{def}}{=} \{r \in \mathcal{R} \mid (r, t) \in F\}$ the set of its *pre-places*;
- by $R^+(t) \stackrel{\text{def}}{=} \{r \in \mathcal{R} \mid (t, r) \in F\}$ the set of its *post-places*.

Similarly, for $r \in \mathcal{R}$, we denote

$$T^-(r) \stackrel{\text{def}}{=} \{t \in T \mid (t, r) \in F\} \text{ the set of its } \textit{incoming} \text{ transitions.}$$

If $(r, t', t) \in I$, for some $t' \in T^*$, we say that r is an *inhibitor place* for t . Finally, we denote

$$I(t) \stackrel{\text{def}}{=} \{(r, t') \in \mathcal{R} \times T^* \mid (r, t', t) \in I\}.$$

\mathcal{N} is a *cfNet*, if

1. $R^-(t) \cap R^+(t) = \emptyset$, for any $t \in T$ (i.e. there are no looping transitions),

2. $R^-(t) \neq \emptyset$, for any $t \in T$, and
3. $t' \in T^-(r)$, for all $(r, t', t) \in I$.

As will be apparent from the following definitions, an inhibitor arc (r, t', t) checks for the absence of a token in the place r produced by the transition t' . The asterisk $*$ represents a *virtual initial transition* (cf. Definition 2.2 below).

Definition 2.2. A *marking* of a cfNet (\mathcal{R}, T, F, I) is a set of tokens $M \subseteq \mathcal{R} \times T^*$. We say that a token $(r, t) \in \mathcal{R} \times T^*$, has the *colour* t and denote $T_M \stackrel{\text{def}}{=} \{t \in T^* \mid (r, t) \in M\}$ the set of colours involved in the marking M . A marking M is *initial* if $T_M = \{*\}$.

Notice that a marking M can also be viewed as the characteristic function $M : \mathcal{R} \times T^* \rightarrow \mathbb{B}$, where we denote $\mathbb{B} = \{\mathbf{tt}, \mathbf{ff}\}$ the set of truth values *true* and *false*, respectively. Under marking M , a place $r \in \mathcal{R}$ contains a token (r, t) of colour $t \in T^*$, when $M(r, t) = \mathbf{tt}$. In the rest of the report, we will use these two notations interchangeably.

Notice also that in Coloured Petri nets the colour of the tokens is used to specify which tokens (of which colour) a transition can consume. In the cfNet semantics, each transition can consume any token regardless of its colour: colours are relevant only for post-places and inhibitor arcs of transitions.

Below, we provide the formal semantics of cfNets.

Definition 2.3. A transition $t \in T$ of a cfNet (\mathcal{R}, T, F, I) is *enabled* with a marking M if the following three conditions hold:

1. for all $r \in R^-(t)$, there is a token $(r, t') \in M$ for some $t' \in T^-(r) \cup \{*\}$;
2. for all $r \in R^+(t)$, the corresponding token is not in M , i.e. $(r, t) \notin M$;
3. for all $(r, t', t) \in I$, the corresponding token is not in M , i.e. $(r, t') \notin M$.

Definition 2.4. A marking M' of a cfNet (\mathcal{R}, T, F, I) is *reachable from a marking M in one step* if there exists a transition $t \in T$, enabled with M , such that, for all $r \in \mathcal{R}$ and all $t' \in T^*$, holds the following equality:

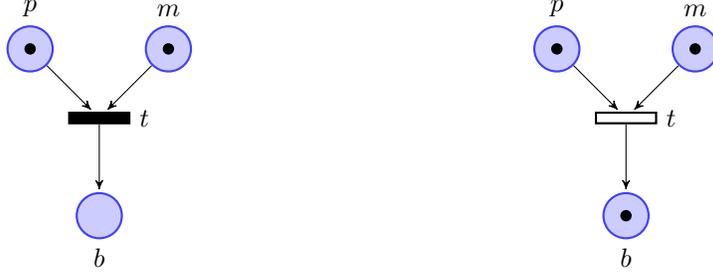
$$M'(r, t') = \begin{cases} \mathbf{tt}, & \text{if } r \in R^+(t) \text{ and } t' = t, \\ M(r, t'), & \text{otherwise.} \end{cases}$$

This is denoted by $M \xrightarrow{t} M'$.

Notice that this definition of a step indeed implies that, contrary to classical semantics of Petri nets, transitions do not remove tokens from their pre-places.

Definition 2.5. A marking M is *final* if there are no transitions enabled with M .

In the rest of the report, we use the following convention for the graphical representation of cfNets: transitions that have not been fired are shown in black, whereas transitions that have already been fired—and therefore cannot be fired again—are shown in white (cf. Figure 1a and Figure 1b). Moreover, for the sake of clarity, when there are several inhibitor arcs from the same place to the same transition labelled with different colours, we draw only one arc and annotate it with all the colours. Finally, in all the illustrations in the report, the colour of a token can be unambiguously derived by considering which transitions have been fired (visible from the black or white colour of the transition in the diagram). Therefore, we use the usual graphical notation for tokens in Petri nets, i.e. a bullet within the corresponding place.



(a) Initial marking for the request $\{p, m\}$ (b) Final marking for the request $\{p, m\}$

Figure 1: The cfNet modelling the dependency from Example 2.6

Example 2.6 (Memory and Bus). Consider a system with one processor and one memory unit connected by a bus. Whenever an application uses the processor to access the memory, access to the bus is required implicitly. This dependency is modelled by the cfNet shown in Figure 1. The cfNet has three places p , m and b , corresponding to the processor, the memory and the bus. The resource dependency is modelled by the transition t with incoming arcs from p and m , and one outgoing arc to b .

Consider an initial resource request $R = \{p, m\}$. The corresponding initial marking M_0 of the cfNet has two tokens: $(p, *)$ and $(m, *)$ (Figure 1a). Transition t is enabled and can be fired, generating the token (b, t) . Thus, we have $M_0 \xrightarrow{t} M$ with M shown in Figure 1b. Since t is not enabled with M , this marking is final. \diamond

Definition 2.7. A *run* of a cfNet from a marking M_0 is a sequence $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} \dots \xrightarrow{t_n} M_n$. When such a run exists, we say that M_n is *reachable from* M_0 and write $M_0 \xrightarrow{\langle t_1, \dots, t_n \rangle} M_n$. We say that a marking is *reachable* if it is reachable from some initial marking.

Notice that, for any marking M obtained by firing a sequence of transitions, T_M (see Definition 2.2) is the set comprising $*$ and these transitions (see Proposition 2.10 below).

Example 2.8 (Two Memories and two Buses). Consider a system with one processor connected to two memory units via two different buses. The dependency between resources is the same as in the previous example: whenever an application uses the processor to access a memory, access to the corresponding bus is required implicitly. This dependency is modelled by the cfNet shown in Figure 2. The cfNet has five places p , m_1 , m_2 , b_1 and b_2 , corresponding to the processor, the two memories and the two buses. The resource dependencies are modelled by the transition t_1 (resp. t_2) with incoming arcs from p and m_1 (resp. p and m_2), and one outgoing arc to b_1 (resp. b_2).

Consider an initial resource request $R = \{p, m_1\}$. Similarly to the previous example, transition t_1 is enabled and can be fired, generating a token in the place b_1 and thus creating a final marking.

Now consider an initial resource request $R = \{p, m_1, m_2\}$. The corresponding initial marking M_0 of the cfNet has three tokens: $(p, *)$, $(m_1, *)$ and $(m_2, *)$ (Figure 2a). Both transitions t_1 and t_2 are enabled and can be fired. Let t_1 fire first, generating a token (b_1, t_1) . Thus, we have $M_0 \xrightarrow{t_1} M_1$ with M_1 shown in Figure 2b. Transition t_1 is disabled, but transition t_2 is enabled and can fire, generating a token (b_2, t_2) . We have $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2$, with M_2 being the final marking (Figure 2c) and $T_{M_2} = \{*, t_1, t_2\}$.

The situation when t_2 fires first before t_1 is symmetrical. \diamond

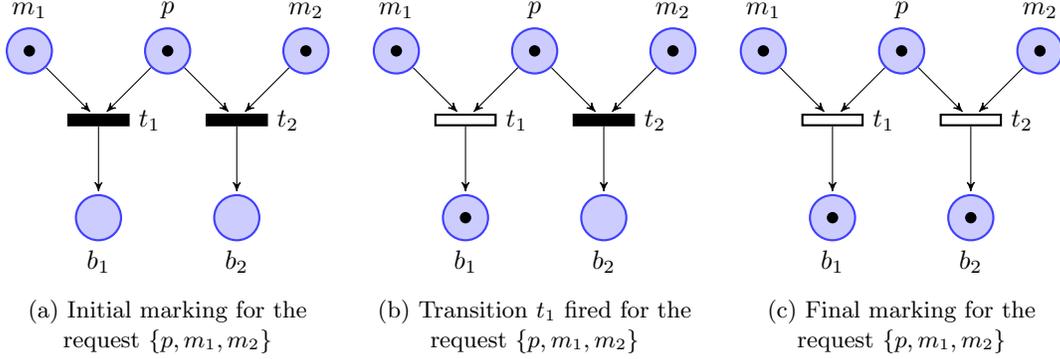


Figure 2: The cfNet modelling the dependency from Example 2.8

Definition 2.9. A marking M of a cfNet (\mathcal{R}, T, F, I) is *well-formed* if, for all $t \in T_M \setminus \{*\}$, the following three conditions hold:

1. for all $r \in R^-(t)$, there exists a token $(r, t') \in M$ for some $t' \in T^-(r) \cup \{*\}$;
2. for all $r \in R^+(t)$, $(r, t) \in M$;
3. for all $(r, *) \in I(t)$, $(r, *) \notin M$.

In Definition 2.9, conditions 1 and 3 are necessary for the transition t to have been enabled. They are not sufficient, since, for the transition to be enabled, inhibitor tokens referring to colours other than $*$ must also be absent from the marking. However, we cannot include this stronger requirement in the definition of well-formedness. Indeed, such inhibitor tokens can appear once t has already been fired. Condition 2 requires that all the tokens generated by firing t be, indeed, present in the marking.

Proposition 2.10. Let M_0 be an initial marking. Let M' be a marking reachable from M_0 with $M_0 \xrightarrow{\langle t_1, \dots, t_n \rangle} M'$. Then M' is well-formed and $T_{M'} = \{*, t_1, \dots, t_n\}$.

Proof. Let $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} \dots \xrightarrow{t_n} M_n$ with $M_n = M'$. By Definition 2.2, the initial marking M_0 contains only tokens $(r, *)$, for some $r \in \mathcal{R}$. By Definition 2.4, for $i \in [1, n]$, each t_i creates in M_i tokens of the form (r, t_i) , for some $r \in \mathcal{R}$. No other tokens are created. Thus, $\Sigma_{M'} = \{*, t_1, \dots, t_n\}$.

Each $t_i \in T_{M'} \setminus \{*\}$ is enabled with M_{i-1} . Hence, by Definition 2.3, for all $r \in R^-(t_i)$ there exists a token $(r, t') \in M_{i-1}$ for some $t' \in T^-(r) \cup \{*\}$. Since transitions can only add tokens to a marking, we have $(r, t') \in M'$. Since t_i has been fired to reach M_i , we have $(r, t_i) \in M_i$, for each $r \in R^+(t_i)$. Hence, $(r, t_i) \in M'$. Furthermore, none of the tokens inhibiting t_i are present in M_{i-1} . In particular, $(r, *) \notin M_{i-1}$, for any $(r, *) \in I(t_i)$. Since initial tokens cannot be added by firing a transition, this implies $(r, *) \notin M'$. Thus, by Definition 2.9, M' is well-formed. \square

The well-formedness of a marking is an overapproximation of reachability: all reachable markings are well-formed, but there exist well-formed markings which are not reachable.

2.2 Constraints

To account for quantitative dependencies between resources, we extend the definition of cfNets by associating to each transition a constraint schema, which is instantiated into a constraint for a final marking of the cfNet. These constraints are then used to build the global allocation constraint problem, as shown in Section 2.4.

We assume that each resource $r \in \mathcal{R}$ has a value domain D_r , where D_r is a commutative monoid, i.e. it has an addition operation $+$ with an identity element 0 .

Definition 2.11. Consider a cfNet (\mathcal{R}, T, F, I) . For any transition $t \in T$, denote

$$X_t \stackrel{\text{def}}{=} \{x_r \mid r \in R^-(t) \cup R^+(t)\},$$

where each x_r is a variable with the value domain D_r . A *constraint schema* associated to t is a predicate over the set of variables X_t , i.e.

$$c_t : \prod_{r \in R^-(t) \cup R^+(t)} D_r \rightarrow \mathbb{B}.$$

Definition 2.12. A *cfNet with constraints* is a tuple $(\mathcal{R}, T, F, I, \mathcal{C})$, where (\mathcal{R}, T, F, I) is a cfNet and $\mathcal{C} = \{c_t \mid t \in T\}$ is a set of constraint schemata associated to the transitions in T .

In the rest of the report, we will only consider cfNets with constraints. For the sake of conciseness, we will refer to them simply as cfNets.

We build global constraint problems encoding resource allocations compatible with the causal dependencies defined by the cfNet. A constraint problem is based on the initial resource request of the application and the constraint schemata associated to transitions constituting a run of the cfNet. To this end, we introduce, for each place-colour pair $(r, t) \in \mathcal{R} \times T^*$, a variable d_r^t with the domain value D_r .

Definition 2.13. Let M be a well-formed marking of a cfNet $(\mathcal{R}, T, F, I, \mathcal{C})$. We define a *platform constraint*

$$C[M] \stackrel{\text{def}}{=} \bigwedge_{t \in T_M} c_t \left[\left(\sum_{t': (r, t') \in M} d_r^{t'} \right) / x_r \mid r \in R^-(t) \right] \left[d_r^t / x_r \mid r \in R^+(t) \right], \quad (1)$$

where, for an expression E , we denote by $E[x/y \mid C]$ the expression obtained by substituting in E all occurrences of y , which satisfy the condition C , by x . Thus each conjunct in (1) is the predicate obtained by replacing, in the corresponding constraint schema c_t ,

- for each pre-place $r \in R^-(t)$, the variable x_r with the sum of all variables $d_r^{t'}$ corresponding to all the tokens $(r, t') \in M$;
- for each post-place $r \in R^+(t)$, the variable x_r with the corresponding variable d_r^t .

Notice that the conjuncts in (1) are unambiguously defined, since, by Definition 2.1, there are no looping transitions in the cfNet, i.e. $R^-(t) \cap R^+(t) = \emptyset$, for all $t \in T$. Hence the two substitutions operate on disjoint sets of variables.¹

Example 2.14 (Memory and Bus—continued). Building on Example 2.6, we introduce the constraint linking the actual resource requirements. Since any data to be written or read from the memory must transit through the bus, we associate to the transition t a constraint schema imposing that the required bus capacity be greater than or equal to the requested amount of memory:

$$c_t = x_b \geq x_m.$$

¹Strictly speaking these conjuncts are unambiguously defined even without this restriction: the application of the second substitution will have no effect on the variables that have been substituted by the first. However, it is more elegant to have this separation explicitly.

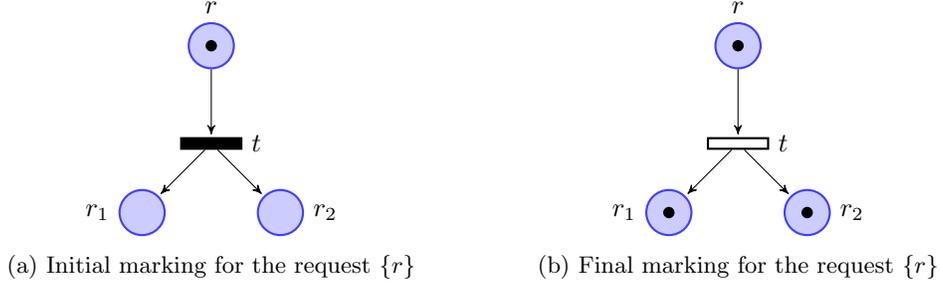


Figure 3: The cfNet modelling the dependency from Example 2.15

Consider again the initial request $R = \{p, m\}$ with the corresponding initial marking in Figure 1a. The variables corresponding to the initial tokens are d_p^* and d_m^* . Since the final marking M in Figure 1b contains a token (b, t) , we also introduce the corresponding variable d_b^t . Substituting these variables in the constraint schema for t , we obtain the platform constraint

$$C[M] = c_t[d_p^*/x_p, d_m^*/x_m, d_b^t/x_b] = d_b^t \geq d_m^*.$$

◇

Example 2.15 (Virtual resources). Consider a system with two physical resources of the same type, r_1 and r_2 . For example, these could be two storage devices. Applications can access any of the two directly or, alternatively, request a certain amount without specifying which of r_1 and r_2 must provide it. This can be modelled by introducing a “virtual” resource r .

In case when the virtual resource is requested, the actual allocation depends on the policy that the system implements, for instance:

1. one of r_1 and r_2 must provide the requested amount (dispatching the request);
2. both r_1 and r_2 must provide the requested amount (redundant allocation);
3. part of the requested amount is provided by one of the two physical resources and the rest is provided by the other (joint allocation).

In all the three cases, the utilization of r requires the utilization of both r_1 and r_2 . In case of dispatching the request, only one of the resources is actually allocated. However, at design time, it is not known which one will be allocated. Therefore, both resources have to be included in the constraint.

Figure 3 shows a cfNet modelling the above dependencies. The constraint scheme of the transition depends on the policy:

1. for dispatching the request: $c_t = (x_{r_1} = x_r \wedge x_{r_2} = 0) \vee (x_{r_1} = 0 \wedge x_{r_2} = x_r)$,
2. for redundant allocation: $c_t = (x_{r_1} = x_r \wedge x_{r_2} = x_r)$,
3. for joint allocation: $c_t = (x_{r_1} + x_{r_2} = x_r)$.

Consider the initial request $R = \{r\}$ with the joint allocation policy. The corresponding initial marking M_0 is shown in Figure 3a. The variable corresponding to the initial token is d_r^* . Since the final marking M in Figure 3b contains tokens (r_1, t) and (r_2, t) , we also introduce the

corresponding variables $d_{r_1}^t$ and $d_{r_2}^t$. Substituting these variables in the constraint schema for t for the joint allocation policy, we obtain the platform constraint

$$C[M] = c_t [d_r^*/x_r, d_{r_1}^t/x_{r_1}, d_{r_2}^t/x_{r_2}] = (d_{r_1}^t + d_{r_2}^t = d_r^*).$$

◇

Example 2.16 (Processor and Virtual Memory). This example combines Example 2.8 and Example 2.15. We again consider a system with one processor connected to two memory units via two different buses. However now we add a virtual memory resource representing the two memories, which gives the possibility to ask for a memory without specifying which one is needed exactly. The other dependencies between resources are the same as in Example 2.8: whenever an application uses the processor to access a memory, access to the corresponding bus is required implicitly.

These dependencies are modelled by the cfNet shown in Figure 4. Assume that transition t implements dispatching policy. The constraint schemata for the transitions then are:

- $c_t = (x_{m_1} = x_m \wedge x_{m_2} = 0) \vee (x_{m_1} = 0 \wedge x_{m_2} = x_m)$,
- $c_{t_1} = x_{b_1} \geq x_{m_1}$,
- $c_{t_2} = x_{b_2} \geq x_{m_2}$.

Consider an initial resource request $R = \{p, m\}$. Two token variables are created: d_p^* and d_m^* . The corresponding marking is shown in Figure 4a. Transition t is enabled and fires, generating tokens which enable transitions t_1 and t_2 , alongside with corresponding token variables $d_{m_1}^t$ and $d_{m_2}^t$ (Figure 4b). Let transition t_1 fire first, generating a token in b_1 and a token variable $d_{b_1}^{t_1}$. Finally, t_2 fires, generating a token $d_{b_2}^{t_2}$ in b_2 .

Substituting these variables in the constraint schemata, we obtain the platform constraint

$$C[M] = c_t [d_m^*/x_m, d_{m_1}^t/x_{m_1}, d_{m_2}^t/x_{m_2}] \wedge c_{t_1} [d_p^*/x_p, d_{m_1}^{t_1}/x_{m_1}, d_{b_1}^{t_1}/x_{b_1}] \wedge c_{t_2} [d_p^*/x_p, d_{m_2}^{t_2}/x_{m_2}, d_{b_2}^{t_2}/x_{b_2}] =$$

$$((d_{m_1}^t = d_m^* \wedge d_{m_2}^t = 0) \vee (d_{m_1}^t = 0 \wedge d_{m_2}^t = d_m^*)) \wedge (d_{b_1}^{t_1} \geq d_{m_1}^t) \wedge (d_{b_2}^{t_2} \geq d_{m_2}^t).$$

◇

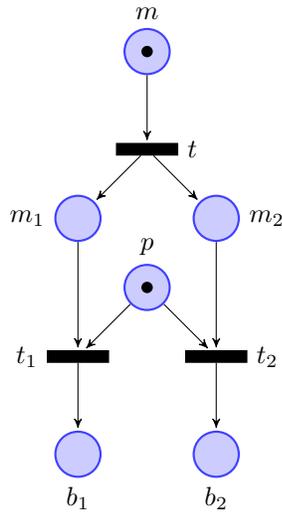
2.3 Examples of cfNets with inhibitor arcs

In this section, we provide some additional examples illustrating the use of inhibitor arcs.

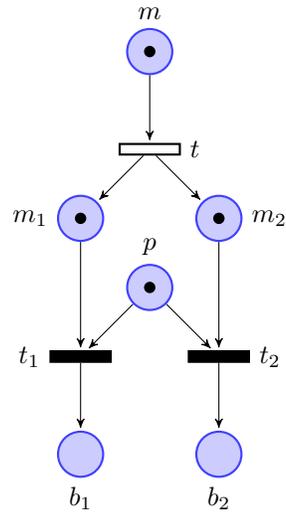
Example 2.17 (Two interdependent resources). Consider a system with three resources r_1, r_2 and r_3 , such that 1) r_1 and r_2 are mutually dependent, i.e. if only r_1 is requested, r_2 must be added to the request, and vice versa; and 2) when both r_1 and r_2 are requested, r_3 must also be allocated. These dependencies are modelled by a cfNet shown in Figure 5.

Consider the transition t_1 . Apart from the regular arcs, it has an inhibitor arc from r_2 , i.e. we have $(r_2, *, t_1) \in I$, which prevents an additional request for r_2 when this latter is already part of the initial request. The transition t_2 has a symmetric inhibitor arc from r_1 .

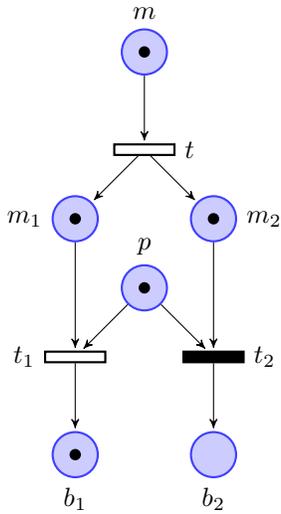
First consider the initial request $\{r_1\}$ (Figure 5a). Since there are no tokens in r_2 , transition t_1 is enabled. After firing t_1 , a new token (r_2, t_1) is added to the place r_2 (Figure 5b). Both t_1 and t_2 are disabled with this marking: t_2 due to the influence of the corresponding inhibitor arc, t_1 due to the presence of the token (r_2, t_1) . Thus, the only enabled transition is t_3 . After firing t_3 , a new token appears in the place r_3 and the cfNet reaches the final marking shown in Figure 5c. Assuming



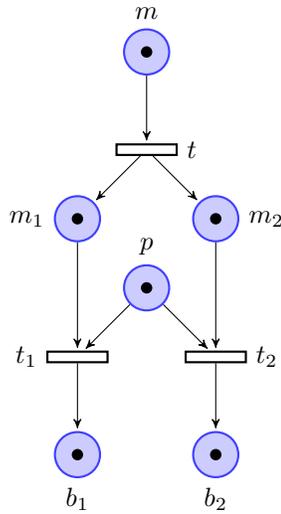
(a) Initial marking for the request $\{p, m\}$



(b) Transition t fired for the request $\{p, m\}$



(c) Transition t_1 fired for the request $\{p, m\}$



(d) Final marking for the request $\{p, m\}$

Figure 4: The cfNet modelling the dependency from Example 2.16

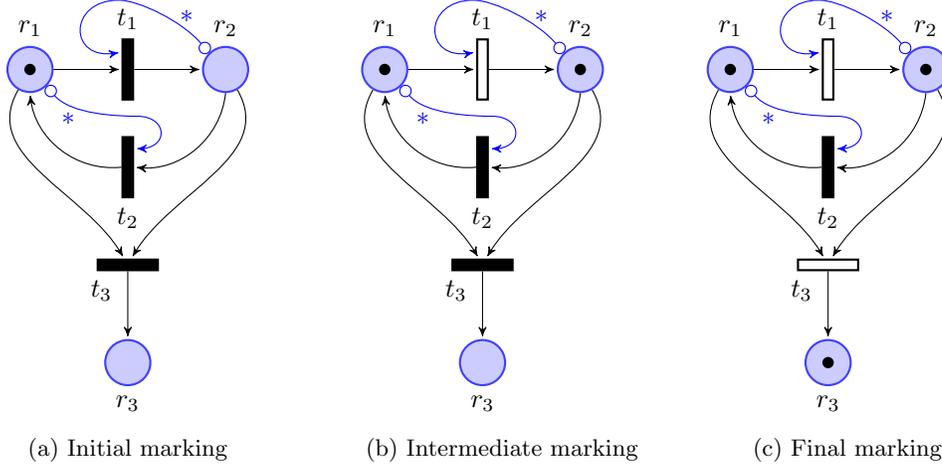


Figure 5: cfNet illustrating Example 2.17 with the successive markings for the request $\{r_1\}$

that constraint schemata associated to t_1 , t_2 and t_3 are, respectively, c_1 , c_2 and c_3 , we obtain, for this final marking, the platform constraint $c_1[d_{r_1}^*/x_{r_1}, d_{r_2}^{t_1}/x_{r_2}] \wedge c_3[d_{r_1}^*/x_{r_1}, d_{r_2}^{t_1}/x_{r_2}, d_{r_3}^{t_3}/x_{r_3}]$.

Consider now the initial request $\{r_1, r_2\}$. The corresponding marking has both tokens $(r_1, *)$ and $(r_2, *)$. Hence both transitions t_1 and t_2 are disabled by the corresponding inhibitor arcs. Only t_3 is enabled and is fired once before reaching a final marking. The resulting platform constraint is $c_3[d_{r_1}^*/x_{r_1}, d_{r_2}^*/x_{r_2}, d_{r_3}^{t_3}/x_{r_3}]$. \diamond

Example 2.18 (Inhibitors referring to transitions). In the previous example each of the interdependent resources could be required only once. However, a different situation can also exist: it might be possible that a resource can be explicitly requested by an application, and additional amounts of the same resource are requested due to dependencies.

Consider a situation when there are different dependency paths through the cfNet to this resource, and once the resource is requested due to one dependency, it should not be requested again due to other dependencies.

The cfNet for a system with such a resource r is shown in Figure 6:

1. there is a place r with two incoming transitions t_1 and t_2 ;
2. there are some other places and dependencies, different for t_1 and t_2 , denoted by a cloud;
3. there is a inhibitor arc from r to t_1 referring to t_2 and to t_2 referring to t_1 .

Consider an initial request with no token in r which leads to both t_1 and t_2 being enabled (Figure 6a). Let t_1 fire. After the firing, a new token in r is created (Figure 6b). This token comes from transition t_1 , therefore, transition t_2 is disabled because of the inhibitor arc. The platform constraint then contains $c_1[d_r^{t_1}/x_r, \dots]$ as a conjunct, where the dots stand for other resources of the cfNet required for t_1 to fire.

The case when transition t_2 fires instead of t_1 is symmetrical (Figure 6c).

Consider an initial request with a token in r which leads to both t_1 and t_2 being enabled. This can happen since even though there is a token in r , it is an initial token, and inhibitor arcs refer to tokens having come from transitions. After the firing of, for example, t_2 , a new token in r is created. Transition t_1 is disabled because of the inhibitor arc, and the platform constraint contains $c_2[(d_r^* + d_r^{t_2})/x_r, \dots]$ as a conjunct. \diamond

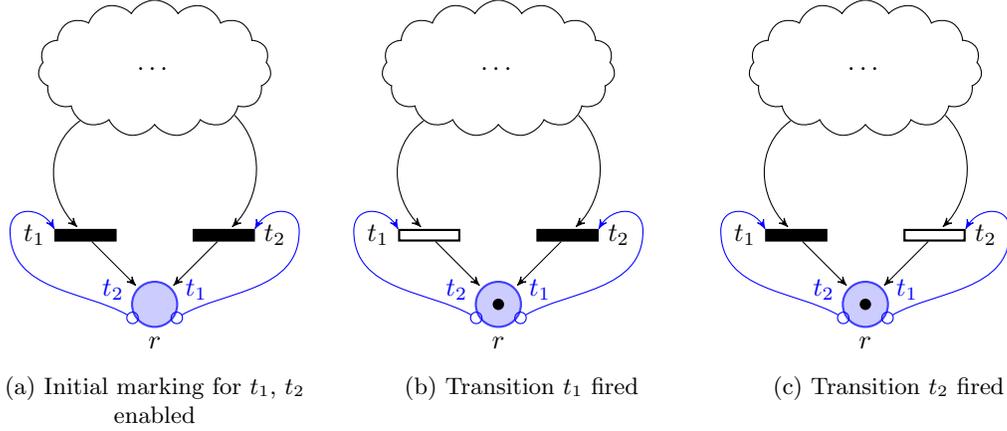


Figure 6: cfNet for Example 2.18

2.4 Allocation constraint problem

In the following, we assume that a partial cost function $cost_r : D_r \rightarrow \mathbb{R}$ is associated with each resource $r \in \mathcal{R}$. When defined, the value $cost_r(d)$ represents the cost of allocating the amount $d \in D_r$ of the resource r . When $cost_r(d)$ is undefined, this means that it is not possible to allocate the amount d of the resource r (e.g. d is greater than the resource capacity).

Example 2.19 (Memory cost). Consider a resource m , representing some abstract memory with the capacity M . The value domain is $D_m = \mathbb{N}$. Assume that the cost of memory usage is linear in the size of the allocated block. This assumption can be modelled by the following cost function:

$$cost_m(d) = \begin{cases} \frac{d}{M}, & 0 \leq d \leq M, \\ \perp, & \text{otherwise.} \end{cases}$$

◇

Example 2.20 (Processor cost). Consider a resource p , representing a processor. For a processor, the value domain represents the number of running applications. Hence, again $D_p = \mathbb{N}$. The following cost function does not affect the cost of using the allocated resources, but restricts the use of the processor to at most one application:

$$cost_p(d) = \begin{cases} 0, & d \in \{0, 1\}, \\ \perp, & \text{otherwise.} \end{cases}$$

◇

Definition 2.21. Let $R \subseteq \mathcal{R}$ be a set of resources. A *utility function* over R is a partial function $u : \prod_{r \in \mathcal{R}} D_r \rightarrow \mathbb{R}$ such that u is constant on all D_r for $r \notin R$ (i.e. u depends only on resources belonging to R).

Definition 2.22. An *allocation* over a set of resources $R \subseteq \mathcal{R}$ is a value $d = (d_r)_{r \in \mathcal{R}} \in \prod_{r \in \mathcal{R}} D_r$, such that $d_r = 0$ for all $r \notin R$.

Definition 2.23. Consider a system of resource dependencies defined by a cfNet N . Let $R_0 \subseteq \mathcal{R}$ be a set of resources corresponding to an initial request, and let u be a utility function over R_0 . Let $M_0 = \{(r, *) \mid r \in R_0\}$ be the initial marking corresponding to R_0 and let M be a final marking

obtained by running N . Let $C[M]$ be the corresponding platform constraint (see Definition 2.13). Finally, let $cost_r$, for all $r \in \mathcal{R}$, be the corresponding cost functions. An allocation $d = (d_r)_{r \in \mathcal{R}}$ over R is *valid*, if the predicate

$$C_M(d) \stackrel{def}{=} C[M] \wedge \bigwedge_{r \in \mathcal{R}} \left(d_r = \sum_{t: (r,t) \in M} d_r^t \right) \quad (2)$$

evaluates to true and if the following value is defined:

$$U_M(d) \stackrel{def}{=} u(d) - \sum_{r \in \mathcal{R}} cost_r(d_r). \quad (3)$$

We call the function $U_M(d)$ the *global utility* of the allocation d .

The problem of finding an optimal resource allocation for a request R_0 is modelled by the following problem of constrained optimisation:

$$\operatorname{argmax}_{\{d \mid C_M(d)\}} U_M(d). \quad (4)$$

Notice that both the notions of validity and global utility, and the optimisation problem above depend on the marking M obtained by running the cfNet. In the next section we characterise those cfNets, where this dependency does not hold and provide a disambiguating mechanism for the rest of cfNets.

3 Conflicting dependencies

In the previous sections, we have demonstrated how a constraint problem is built for a given run of a cfNet. However, Example 3.1 below shows that, in general, a cfNet can have several distinct runs starting with the same initial marking, due to conflicts between transitions.

Example 3.1 (Inhibitor causing conflict). Consider the cfNet shown in Figure 7. Notice that the inhibitor arc (r_3, t_2, t_3) refers only to the token (r_3, t_2) , but not to $(r_3, *)$, nor (r_3, t_1) .

First, consider the request $\{r_1, r_4\}$, corresponding to the initial marking shown in Figure 7a. The enabled transitions are t_1 and t_3 . Firing the transition t_1 generates a new token (r_3, t_1) (Figure 7b). Since the inhibitor arc of the transition t_3 refers only to (r_3, t_2) , t_3 remains enabled. After the firing of t_3 , a new token (r_5, t_3) appears in the place r_5 and the cfNet reaches the final marking shown in Figure 7c. It is easy to see that, if t_3 were to be fired before t_1 , the final marking (hence also the platform constraint) would be the same.

Now, let us consider the request $\{r_2, r_4\}$, corresponding to the initial marking shown in Figure 8a. Again, two transitions are enabled: t_2 and t_3 . However, if t_2 fires first, t_3 becomes disabled due to the influence of the inhibitor arc and the obtained marking is final (Figure 8b). On the contrary, if t_3 fires before t_2 , the final marking (hence, also the platform constraint) is different (Figure 8c). \diamond

In the rest of this section, we show that a cfNet is guaranteed to be conflict-free if it does not contain any inhibitor arcs referring to token colours other than $*$. We then provide methods for the detection of conflicts in cfNets that do contain such inhibitors.

3.1 Conflicting transitions

Definition 3.2. A cfNet $(\mathcal{R}, T, F, I, \mathcal{C})$ under a marking M has a *conflict*, if there exist two distinct enabled transitions $t_1, t_2 \in T$, such that $M \xrightarrow{t_1} M'$ and t_2 is disabled with M' .

We also say that *transitions $t_1, t_2 \in T$ are in conflict under the marking M* . A cfNet is *conflict-free* if it does not have conflicts under any reachable marking.

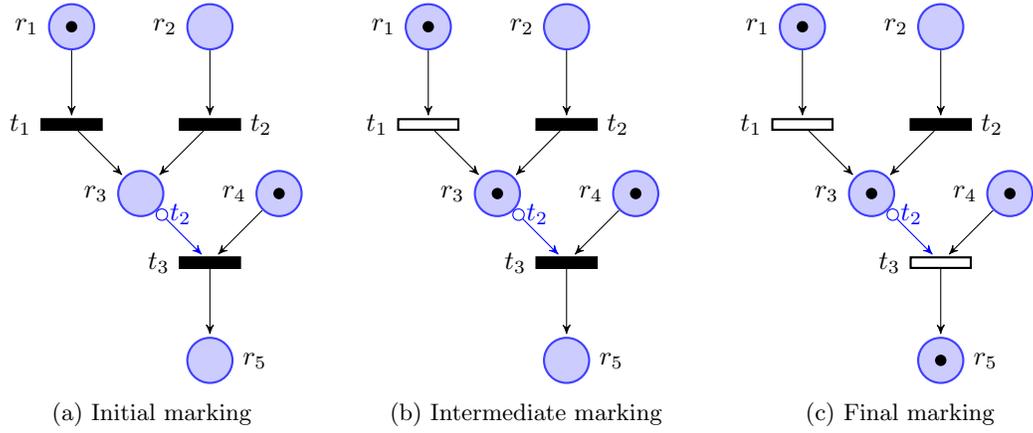


Figure 7: cfNet illustrating Example 3.1 with the successive markings for the request $\{r_1, r_4\}$

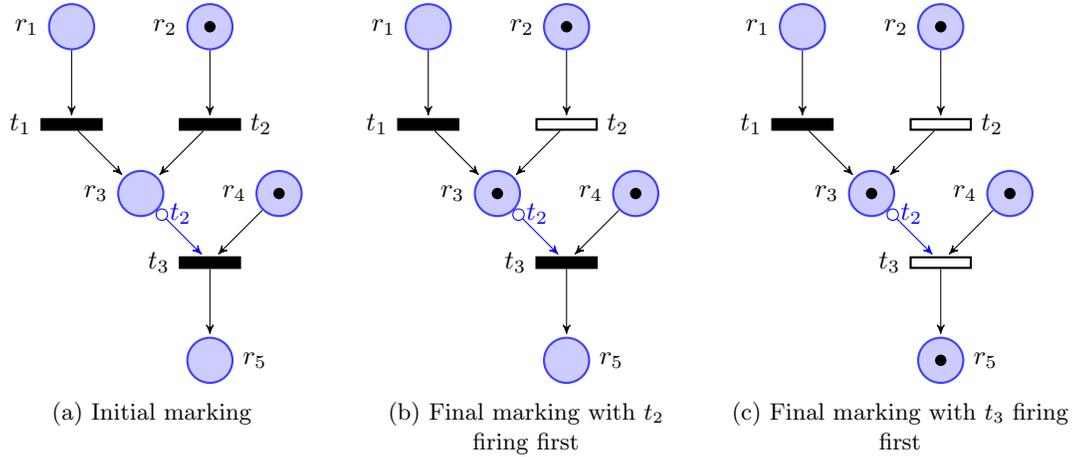


Figure 8: cfNet illustrating Example 3.1 for the request $\{r_2, r_4\}$

An example of a cfNet with conflicts is shown in Figure 8 and described in Example 3.1.

Proposition 3.3. *Any transition t , enabled with a reachable marking M_0 of a conflict-free cfNet, will be enabled with any marking reachable from M_0 without firing t .*

Proof. Let $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} \dots \xrightarrow{t_n} M_n$ be the shortest sequence such that $t \notin \{t_1, \dots, t_n\}$ and t is not enabled in M_n . Hence, t is enabled in M_{n-1} and in conflict with t_n , which contradicts the proposition assumption. \square

An important consequence of Proposition 3.3 is that a platform constraint obtained by running a conflict-free cfNet depends only on the initial marking. Indeed, for a given initial marking the runs of the cfNet can only differ in the order of transition firing. However, the set of transitions is the same, generating the same conjuncts contributing to the platform constraint (1).

Proposition 3.4. *Let $(\mathcal{R}, T, F, I, C)$ be a cfNet and $t_1, t_2 \in T$ (with $t_1 \neq t_2$) be two transitions in conflict under some marking M . Then there exists a place $r \in \mathcal{R}$, such that either $(r, t_2, t_1) \in I$ or $(r, t_1, t_2) \in I$.*

Proof. Suppose that, for all $r \in \mathcal{R}$, $(r, t_1, t_2) \notin I$ and let $M \xrightarrow{t_1} M'$. Transition t_2 is enabled with M . Hence, by Definition 2.3, we have

- for all $r \in R^-(t_2)$, there exists $(r, t') \in M$ for some $t' \in T^-(r) \cup \{*\}$, hence also $(r, t') \in M'$;
- for all $r \in R^+(t_2)$, $(r, t_2) \notin M$, hence also $(r, t_2) \notin M'$, since $t_2 \neq t_1$ and, by Definition 2.4, we have $M'(r, t_2) = M(r, t_2)$ for all $r \in \mathcal{R}$;
- for all $(r, t, t_2) \in I$, $M(r, t) = \mathbf{ff}$ and, therefore, $M'(r, t) = \mathbf{ff}$, since $(r, t_1, t_2) \notin I$.

Therefore, transition t_2 is enabled under M' , which contradicts the assumption that the two transitions are in conflict under M . A symmetrical argument on $M \xrightarrow{t_2} M''$ proves the lemma. \square

Corollary 3.5. *A cfNet $(\mathcal{R}, T, F, I, C)$, such that $t' = *$, for all $(r, t', t) \in I$, is conflict-free.*

Proof. Suppose that the cfNet is not conflict-free. Then, by Definition 3.2, there exist two transitions $t_1 \neq t_2$, in conflict under some marking M . By Proposition 3.4, there exists a place $r \in \mathcal{R}$, such that either $(r, t_2, t_1) \in I$ or $(r, t_1, t_2) \in I$. However, this contradicts the corollary assumption, since both $t_1 \neq *$ and $t_2 \neq *$. \square

Corollary 3.5 provides a simple condition guarantying that a cfNet is conflict-free. Notice, however, that this proposition does not rely on the reachability of markings. Indeed, a conflict-free cfNet can still have conflicting transitions, provided that they are not enabled together under any reachable marking.

Definition 3.6. Transitions t_1 and t_2 are *mutually exclusive* if there is no reachable marking that enables them both.

Figure 9 shows an example of two mutually exclusive transitions. Transitions t_1 and t_2 cannot be enabled simultaneously, since the place r_2 has a regular arc to t_2 and an inhibitor arc to t_1 , thus one transition requires a token in r_2 while the other requires the place to be empty.

Definition 3.7. An inhibitor arc (r, t', t) (with $t' \neq *$) is called *non-conflicting* if t is mutually exclusive with t' .

In Figure 9, the inhibitor arc (r_3, t_1, t_2) is non-conflicting.

Lemma 3.8. *A cfNet where all inhibitor arcs are non-conflicting is conflict-free.*

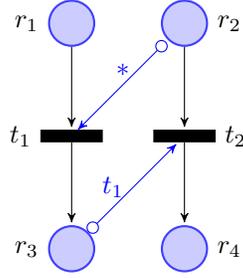


Figure 9: A simple net with mutually exclusive transitions

Proof. Let $(\mathcal{R}, T, F, I, \mathcal{C})$ be a cfNet with only non-conflicting inhibitor arcs. By Corollary 3.5, a conflict may be introduced only by some inhibitor arc $(r, t', t) \in I$, such that $t' \neq *$. By the lemma assumption, (r, t', t) is non-conflicting, i.e. t is mutually exclusive with t' . Therefore, the cfNet is conflict-free. \square

Lemma 3.9. *Let $N = (\mathcal{R}, T, F, I, \mathcal{C})$ be a cfNet and $t_1, t_2 \in T$ be two non-mutually-exclusive transitions, such that $(r, t_1, t_2) \in I$. Then N has a conflict.*

Proof. Since t_1 and t_2 are not mutually exclusive, there exists a reachable marking M enabling both t_1 and t_2 . Let $M \xrightarrow{t_1} M'$. By Definition 2.4, $M'(r, t_1) = \mathbf{tt}$, therefore, t_2 is not enabled with M' and, consequently, N has a conflict. \square

Theorem 3.10. *A cfNet is conflict-free if and only if all its inhibitor arcs refer to initial tokens or are non-conflicting.*

Proof. Follows immediately from Corollary 3.5 and Lemmas 3.8, 3.9. \square

Lemma 3.11. *Let $(\mathcal{R}, T, F, I, \mathcal{C})$ be a conflict-free cfNet, M be a marking enabling two transitions $t_1 \neq t_2$ and $M \xrightarrow{t_1} M'_1 \xrightarrow{t_2} M'_2$ and $M \xrightarrow{t_2} M''_1 \xrightarrow{t_1} M''_2$ be two possible runs of N . Then $M'_2 = M''_2$.*

Proof. Since the cfNet has no conflicts and $t_1 \neq t_2$, the statement of the lemma is valid: in the first run, t_2 is enabled with M'_1 and, in the second run, t_1 is enabled with M''_1 .

For all $r \in \mathcal{R}$ and for all $t \in T^*$:

$$\begin{aligned}
 M'_1(r, t) &= \begin{cases} \mathbf{tt}, & \text{if } t = t_1 \text{ and } r \in R^+(t_1), \\ M(r, t), & \text{otherwise,} \end{cases} \\
 M'_2(r, t) &= \begin{cases} \mathbf{tt}, & \text{if } t = t_2 \text{ and } r \in R^+(t_2), \\ M'_1(r, t), & \text{otherwise,} \end{cases} \\
 &= \begin{cases} \mathbf{tt}, & \text{if } t = t_1 \text{ and } r \in R^+(t_1), \\ \mathbf{tt}, & \text{if } t = t_2 \text{ and } r \in R^+(t_2), \\ M(r, t), & \text{otherwise.} \end{cases}
 \end{aligned}$$

On the other hand, we also have

$$\begin{aligned}
M_1''(r, t) &= \begin{cases} \mathbf{tt}, & \text{if } t = t_2 \text{ and } r \in R^+(t_2), \\ M(r, t), & \text{otherwise,} \end{cases} \\
M_2''(r, t) &= \begin{cases} \mathbf{tt}, & \text{if } t = t_1 \text{ and } r \in R^+(t_1), \\ M_1''(r, t), & \text{otherwise,} \end{cases} \\
&= \begin{cases} \mathbf{tt}, & \text{if } t = t_1 \text{ or } r \in R^+(t_1) \\ \mathbf{tt}, & \text{if } t = t_2 \text{ or } r \in R^+(t_2) \\ M(r, t), & \text{otherwise.} \end{cases}
\end{aligned}$$

Therefore, $M_2' = M_2''$. □

Lemma 3.11 means that, in a conflict-free cfNet, permutation of transitions preserves the resulting marking.

Theorem 3.12. *A conflict-free cfNet is terminating and confluent, i.e. for any initial marking M_0 , there exists a unique final marking M reachable from M_0 .*

Proof. Each transition can be fired only once and the number of transitions is finite, therefore, the cfNet is terminating. Hence, there exists at least one final marking. Let M_1 and M_2 be two final markings, such that $M_0 \xrightarrow{\langle t_1, \dots, t_k \rangle} M_1$ and $M_0 \xrightarrow{\langle t'_1, \dots, t'_i \rangle} M_2$. Assume that $\{t_1, \dots, t_k\} \neq \{t'_1, \dots, t'_i\}$. Without loss of generality, there exists a smallest $i \in [1, k]$, such that $t_i \notin \{t'_1, \dots, t'_i\}$. Hence, $\{t_1, \dots, t_{i-1}\} \subseteq \{t'_1, \dots, t'_i\}$. Let $\langle t_1, \dots, t_{i-1}, t''_i, \dots, t''_i \rangle$ be a permutation of $\langle t'_1, \dots, t'_i \rangle$. By Lemma 3.11, we have $M_0 \xrightarrow{\langle t_1, \dots, t_{i-1}, t''_i, \dots, t''_i \rangle} M_2$. Since $t_i \notin \{t''_i, \dots, t''_i\}$, we conclude, by Proposition 3.3, that t_i is enabled in M_2 , which contradicts the assumption that M_2 is final. Thus, $\{t_1, \dots, t_k\} = \{t'_1, \dots, t'_i\}$, by Lemma 3.11, $M_1 = M_2$. □

Theorem 3.12 implies that in a conflict-free cfNet, the platform constraint (1) depends only on the initial marking M_0 , given by a request R_0 . Therefore, the problem of finding a resource allocation defined by equations (2) and (3) in a conflict-free cfNet is defined uniquely.

3.2 Conflict detection

Theorem 3.12 of the previous section implies that the resource allocation problem is defined uniquely for any initial marking of a conflict-free cfNet. Theorem 3.10 provides a criterion characterising conflict-free cfNets: all the inhibitor arcs must refer to initial tokens or be non-conflicting. In order to determine whether an inhibitor arc (r, t', t) is non-conflicting, we must check whether t and t' are mutually exclusive. Mutual exclusiveness of two transitions requires that there be no reachable marking enabling them simultaneously (Definition 3.6). However, checking the existence of such a reachable marking—with some arbitrary initial one—by direct exploration is complex. Instead, we exploit the notion of marking well-formedness, which overapproximates reachability (Proposition 2.10). Given two transitions that we want to check for mutual exclusiveness, we proceed in three following steps.

1. We encode the existence of a well-formed marking enabling both transitions as a Boolean satisfiability problem and submit it to a SAT-solver.
2. If the problem is unsatisfiable, the two transitions are mutually exclusive. Otherwise, the satisfying valuation returned by the SAT-solver encodes a well-formed marking, which can be checked for reachability in an efficient manner.

3. If this marking is reachable, the two transitions are not mutually exclusive. Otherwise, we repeat step 1 with a refined encoding excluding this marking.

In the remainder of this sub-section, we fix a cfNet $(\mathcal{R}, T, F, I, \mathcal{C})$ and develop in detail the steps of the above algorithm.

3.2.1 Boolean encoding of transition enabledness

With each place-colour pair (r, t) we associate a Boolean variable y_r^t which evaluates to \mathbf{tt} if the corresponding token is present in a given marking and to \mathbf{ff} otherwise. For a transition $t \in T$, we define four predicates on markings $\mathcal{T}_t^-, \mathcal{T}_t^+, \mathcal{I}_t^*, \mathcal{I}_t^\circ \in \mathbb{B}[R \times T^*]$, i.e. $\mathcal{T}_t^-, \mathcal{T}_t^+, \mathcal{I}_t^*, \mathcal{I}_t^\circ : \mathbb{B}^{R \times T^*} \rightarrow \mathbb{B}$ (\mathcal{T} stands for *tokens*, \mathcal{I} stands for *inhibitors*):

$$\mathcal{T}_t^- \stackrel{\text{def}}{=} \bigwedge_{r \in R^-(t)} \left(\bigvee_{t \in T^-(r)} y_r^t \right), \quad // \text{ tokens are present in pre-places of } t, \quad (5)$$

$$\mathcal{T}_t^+ \stackrel{\text{def}}{=} \bigwedge_{r \in R^+(t)} y_r^t, \quad // \text{ tokens are present in post-places of } t, \quad (6)$$

$$\mathcal{I}_t^* \stackrel{\text{def}}{=} \bigwedge_{(r, *, t) \in I} \overline{y_r^*}, \quad // \text{ tokens are absent from initial inhibitors of } t, \quad (7)$$

$$\mathcal{I}_t^\circ \stackrel{\text{def}}{=} \bigwedge_{\substack{(r, t', t) \in I \\ t' \neq *}} \overline{y_r^{t'}}, \quad // \text{ tokens are absent from non-initial inhibitors of } t. \quad (8)$$

For a well-formed marking M , if the transition t has already been fired in the run leading to M , then $\mathcal{T}_t^+(M)$ evaluates to \mathbf{tt} . Thus, t is *enabled* under M iff $\mathcal{E}_t(M) = \mathbf{tt}$, for

$$\mathcal{E}_t \stackrel{\text{def}}{=} \mathcal{T}_t^- \wedge \overline{\mathcal{T}_t^+} \wedge \mathcal{I}_t^* \wedge \mathcal{I}_t^\circ. \quad (9)$$

Lemma 3.13. *For a transition t enabled with a marking M , holds the equality $\mathcal{E}_t(M) = \mathbf{tt}$. Conversely, if $\mathcal{E}_t(M) = \mathbf{tt}$ and M is well-formed then t is enabled.*

Proof. The first implication is an immediate consequence of the definition of transition enabledness (Definition 2.3). Conversely, assuming $\mathcal{E}_t(M) = \mathbf{tt}$, we have to show that $\mathcal{T}_t^+(M) = \mathbf{ff}$ implies that none of the post-places $r \in R^+(t)$ has a token (r, t) . Let $(r, t) \in M$ be such a token. By Definition 2.9, $(r, t) \in M$, for all $r \in R^+(t)$. Hence, $\mathcal{T}_t^+(M) = \mathbf{tt}$, contradicting the assumption that $\mathcal{E}_t(M) = \mathbf{tt}$. \square

Lemma 3.13 provides a characterisation of transition enabledness under well-formed markings. The following results provide a similar characterisation of the well-formedness of a marking. Combining the two, we will obtain the desired encoding.

Transition t *may have been enabled* in the run leading to a marking M only if $\mathcal{B}_t(M) = \mathbf{tt}$ (see Lemma 3.14 below), for

$$\mathcal{B}_t \stackrel{\text{def}}{=} \mathcal{T}_t^- \wedge \mathcal{I}_t^*. \quad (10)$$

Notice that the stronger predicate $\mathcal{T}_t^- \wedge \mathcal{I}_t^* \wedge \mathcal{I}_t^\circ$ does not characterise the desired property, since some of the tokens inhibiting t may have been generated after t has been fired (clearly, this cannot be the case for the initial tokens). Notice also that once $\mathcal{B}_t(M)$ holds for some marking M , it will hold for all markings reachable from M .

Lemma 3.14. *Let $M \xrightarrow{\langle t, \dots \rangle} M'$ be a run of a cfNet. Then holds the equality $\mathcal{B}_t(M') = \mathbf{tt}$.*

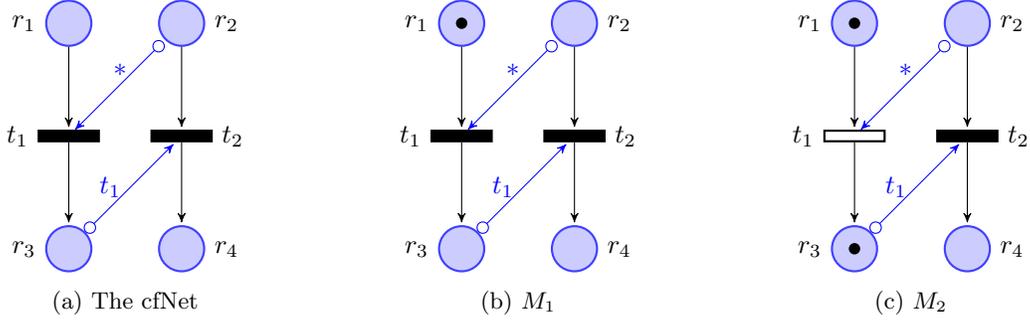


Figure 10: A cfNet with mutually exclusive transitions

Proof. By Lemma 3.13, since t is enabled in M , we have $\mathcal{E}_t(M) = \mathbf{tt}$, which implies $\mathcal{B}_t(M) = \mathbf{tt}$. Since firing the subsequent transitions in the run cannot remove the tokens from the pre-places of t , neither add initial tokens, we conclude that $\mathcal{B}_t(M') = \mathcal{B}_t(M) = \mathbf{tt}$. \square

The well-formedness of a marking essentially means that, for every non-initial token, there is a transition that could have generated it. In order to characterise this, we introduce another predicate on markings:

$$\mathcal{W} \stackrel{\text{def}}{=} \bigwedge_{r \in R} \bigwedge_{t \in T^-(r)} (y_r^t \Rightarrow \mathcal{B}_t \wedge \mathcal{T}_t^+). \quad (11)$$

Lemma 3.15. *A marking M is well-formed iff $\mathcal{W}(M) = \mathbf{tt}$.*

Sketch of the proof. \mathcal{T}_t^+ encodes the condition 2 of Definition 2.9, whereas \mathcal{B}_t encodes the conditions 1 and 3. \square

Example 3.16. Consider the example in Figure 10 (reproduced for convenience from Figure 9). For the transitions of this example, we have

$$\begin{aligned} \mathcal{T}_{t_1}^- &= y_{r_1}^*, & \mathcal{T}_{t_1}^+ &= y_{r_3}^{t_1}, & \mathcal{I}_{t_1}^* &= \overline{y_{r_2}^*}, & \mathcal{I}_{t_1}^\circ &= \mathbf{tt}, \\ \mathcal{T}_{t_2}^- &= y_{r_2}^*, & \mathcal{T}_{t_2}^+ &= y_{r_4}^{t_2}, & \mathcal{I}_{t_2}^* &= \mathbf{tt}, & \mathcal{I}_{t_2}^\circ &= \overline{y_{r_3}^{t_1}}. \end{aligned}$$

Thus, under the marking M_1 (Figure 10b), transition t_1 is enabled, since

$$\mathcal{E}_{t_1}(M_1) = (\mathcal{T}_{t_1}^- \wedge \overline{\mathcal{T}_{t_1}^+} \wedge \mathcal{I}_{t_1}^* \wedge \mathcal{I}_{t_1}^\circ)(M_1) = (y_{r_1}^* \wedge \overline{y_{r_3}^{t_1}} \wedge \overline{y_{r_2}^*})(M_1) = \mathbf{tt}.$$

Once t_1 has been fired to obtain the marking M_2 (Figure 10c), it is no longer enabled. Indeed, we have $\mathcal{E}_{t_1}(M_2) = \mathbf{ff}$, since $y_{r_3}^{t_1}(M_2) = \mathbf{tt}$, but $\mathcal{B}_{t_1}(M_2) = (y_{r_1}^* \wedge \overline{y_{r_2}^*})(M_2) = \mathbf{tt}$.

Observe now that the following Boolean equivalence holds:

$$\mathcal{E}_{t_1} \wedge \mathcal{E}_{t_2} = (y_{r_1}^* \wedge \overline{y_{r_3}^{t_1}} \wedge \overline{y_{r_2}^*}) \wedge (y_{r_2}^* \wedge \overline{y_{r_4}^{t_2}} \wedge \overline{y_{r_3}^{t_1}}) = \mathbf{ff},$$

since $y_{r_2}^*$ appears in both positive and negative form. Indeed, as discussed in the previous section, transitions t_1 and t_2 are mutually exclusive. \diamond

In the above example, we do not need to consider the well-formedness of markings, since $\mathcal{E}_{t_1} \wedge \mathcal{E}_{t_2}$ is not satisfiable. In general, we additionally conjunct the predicate \mathcal{W} to ensure that only well-formed markings are considered. Thus, if the predicate

$$\mathcal{E}_{t_1} \wedge \mathcal{E}_{t_2} \wedge \mathcal{W} \quad (12)$$

is not satisfiable then transitions t_1 and t_2 are mutually exclusive.

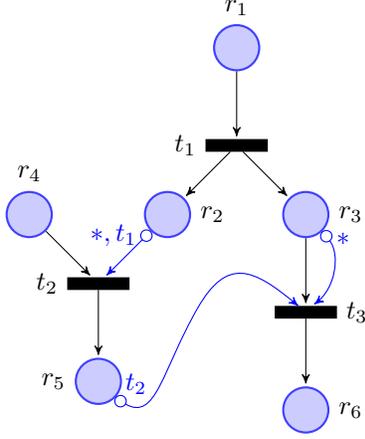


Figure 11: A cfNet with mutually exclusive t_2 and t_3

Example 3.17 (Mutually exclusive transitions). Consider a cfNet shown in Figure 11. There is an inhibitor arc (r_5, t_2, t_3) which might introduce a conflict, so we verify whether transitions t_2 and t_3 are mutually exclusive. Intuitively, for t_3 to be enabled, the token (r_3, t_1) must be present in the marking. Hence, transition t_1 must have fired, whereby generating also the token (r_2, t_1) . However, the presence of this latter token would inhibit t_2 . We now show this through the Boolean encoding. For the cfNet in Figure 11, we have

$$\begin{aligned}
\mathcal{E}_{t_1} &= y_{r_1}^* \wedge \overline{y_{r_2}^{t_1}} \wedge \overline{y_{r_3}^{t_1}}, & \mathcal{B}_{t_1} \wedge \mathcal{T}_{t_1}^+ &= y_{r_1}^* \wedge y_{r_2}^{t_1} \wedge y_{r_3}^{t_1}, \\
\mathcal{E}_{t_2} &= y_{r_4}^* \wedge \overline{y_{r_2}^{t_1}} \wedge \overline{y_{r_2}^*} \wedge \overline{y_{r_5}^{t_2}}, & \mathcal{B}_{t_2} \wedge \mathcal{T}_{t_2}^+ &= y_{r_4}^* \wedge \overline{y_{r_2}^*} \wedge y_{r_5}^{t_2}, \\
\mathcal{E}_{t_3} &= (y_{r_3}^* \vee y_{r_3}^{t_1}) \wedge \overline{y_{r_5}^{t_2}} \wedge \overline{y_{r_3}^*} \wedge \overline{y_{r_6}^{t_3}}, & \mathcal{B}_{t_3} \wedge \mathcal{T}_{t_3}^+ &= (y_{r_3}^* \vee y_{r_3}^{t_1}) \wedge \overline{y_{r_3}^*} \wedge y_{r_6}^{t_3}, \\
&= y_{r_3}^{t_1} \wedge \overline{y_{r_5}^{t_2}} \wedge \overline{y_{r_3}^*} \wedge \overline{y_{r_6}^{t_3}}, & &= y_{r_3}^{t_1} \wedge \overline{y_{r_3}^*} \wedge y_{r_6}^{t_3}.
\end{aligned}$$

In particular,

$$\begin{aligned}
\mathcal{E}_{t_2} \wedge \mathcal{E}_{t_3} \wedge \mathcal{W} &= (y_{r_4}^* \wedge \overline{y_{r_2}^{t_1}} \wedge \overline{y_{r_2}^*} \wedge \overline{y_{r_5}^{t_2}}) \wedge (y_{r_3}^{t_1} \wedge \overline{y_{r_5}^{t_2}} \wedge \overline{y_{r_3}^*} \wedge \overline{y_{r_6}^{t_3}}) \wedge \mathcal{W} \\
&= y_{r_4}^* \wedge \overline{y_{r_2}^{t_1}} \wedge \overline{y_{r_2}^*} \wedge \overline{y_{r_5}^{t_2}} \wedge y_{r_3}^{t_1} \wedge \overline{y_{r_3}^*} \wedge \overline{y_{r_6}^{t_3}} \wedge \mathcal{W} \\
&\quad // \text{applying modus ponens to (11) and } y_{r_3}^{t_1}, \text{ we conjunct } \mathcal{B}_{t_1} \wedge \mathcal{T}_{t_1}^+ \\
&= y_{r_4}^* \wedge \overline{y_{r_2}^{t_1}} \wedge \overline{y_{r_2}^*} \wedge \overline{y_{r_5}^{t_2}} \wedge y_{r_3}^{t_1} \wedge \overline{y_{r_3}^*} \wedge \overline{y_{r_6}^{t_3}} \wedge (y_{r_1}^* \wedge y_{r_2}^{t_1} \wedge y_{r_3}^{t_1}) \wedge \mathcal{W} \\
&= \mathbf{ff}.
\end{aligned}$$

Therefore, we conclude that t_2 and t_3 are mutually exclusive. \diamond

Recall from Section 2.1 that well-formedness is an over-approximation of reachability. Therefore, the converse does not hold: given a marking M satisfying $\mathcal{E}_{t_1} \wedge \mathcal{E}_{t_2} \wedge \mathcal{W}$, one has to check whether M is reachable. However, this can be done efficiently, as explained in the following section.

3.2.2 Marking reachability

Let $M \subseteq \mathcal{R} \times T^*$ be a well-formed marking of a cfNet $(\mathcal{R}, T, F, I, C)$. Recall (Definition 2.2) that T_M denotes the set of all colours of the tokens in M .

We associate to the marking M the corresponding directed *causality hyper-graph* $\mathcal{G}_M \stackrel{\text{def}}{=} (V, E)$ with vertices $V = T_M$ and the set $E \subseteq T_M \times 2^{T_M}$ of edges representing essentially the “must be fired before” relation among the corresponding transitions: an edge (t, S) (from t to S) with $S \in 2^{T_M}$ means that, *for the transition t to be fired, at least one transition in S must be fired before t .*

For an edge (t, S) and a vertex $u \in S$, we call a pair (t, u) a *sub-edge*.

We put $E = E_1 \cup E_2$ with E_1 and E_2 defined by (13) and (14) below.

For a place $r \in \mathcal{R}$, denote $T_r \stackrel{\text{def}}{=} \{t \in T_M \mid (r, t) \in M\}$ the set of colours of the tokens in r present in the marking M . We put

$$E_1 \stackrel{\text{def}}{=} \left\{ (t, T_r) \in T_M \times 2^{T_M} \mid t \in T_M, r \in R^-(t) \right\}. \quad (13)$$

By Definition 2.3, for any transition t to be fired, it is necessary that in each place $r \in R^-(t)$ there be at least one token. Hence, at least one of the transitions generating such tokens must be fired before t .

Furthermore, if firing a transition t' generates a token that inhibits some transition t , then firing of t cannot happen after that of t' . Thus, we put

$$E_2 \stackrel{\text{def}}{=} \left\{ (t', \{t\}) \in T_M \times 2^{T_M} \mid \exists r \in \mathcal{R} : (r, t', t) \in I \right\}. \quad (14)$$

Notice that, if such $(r, t', t) \in I$ actually exists, necessarily $(r, t') \in M$, since $t' \in T_M$. Thus, we do not have to state this condition explicitly in (14).

Definition 3.18. Let $G = (V, E)$ with $E \subseteq V \times 2^V$ be a hyper-graph. A *path* in G is a sequence $(e_i)_{i=0}^n$, with $e_i = (v_i, S_i) \in E$, such that $v_{i+1} \in S_i$, for all $i < n$. When $n \in \mathbb{N}$, we say that the path is *finite*, otherwise, when $n = \infty$, it is *infinite*. We say that the path *starts with the edge* e_0 .

Example 3.19. Figure 12a shows a cfNet and a marking M . This marking is reachable by a run $\langle t_1, t_2, t_4, t_3 \rangle$. Figure 12b shows the corresponding causality hyper-graph. Each of the transitions t_1, t_2, t_3 requires an initial token to fire. Transition t_4 requires either a token from t_1 , or a token from t_2 , hence the hyper-arc $(t_4, \{t_1, t_2\})$. The inhibitor arc (r_5, t_3, t_4) induces the arc $(t_3, \{t_4\})$ in the causality graph. A sequence $((t_3, \{t_4\}), (t_4, \{t_1, t_2\}), (t_1, \{*\}))$ is an example of a path. \diamond

Definition 3.20. A hyper-graph $G = (V, E)$ with $E \subseteq V \times 2^V$ has a *cycle* $C \subseteq V$, if there exists a set of finite paths $\{(e_i^j)_{i=0}^{n_j}\}_{j \in J}$, with $e_i^j = (v_i^j, S_i^j) \in E$, such that $C = \{v_i^j \mid j \in J, i \in [0, n_j]\}$ and, for all $j \in J$ and $i \in [0, n_j]$, we have $S_i^j \subseteq C$. Otherwise, G is said to be *free from cycles*.

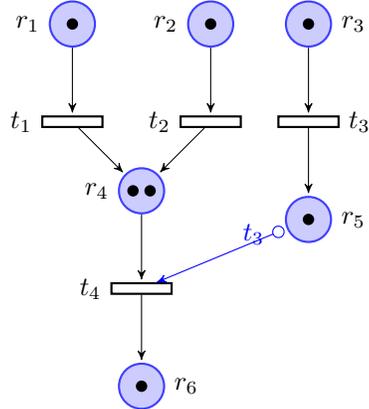
Example 3.21. Figure 13a shows a cfNet and an unreachable marking M . Figure 13b shows the corresponding causality hyper-graph with a cycle $C = (t_1, t_2, t_3, t_4, t_5)$. The set of paths inducing the cycle is $\left\{ ((t_3, \{t_1, t_2\}), (t_1, \{t_4\}), (t_4, \{t_3\})), ((t_3, \{t_1, t_2\}), (t_2, \{t_5\}), (t_5, \{t_3\})) \right\}$. \diamond

Notice that in the causality hyper-graph \mathcal{G}_M corresponding to a well-formed marking M , the only vertex that does not have any outgoing edges is $*$.

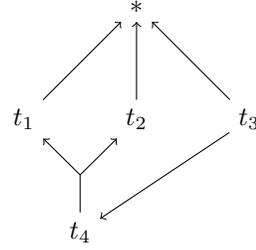
Lemma 3.22. *Let M be a well-formed marking. If $* \notin T_M$ then the causality graph \mathcal{G}_M has a cycle.*

Proof. Let $\mathcal{G}_M = (V, E)$. The set V is trivially a cycle. \square

Given a dependency graph which represents a reachable marking M , we can obtain a run leading to this marking by traversing the graph backwards from the vertex $*$ corresponding to the initial tokens. Given \mathcal{G}_M on input, Algorithm 1 constructs a list *Order* which contains the transitions in T_M in the order of firing that leads to M .

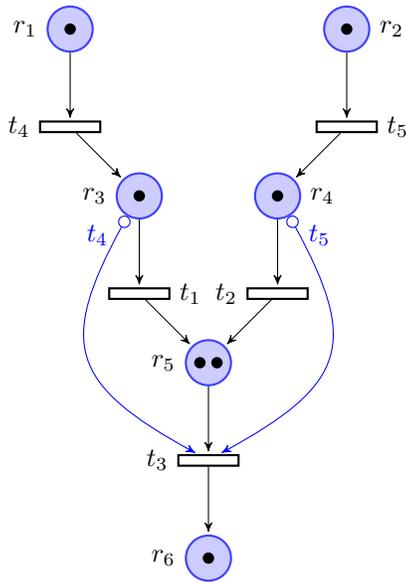


(a) A reachable marking M

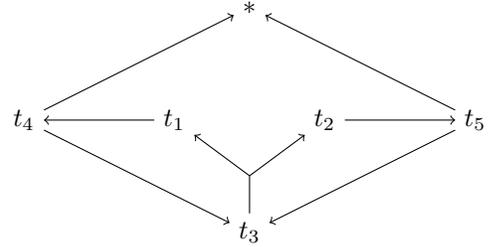


(b) The causality hyper-graph \mathcal{G}_M

Figure 12: An example of a cfNet and its causality hyper-graph



(a) An unreachable marking M



(b) The causality hyper-graph \mathcal{G}_M

Figure 13: An example of a cfNet and its causality hyper-graph with a cycle

Algorithm 1 Topological sort for hyper-graphs

Input: Hyper-graph $\mathcal{G} = (V, E)$

Output: If \mathcal{G} is free from cycles, a total order on vertices, respecting the graph edges; otherwise the empty list

```
1: Order = EmptyList
2: for  $v \in V$  do
3:    $E_o(v) = \{(v, S) \in E \mid S \subseteq V\}$        $\triangleright$  construct the set of outgoing edges for each vertex
4:    $E_i(v) = \{(u, S) \in E \mid v \in S\}$        $\triangleright$  construct the set of incoming edges for each vertex
5: end for
6:  $K = \{v \in V \mid E_o(v) = \emptyset\}$            $\triangleright$  working set  $K$  consists of vertices without outgoing edges
7: while  $K \neq \emptyset$  do
8:    $K = K \setminus \{v\}$  for some  $v \in K$            $\triangleright$  choose a random vertex  $v$  from  $K$ 
9:   Order.append( $v$ )
10:  for  $e = (u, S) \in E_i(v)$                      $\triangleright$  for each edge incoming into  $v$  do
11:     $E = E \setminus \{e\}$                            $\triangleright$  remove it from the set of all edges
12:     $E_o(u) = E_o(u) \setminus \{e\}$                    $\triangleright$  remove it from outgoing edges of  $u$ 
13:    for  $w \in S$  do
14:       $E_i(w) = E_i(w) \setminus \{e\}$                  $\triangleright$  remove it from incoming edges of each  $w$ 
15:    end for
16:    if  $E_o(u) = \emptyset$                            $\triangleright$  if  $u$  has no more outgoing edges then
17:       $K = K \cup \{u\}$                                $\triangleright$  add  $u$  it to the working set  $K$ 
18:    end if
19:  end for
20: end while
21: if  $E \neq \emptyset$  then
22:   return EmptyList                                $\triangleright$   $\mathcal{G}$  contains cycles
23: else
24:   return Order
25: end if
```

Lemma 3.23. *Given a hyper-graph (V, E) of well-formed marking M , Algorithm 1 either constructs a run leading to M or finds a cycle.*

Proof. Since the marking M is well-formed, we have $* \in V$, hence also $* \in K$. Furthermore, for all $t \in T$ and $(r, *) \in I(t)$, holds $(r, *) \notin M$. Finally, $(*, S) \notin E$, for any S . Thus, $E_o(*) = \emptyset$.

Assume that Algorithm 1 terminates with $K = \emptyset$ and $E = \emptyset$. $Order = \langle *, t_1, \dots, t_n \rangle$ is a sequence of colours. Let us prove that it is a run. Each vertex t_i is only added when $E_o(t_i) = \emptyset$. This can happen only when the edges are removed. By the algorithm, the edges can be removed only if they lead to transition which has been in the run before the current one. Therefore, all the transitions on which t_i depends have already fired, t_i can fire as well, and $Order$ is indeed a run.

Assume now that Algorithm 1 terminates with $K = \emptyset$ and $E \neq \emptyset$. Let $C = V \setminus Order$ be the set of remaining vertices. Since a vertice is only placed in K (hence also in $Order$) when all its outgoing edges have been removed, $E \neq \emptyset$ implies $C \neq \emptyset$. Consider some $v \in C$. Since $v \notin Order$, there exists an edge $(v, S_v) \in E$. Furthermore $S_v \cap Order = \emptyset$, which implies that $S_v \subseteq C$. Hence C is a cycle, as witnessed by the finite set of edges $\{(v, S_v) \mid v \in C\}$ (trivially, a finite set of edges is also a finite set of paths). \square

Theorem 3.24. *Let \mathcal{G}_M be a causality graph for a well-formed marking M . M is reachable iff \mathcal{G}_m is free from cycles.*

Proof. If the causality graph is free from cycles, then Algorithm 1 constructs a run, therefore, M is reachable.

Assume that M is reachable from an initial marking M_0 and \mathcal{G}_M has a cycle $C \subseteq T_M$. First notice that, by Definition 3.20, $* \notin C$.

We have $M_0 \xrightarrow{\langle t_1, \dots, t_n \rangle} M$ with $\{*, t_1, \dots, t_n\} = T_M$. Let $k \in [1, n]$ be the smallest index, such that $t_k \in C$. By Definition 3.20, there exists an edge (t_k, S) in \mathcal{G}_M , such that $S \subseteq C$. Two cases are possible: 1) $S = T_r$, for some $r \in R^-(t_k)$ (cf. (13)) or 2) $S = \{t\}$, for some $t \in T_M$, such that $(r, t_k, t) \in I$ (cf. (14)). By the construction of \mathcal{G}_M , there is at least one $t \in S$, such that t must have been fired before t_k . Since $S \subseteq C$, we have $t \in C$, which contradicts the assumption that k is the minimal index, such that $t_k \in C$, thereby proving that reachability of M implies cycle-freedom of \mathcal{G}_M . \square

Proposition 3.25. *The time complexity of Algorithm 1 is linear in the size of the input hyper-graph.*

Proof. Each node $t \in V$ is put in K and hence visited at most once (lines 6, 17):

- each t is removed from K (lines 7, 8);
- t is added to K only when it has become a vertex without outgoing edges after the removal of an edge (lines 10–17), therefore:
 - once t is removed from K , it can never be put back;
 - once t is added to K , it cannot be added for the second time.

The construction of the edge sets E_o and E_i runs in linear time:

- the set $E_o(v)$ is constructed in $O(|V|)$ (Line 3);
- during the construction of $E_i(v)$ (Line 4), each edge and each its sub-edge is visited only once, therefore, the time for it is $O(H)$, where H is the number of all sub-edges: $H = \sum_{(v,S) \in E} |S|$.

The total time for the update of the edge sets E_o and E_i is also linear:

- each edge is removed from $E_o(u)$ only once in Line 12 and is never added again;
- there are at most H removals from all the $E_i(w)$ 14, as each edge is removed only once, no new edges are added and $\sum_{w \in V} |E_i(w)| = H$.

Finally, each edge $e \in E$ is visited at most once: once we consider it, we remove it and cannot visit it again (lines 10–14).

Therefore, the algorithm runs in $O(|V| + H)$ time, where H is the number of the sub-edges of the graph. □

3.2.3 Encoding refinement

Let $M \subseteq \mathcal{R} \times T^*$ be a well-formed marking of a cfNet $\mathcal{N} = (\mathcal{R}, T, F, I, \mathcal{C})$, enabling two conflicting transitions t_1 and t_2 . If M is reachable, \mathcal{N} has a conflict. However, if M is not reachable, the encoding has to be refined to exclude M . Let Φ be the predicate used at the previous step of the process (initially $\Phi = \mathcal{E}_{t_1} \wedge \mathcal{E}_{t_2} \wedge \mathcal{W}$). We refine this predicate by taking the conjunction $\Phi \wedge \overline{\Phi_M}$, where

$$\Phi_M = \bigwedge_{(r,t) \in M} y_r^t \wedge \bigwedge_{(r,t) \notin M} \overline{y_r^t}$$

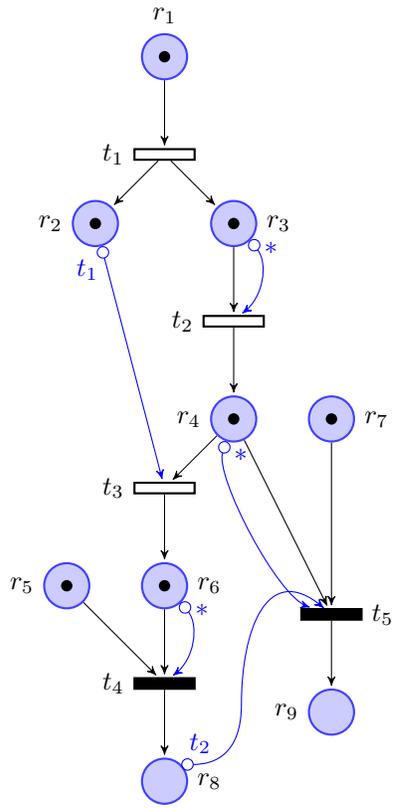
is the characteristic predicate of the marking M .

Example 3.26. Consider the cfNet in Figure 14. Let us check transitions t_4 and t_5 for mutual exclusiveness.

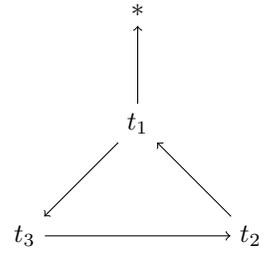
$$\begin{aligned} \mathcal{E}_{t_4} &= y_{r_5}^* \wedge (y_{r_6}^* \vee y_{r_6}^{t_3}) \wedge \overline{y_{r_8}^{t_4}} \wedge \overline{y_{r_6}^*} = y_{r_5}^* \wedge y_{r_6}^{t_3} \wedge \overline{y_{r_8}^{t_4}} \wedge \overline{y_{r_6}^*} \\ \mathcal{E}_{t_5} &= y_{r_7}^* \wedge (y_{r_4}^* \vee y_{r_4}^{t_2}) \wedge \overline{y_{r_9}^{t_5}} \wedge \overline{y_{r_4}^*} = y_{r_7}^* \wedge y_{r_4}^{t_2} \wedge \overline{y_{r_9}^{t_5}} \wedge \overline{y_{r_4}^*} \wedge \overline{y_{r_8}^{t_4}} \end{aligned}$$

$$\begin{aligned} \mathcal{B}_{t_1} \wedge \mathcal{T}_{t_1}^+ &= y_{r_1}^* \wedge y_{r_2}^{t_1} \wedge y_{r_3}^{t_1}, & \mathcal{B}_{t_2} \wedge \mathcal{T}_{t_2}^+ &= (y_{r_3}^* \vee y_{r_3}^{t_1}) \wedge \overline{y_{r_3}^*} \wedge y_{r_4}^{t_2}, \\ \mathcal{B}_{t_3} \wedge \mathcal{T}_{t_3}^+ &= (y_{r_4}^* \vee y_{r_4}^{t_2}) \wedge \overline{y_{r_2}^*} \wedge y_{r_6}^{t_3}, & \mathcal{B}_{t_4} \wedge \mathcal{T}_{t_4}^+ &= (y_{r_6}^{t_3} \vee y_{r_6}^*) \wedge \overline{y_{r_6}^*} \wedge y_{r_5}^* \wedge y_{r_8}^{t_4}, \\ \mathcal{B}_{t_5} \wedge \mathcal{T}_{t_5}^+ &= y_{r_7}^* \wedge (y_{r_4}^* \vee y_{r_4}^{t_2}) \wedge \overline{y_{r_4}^*} \wedge y_{r_9}^{t_5}. \end{aligned}$$

$$\begin{aligned} \mathcal{E}_{t_4} \wedge \mathcal{E}_{t_5} \wedge \mathcal{W} &= (y_{r_5}^* \wedge y_{r_6}^{t_3} \wedge \overline{y_{r_8}^{t_4}} \wedge \overline{y_{r_6}^*}) \wedge (y_{r_7}^* \wedge y_{r_4}^{t_2} \wedge \overline{y_{r_9}^{t_5}} \wedge \overline{y_{r_4}^*} \wedge \overline{y_{r_8}^{t_4}}) \wedge \mathcal{W} \\ &= y_{r_5}^* \wedge y_{r_7}^* \wedge \overline{y_{r_6}^*} \wedge \overline{y_{r_4}^*} \wedge y_{r_6}^{t_3} \wedge y_{r_4}^{t_2} \wedge \overline{y_{r_9}^{t_5}} \wedge \overline{y_{r_8}^{t_4}} \wedge \mathcal{W} \\ &\quad // \text{applying modus ponens to (11) and } y_{r_6}^{t_3}, \text{ we conjunct } \mathcal{B}_{t_3} \wedge \mathcal{T}_{t_3}^+ \\ &= y_{r_5}^* \wedge y_{r_7}^* \wedge \overline{y_{r_6}^*} \wedge \overline{y_{r_4}^*} \wedge y_{r_4}^{t_2} \wedge \overline{y_{r_9}^{t_5}} \wedge \overline{y_{r_8}^{t_4}} \wedge y_{r_6}^{t_3} \wedge ((y_{r_4}^* \vee y_{r_4}^{t_2}) \wedge \overline{y_{r_2}^*} \wedge y_{r_6}^{t_3}) \wedge \mathcal{W} \\ &= y_{r_5}^* \wedge y_{r_7}^* \wedge \overline{y_{r_6}^*} \wedge \overline{y_{r_4}^*} \wedge \overline{y_{r_2}^*} \wedge y_{r_4}^{t_2} \wedge y_{r_6}^{t_3} \wedge \overline{y_{r_9}^{t_5}} \wedge \overline{y_{r_8}^{t_4}} \wedge \mathcal{W} \\ &\quad // \text{applying modus ponens to (11) and } y_{r_4}^{t_2}, \text{ we conjunct } \mathcal{B}_{t_2} \wedge \mathcal{T}_{t_2}^+ \\ &= y_{r_5}^* \wedge y_{r_7}^* \wedge \overline{y_{r_6}^*} \wedge \overline{y_{r_4}^*} \wedge \overline{y_{r_2}^*} \wedge y_{r_4}^{t_2} \wedge y_{r_6}^{t_3} \wedge \overline{y_{r_9}^{t_5}} \wedge \overline{y_{r_8}^{t_4}} \wedge (y_{r_3}^{t_1} \wedge \overline{y_{r_3}^*} \wedge y_{r_4}^{t_2}) \wedge \mathcal{W} \\ &= y_{r_5}^* \wedge y_{r_7}^* \wedge \overline{y_{r_6}^*} \wedge \overline{y_{r_4}^*} \wedge \overline{y_{r_2}^*} \wedge \overline{y_{r_3}^*} \wedge y_{r_4}^{t_2} \wedge y_{r_6}^{t_3} \wedge y_{r_3}^{t_1} \wedge \overline{y_{r_9}^{t_5}} \wedge \overline{y_{r_8}^{t_4}} \wedge \mathcal{W} \\ &\quad // \text{applying modus ponens to (11) and } y_{r_3}^{t_1}, \text{ we conjunct } \mathcal{B}_{t_1} \wedge \mathcal{T}_{t_1}^+ \\ &= y_{r_5}^* \wedge y_{r_7}^* \wedge \overline{y_{r_6}^*} \wedge \overline{y_{r_4}^*} \wedge \overline{y_{r_2}^*} \wedge \overline{y_{r_3}^*} \wedge y_{r_4}^{t_2} \wedge y_{r_6}^{t_3} \wedge y_{r_3}^{t_1} \wedge \overline{y_{r_9}^{t_5}} \wedge \overline{y_{r_8}^{t_4}} \wedge y_{r_1}^* \wedge y_{r_2}^{t_1} \wedge y_{r_3}^{t_1} \wedge \mathcal{W} \\ &= y_{r_5}^* \wedge y_{r_7}^* \wedge y_{r_1}^* \wedge \overline{y_{r_6}^*} \wedge \overline{y_{r_4}^*} \wedge \overline{y_{r_2}^*} \wedge \overline{y_{r_3}^*} \wedge y_{r_4}^{t_2} \wedge y_{r_6}^{t_3} \wedge y_{r_3}^{t_1} \wedge y_{r_2}^{t_1} \wedge \overline{y_{r_9}^{t_5}} \wedge \overline{y_{r_8}^{t_4}} \wedge \mathcal{W}. \end{aligned}$$



(a) A cfNat with a marking M



(b) The causality graph

Figure 14: A cfNat with mutually exclusive t_4 and t_5 which can be enabled by an unreachable marking

The well-formed marking $M_1 = \{(r_1, *), (r_5, *), (r_7, *), (r_2, t_1), (r_3, t_1), (r_4, t_2), (r_6, t_3)\}$ (see Figure 14) satisfies the formula above. However, this marking is not reachable: once t_1 is fired, it disables t_3 , therefore there cannot be a token (r_6, t_3) . Transition t_3 cannot fire before t_1 either, since it requires a token in r_4 which is put there by transition t_2 , which is enabled only after the firing of t_1 .

The causality graph in Figure 14b indeed shows that there is a cycle (t_3, t_2, t_1) . Therefore, we refine the formula above by conjuncting it with the negation of the characteristic predicate of M_1 :

$$\begin{aligned} \Phi_{M_1} &= y_{r_1}^* \wedge y_{r_5}^* \wedge y_{r_7}^* \wedge y_{r_2}^{t_1} \wedge y_{r_3}^{t_1} \wedge y_{r_4}^{t_2} \wedge y_{r_6}^{t_3} \wedge \overline{y_{r_2}^*} \wedge \overline{y_{r_3}^*} \wedge \overline{y_{r_4}^*} \wedge \overline{y_{r_6}^*} \wedge \overline{y_{r_8}^*} \wedge \overline{y_{r_9}^*} \wedge \overline{y_{r_8}^{t_4}} \wedge \overline{y_{r_9}^{t_5}} \\ \Phi &= \mathcal{E}_{t_4} \wedge \mathcal{E}_{t_5} \wedge \mathcal{W} \wedge \overline{\Phi_{M_1}} \\ &= \overline{y_{r_1}^* \wedge y_{r_5}^* \wedge y_{r_7}^* \wedge y_{r_2}^{t_1} \wedge y_{r_3}^{t_1} \wedge y_{r_4}^{t_2} \wedge y_{r_6}^{t_3} \wedge \overline{y_{r_2}^*} \wedge \overline{y_{r_3}^*} \wedge \overline{y_{r_4}^*} \wedge \overline{y_{r_6}^*} \wedge \overline{y_{r_8}^{t_4}} \wedge \overline{y_{r_9}^{t_5}} \wedge \mathcal{W}} \\ &\quad \wedge \overline{y_{r_1}^* \wedge y_{r_5}^* \wedge y_{r_7}^* \wedge y_{r_2}^{t_1} \wedge y_{r_3}^{t_1} \wedge y_{r_4}^{t_2} \wedge y_{r_6}^{t_3} \wedge \overline{y_{r_2}^*} \wedge \overline{y_{r_3}^*} \wedge \overline{y_{r_4}^*} \wedge \overline{y_{r_6}^*} \wedge \overline{y_{r_8}^{t_4}} \wedge \overline{y_{r_9}^{t_5}} \wedge \mathcal{W}} \\ &= \overline{y_{r_1}^* \wedge y_{r_5}^* \wedge y_{r_7}^* \wedge y_{r_2}^{t_1} \wedge y_{r_3}^{t_1} \wedge y_{r_4}^{t_2} \wedge y_{r_6}^{t_3} \wedge \overline{y_{r_2}^*} \wedge \overline{y_{r_3}^*} \wedge \overline{y_{r_4}^*} \wedge \overline{y_{r_6}^*} \wedge \overline{y_{r_8}^{t_4}} \wedge \overline{y_{r_9}^{t_5}} \wedge \mathcal{W} \wedge (y_{r_8}^* \vee y_{r_9}^*)}. \end{aligned}$$

The marking $M_2 = M_1 \cup \{(r_8, *)\} = \{(r_1, *), (r_5, *), (r_7, *), (r_2, t_1), (r_3, t_1), (r_4, t_2), (r_6, t_3), (r_8, *)\}$ is well-formed and satisfies the refined encoding. We have $T_{M_1} = T_{M_2}$; since there is not t , such that $(r_8, *, t) \in I$ or $r_8 \in R^-(t)$, we have $\mathcal{G}_{M_1} = \mathcal{G}_{M_2}$. Hence, the causality graph \mathcal{G}_{M_2} has the same cycle as \mathcal{G}_{M_1} . Therefore M_2 is not reachable and we refine the predicate Φ again:

$$\begin{aligned} \Phi &= \mathcal{E}_{t_4} \wedge \mathcal{E}_{t_5} \wedge \mathcal{W} \wedge \overline{\Phi_{M_1}} \wedge \overline{\Phi_{M_2}} \\ &= \overline{y_{r_1}^* \wedge y_{r_5}^* \wedge y_{r_7}^* \wedge y_{r_2}^{t_1} \wedge y_{r_3}^{t_1} \wedge y_{r_4}^{t_2} \wedge y_{r_6}^{t_3} \wedge \overline{y_{r_2}^*} \wedge \overline{y_{r_3}^*} \wedge \overline{y_{r_4}^*} \wedge \overline{y_{r_6}^*} \wedge \overline{y_{r_8}^{t_4}} \wedge \overline{y_{r_9}^{t_5}} \wedge \mathcal{W} \wedge (y_{r_8}^* \vee y_{r_9}^*)} \\ &\quad \wedge \overline{y_{r_1}^* \wedge y_{r_5}^* \wedge y_{r_7}^* \wedge y_{r_2}^{t_1} \wedge y_{r_3}^{t_1} \wedge y_{r_4}^{t_2} \wedge y_{r_6}^{t_3} \wedge y_{r_8}^* \wedge \overline{y_{r_2}^*} \wedge \overline{y_{r_3}^*} \wedge \overline{y_{r_4}^*} \wedge \overline{y_{r_6}^*} \wedge \overline{y_{r_8}^{t_4}} \wedge \overline{y_{r_9}^{t_5}} \wedge \overline{y_{r_9}^*}} \\ &= \overline{y_{r_1}^* \wedge y_{r_5}^* \wedge y_{r_7}^* \wedge y_{r_2}^{t_1} \wedge y_{r_3}^{t_1} \wedge y_{r_4}^{t_2} \wedge y_{r_6}^{t_3} \wedge \overline{y_{r_2}^*} \wedge \overline{y_{r_3}^*} \wedge \overline{y_{r_4}^*} \wedge \overline{y_{r_6}^*} \wedge \overline{y_{r_8}^{t_4}} \wedge \overline{y_{r_9}^{t_5}} \wedge \mathcal{W} \wedge y_{r_9}^*}. \end{aligned}$$

The marking $M_3 = M_1 \cup \{(r_9, *)\} = \{(r_1, *), (r_5, *), (r_7, *), (r_2, t_1), (r_3, t_1), (r_4, t_2), (r_6, t_3), (r_9, *)\}$ is well-formed and satisfies Φ . However, it is unreachable for the same reason as \mathcal{G}_{M_2} above. We refine Φ again:

$$\begin{aligned} \Phi &= \mathcal{E}_{t_4} \wedge \mathcal{E}_{t_5} \wedge \mathcal{W} \wedge \overline{\Phi_{M_1}} \wedge \overline{\Phi_{M_2}} \wedge \overline{\Phi_{M_3}} \\ &= \overline{y_{r_1}^* \wedge y_{r_5}^* \wedge y_{r_7}^* \wedge y_{r_2}^{t_1} \wedge y_{r_3}^{t_1} \wedge y_{r_4}^{t_2} \wedge y_{r_6}^{t_3} \wedge \overline{y_{r_2}^*} \wedge \overline{y_{r_3}^*} \wedge \overline{y_{r_4}^*} \wedge \overline{y_{r_6}^*} \wedge \overline{y_{r_8}^{t_4}} \wedge \overline{y_{r_9}^{t_5}} \wedge \mathcal{W} \wedge y_{r_9}^*} \\ &\quad \wedge \overline{y_{r_1}^* \wedge y_{r_5}^* \wedge y_{r_7}^* \wedge y_{r_2}^{t_1} \wedge y_{r_3}^{t_1} \wedge y_{r_4}^{t_2} \wedge y_{r_6}^{t_3} \wedge y_{r_9}^* \wedge \overline{y_{r_8}^*} \wedge \overline{y_{r_2}^*} \wedge \overline{y_{r_3}^*} \wedge \overline{y_{r_4}^*} \wedge \overline{y_{r_6}^*} \wedge \overline{y_{r_8}^{t_4}} \wedge \overline{y_{r_9}^{t_5}} \wedge \overline{y_{r_9}^*}} \\ &= \overline{y_{r_1}^* \wedge y_{r_5}^* \wedge y_{r_7}^* \wedge y_{r_2}^{t_1} \wedge y_{r_3}^{t_1} \wedge y_{r_4}^{t_2} \wedge y_{r_6}^{t_3} \wedge \overline{y_{r_2}^*} \wedge \overline{y_{r_3}^*} \wedge \overline{y_{r_4}^*} \wedge \overline{y_{r_6}^*} \wedge \overline{y_{r_8}^{t_4}} \wedge \overline{y_{r_9}^{t_5}} \wedge \mathcal{W} \wedge y_{r_8}^* \wedge y_{r_9}^*}. \end{aligned}$$

Finally, the marking $M_4 = M_1 \cup \{(r_8, *), (r_9, *)\} = \{(r_1, *), (r_5, *), (r_7, *), (r_2, t_1), (r_3, t_1), (r_4, t_2), (r_6, t_3), (r_8, *), (r_9, *)\}$ is well-formed and satisfies Φ . As above, the causality graph \mathcal{G}_{M_4} coincides with \mathcal{G}_{M_1} and M_4 is again unreachable. Conjuncting Φ with $\overline{\Phi_{M_4}}$, we obtain

$$\begin{aligned} \Phi &= \mathcal{E}_{t_4} \wedge \mathcal{E}_{t_5} \wedge \mathcal{W} \wedge \overline{\Phi_{M_1}} \wedge \overline{\Phi_{M_2}} \wedge \overline{\Phi_{M_3}} \wedge \overline{\Phi_{M_4}} \\ &= \overline{y_{r_1}^* \wedge y_{r_5}^* \wedge y_{r_7}^* \wedge y_{r_2}^{t_1} \wedge y_{r_3}^{t_1} \wedge y_{r_4}^{t_2} \wedge y_{r_6}^{t_3} \wedge \overline{y_{r_2}^*} \wedge \overline{y_{r_3}^*} \wedge \overline{y_{r_4}^*} \wedge \overline{y_{r_6}^*} \wedge \overline{y_{r_8}^{t_4}} \wedge \overline{y_{r_9}^{t_5}} \wedge \mathcal{W} \wedge y_{r_8}^* \wedge y_{r_9}^*} \\ &\quad \wedge \overline{y_{r_1}^* \wedge y_{r_5}^* \wedge y_{r_7}^* \wedge y_{r_2}^{t_1} \wedge y_{r_3}^{t_1} \wedge y_{r_4}^{t_2} \wedge y_{r_6}^{t_3} \wedge y_{r_8}^* \wedge y_{r_9}^* \wedge \overline{y_{r_2}^*} \wedge \overline{y_{r_3}^*} \wedge \overline{y_{r_4}^*} \wedge \overline{y_{r_6}^*} \wedge \overline{y_{r_8}^{t_4}} \wedge \overline{y_{r_9}^{t_5}} \wedge \overline{y_{r_9}^*}} \\ &= \mathbf{ff}. \end{aligned}$$

Thus, we conclude that transitions t_4 and t_5 are indeed mutually exclusive. \diamond

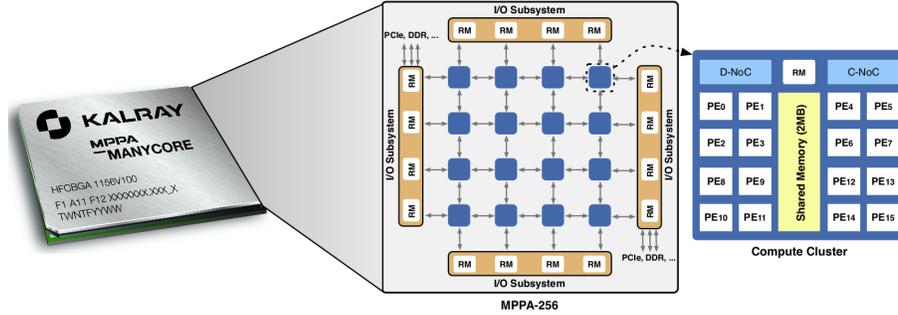


Figure 15

Figure 16: Kalray MPPA-256 many-core architecture

3.3 Priority

As shown in Section 3.1, conflict-free cfNets are confluent: the same platform constraint is obtained by any run of the cfNet, for a given initial marking. In other words, enabled transitions can be fired in arbitrary order. This is not the case for cfNets with conflicts: firing one of two conflicting transitions disables the other one, generating different platform constraints. Thus, for reachable conflicts, the choice of which of the two conflicting transitions should be fired, has to be resolved externally to the cfNet. This can be achieved by introducing priority among the conflicting transitions.

For a cfNet $N = (\mathcal{R}, T, F, I, C)$, a priority relation is a partial order $> \subseteq T \times T$ on its set of transitions. For two transitions t_1 and t_2 , a priority $t_1 > t_2$ means that when both transitions are enabled, t_1 must be fired before t_2 . Priorities can be defined statically or dynamically, depending, for example, on the availability of the resources corresponding to the post-places of the two transitions.

4 Case-study: Kalray architecture with cfNets

Our case-study example is inspired by the many-core architecture of Kalray MPPA-256 [14], which consists of 256 processing elements (PEs) or cores grouped in compute clusters, each consisting of 16 PEs communicating through a shared memory, which consists of 16 independent memory banks of 128KB organised in two sides, *left* and *right* (see Figure 16). For simplicity, in this report we will consider a version of a single cluster of Kalray architecture composed of four PEs and four memory banks (see Figure 17), that can be extrapolated to a complete architecture of one cluster.

In general, two processing cores of one cluster cannot access the same memory bank at the same time. Processing cores are organised in pairs, each pair shares two data-buses, one for each of the memory sides [10]. Therefore, the access to memory banks is arbitrated by two stages of arbiters implementing Round Robin (RR) arbitration policy. Our goal in this report is to allocate PEs, buses and memory banks such that there will be at most one request for any arbiter queue making the resource unavailable otherwise. This way, we assume that two cores of one pair can access different memory sides simultaneously and two processors from two different pairs may access different memory banks of the same side.

Figure 18 presents a cfNet modelling the architecture. For the sake of clarity, we group the resources and some transitions between them in several boxes: one for each group of processors, one for each memory side and one for each bus. The small rectangles in each box represent the

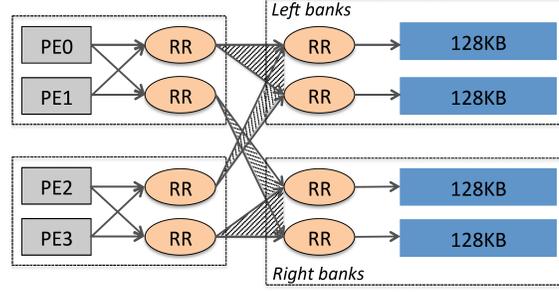


Figure 17: Simplified architecture of a compute cluster with arbitration points

corresponding resource ports or interfaces: for the processor box, the port means both p_1 and p_2 (or p_3 and p_4). For the memory side box, there is one port for each memory bank. The bus box has one port for both processors and a port per each corresponding memory bank. Thus, the arrow between the port of the p_1 - p_2 group and the middle port of the bus_{L12} group represents four arcs: one from each of the places p_1 and p_2 , to each of the transitions t_{61} and t_{62} . With this notation, for transition t_{61} to be enabled, there must be a token in p_1 , p_2 , and m_1 .

It is possible to make a request for either a virtual processing core p or a specific one out of p_1, p_2, p_3 or p_4 . Similarly, it is possible to ask for either a virtual memory m , or specific memory side L or R , or a specific memory bank out of m_1, m_2, m_3 or m_4 .

The architectural constraints are modelled as follows: 1) transition t_1 ensures mutual exclusiveness on processors p_1, p_2, p_3 and p_4 ; 2) transition t_2 ensures the mutual exclusiveness on memory sides, and t_3, t_4 on memory banks of each side respectively; 3) transitions $t_{51}, t_{52}, t_{61}, t_{62}, t_{71}, t_{72}, t_{81}$ and t_{82} ensure that only one of the processors from one group can have access to one memory side using a dedicated bus.

The resource allocation constraints for the existing transitions implementing the dispatching policy are as follows:

$$\begin{aligned}
c_{t_1} = & (x_{p_1} = x_p \wedge x_{p_2} = 0 \wedge x_{p_3} = 0 \wedge x_{p_4} = 0) \\
& \vee (x_{p_2} = x_p \wedge x_{p_1} = 0 \wedge x_{p_3} = 0 \wedge x_{p_4} = 0) \\
& \vee (x_{p_3} = x_p \wedge x_{p_1} = 0 \wedge x_{p_2} = 0 \wedge x_{p_4} = 0) \\
& \vee (x_{p_4} = x_p \wedge x_{p_1} = 0 \wedge x_{p_2} = 0 \wedge x_{p_3} = 0); \\
c_{t_2} = & (x_L = x_m \wedge x_R = 0) \vee (x_R = x_m \wedge x_L = 0); \\
c_{t_3} = & (x_{m_1} = x_L \wedge x_{m_2} = 0) \vee (x_{m_2} = x_L \wedge x_{m_1} = 0);
\end{aligned}$$

constraint c_{t_4} is build similarly to c_{t_3} ;

$$\begin{aligned}
c_{t_{51}} = & (x_{bus_{L34}} > 0 \wedge x_{m_1} > 0 \wedge \\
& ((x_{p_3} = 0 \wedge x_{p_4} > 0) \vee (x_{p_4} = 0 \wedge x_{p_3} > 0))) \\
& \vee (x_{bus_{L34}} = 0 \wedge x_{p_3} = 0 \wedge x_{p_4} = 0) \\
& \vee (x_{bus_{L34}} = 0 \wedge x_{m_1} = 0);
\end{aligned}$$

constraints $c_{t_{52}}, c_{t_{61}}, c_{t_{62}}, c_{t_{71}}, c_{t_{72}}, c_{t_{81}}$ and $c_{t_{82}}$ are build similarly to $c_{t_{51}}$.

The initial cost functions for the resources are:

for $i \in 1..4$,

$$cost_{p_i}(d) = \begin{cases} 0, & d \in \{0, 1\}, \\ \perp, & \text{otherwise,} \end{cases}$$

for $i \in 1..4$,

$$cost_{m_i}(d) = \begin{cases} 0, & 0 \leq d \leq 128, \\ \perp, & \text{otherwise,} \end{cases}$$

for $i \in \{L, R\}$ $j \in \{12, 34\}$,

$$cost_{bus_{ij}}(d) = \begin{cases} 0, & d \in \{0, 1\}, \\ \perp, & \text{otherwise,} \end{cases}$$

for p, m, L, R the cost function is defined everywhere as 0.

Consider a different model of “virtual” resources, shown in Figure 19 representing two processing cores of Kalray architecture. The constraint schemata associated, respectively, to transitions t_1 and t_2 are $c_{t_1} = (x_p = x_{p_1})$ and $c_{t_2} = (x_p = x_{p_2})$.

In the dispatching allocation of Figure 18, the constraint schemata ensured that only one core can be allocated for a single request. In the cfNet of Figure 19, this is ensured by the inhibitor arcs (p_1, t_1, t_2) and (p_2, t_2, t_1) . The initial marking for the request of a “virtual” processing core p is shown in Figure 19a. Figures 19b and 19c show the two possible runs of the cfNet, where the firing of transition t_1 inhibits the firing of transition t_2 and vice versa.

Notice that the constraint schemata associated to the transitions t_1 and t_2 involve less variables than the dispatching schema in Figure 18, simplifying the task of the constraint solver.

The fact that a virtual resource is used to represent the two cores implies that they are functionally equivalent. However, we can consider a scenario, where the two cores differ in their non-functional properties. For instance, suppose that p_1 has better energy efficiency than p_2 . Imposing the priority $t_1 > t_2$ for the cfNet in Figure 19a, would ensure that p_1 is allocated rather than p_2 .

Consider now the second scenario, where two applications are running on this platform, both requiring a processing core, but unaware of the platform architecture. In the first cycle, one of the applications requests p and p_1 is allocated as discussed above. In the next cycle, the second application also requests p . Since p_1 is not available (indicated by its cost function being undefined for all values), we inverse the priority, setting $t_1 < t_2$. Thus, t_2 will be fired leading to an allocation of p_2 to the second application.

Finally, notice that, if none of p_1 and p_2 is available, the choice of priority is irrelevant, since the constraint problems generated from both markings in Fig. 19b and 19c will be unsatisfiable.

5 Conclusion

In this report, we have introduced Constraint-Flow Nets (cfNets) that allow modelling of resource dependencies, thereby bridging a gap in the current state-of-the-art approaches to the specification of requests for resources and resource allocation to applications: resource allocation commonly relies on the assumption that the requested resources are completely specified, whereas specification languages operate with high-level abstractions of resources, e.g. “memory” or “thread”, leaving the resource manager to endow these with precise semantics.

The cfNet model provides a means to formally specify the structure of resources provided by the platform and the dependencies among them. This specification serves as an abstraction layer, which allows designers to focus on the resources immediately relevant to the application functionality, while taking care of low-level structure and dependencies inherent to the specific target platform. Thus, our approach simplifies application design and greatly enhances portability. This is particularly useful for platforms with complex resource architectures, such as the Massively Parallel Processor Arrays (MPPA) (e.g. Kalray), or cloud platforms, where the resource architecture can change dynamically at run time.

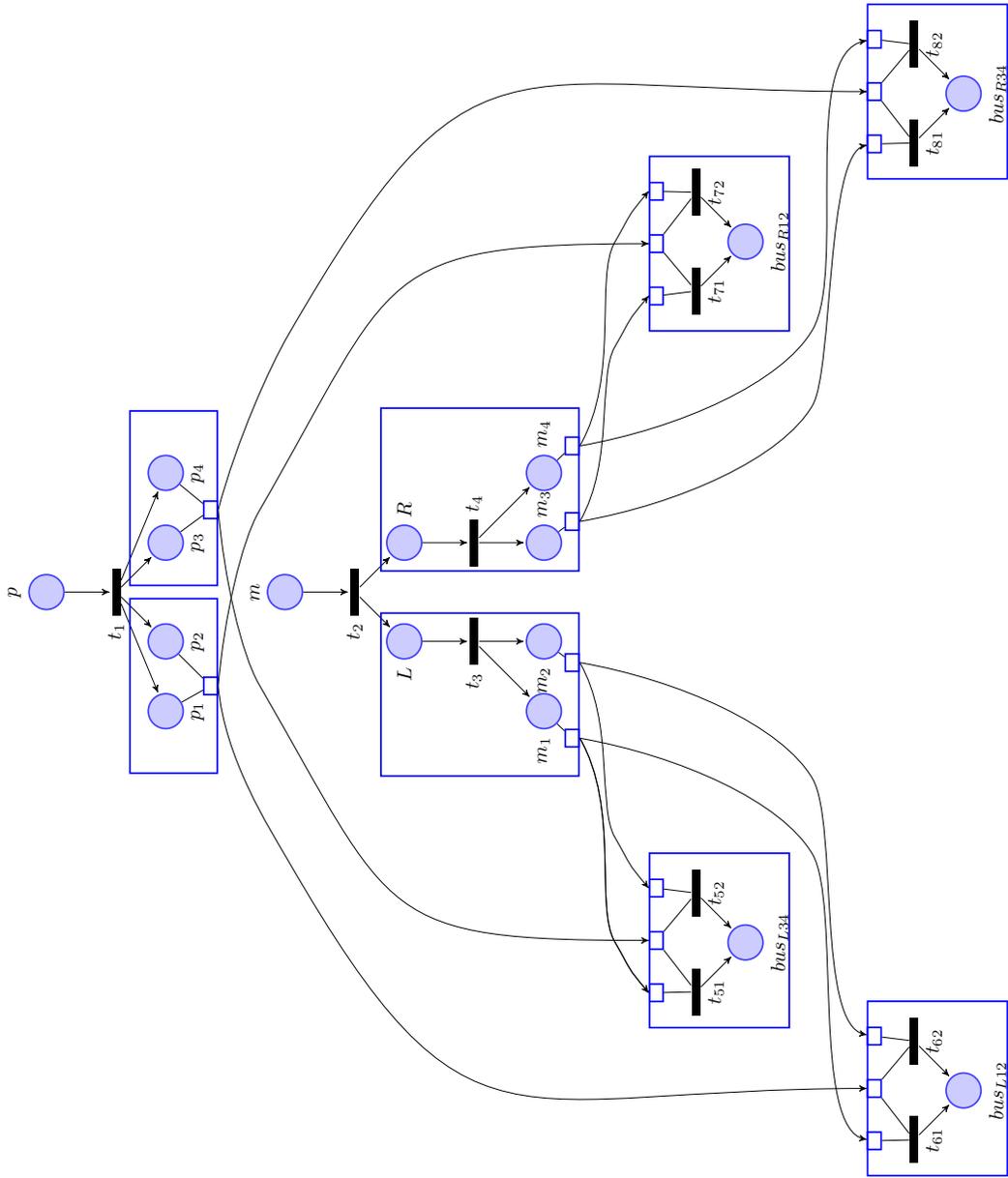


Figure 18: A complete cfNet modelling a single cluster of the Kalray architecture

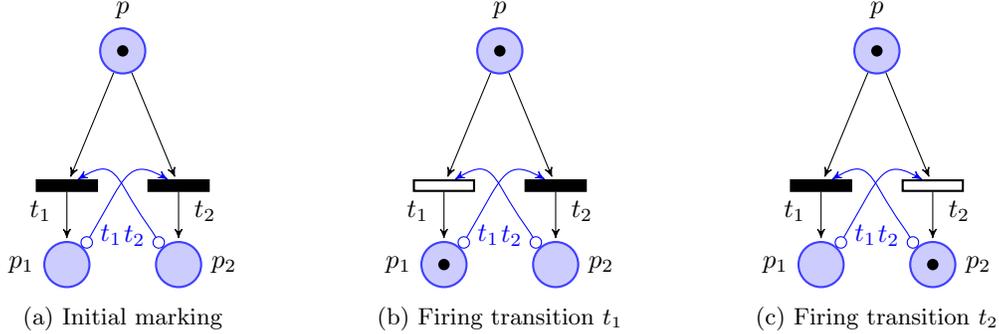


Figure 19: The cfNet modelling virtual processing cores using inhibitor arcs

In this report, we have defined the cfNet model and have provided its semantics, defining a constraint problem for a given initial resource request. The cfNet model allows for the use of inhibitor arcs. On one hand, these increase the expressiveness of the model and simplify certain constraint problems by reducing the number of variables involved. On the other hand, inhibitors can generate conflicts introducing ambiguity in the constraint problem definition. We have provided a sufficient condition, which can be easily checked syntactically, for the cfNet to be conflict-free. For cfNets that do not satisfy this condition, we have provided an efficient method for determining whether a given inhibitor induces a conflict. This is achieved in three steps: 1) given two transitions, we encode the existence of a conflict as a Boolean formula 2) a satisfactory valuation of this formula represents a marking exhibiting the conflict; 3) by analysing the causality hyper-graph for cycles, we check whether this marking is reachable and, if not, refine the encoding to exclude it. These steps are repeated until a reachable marking exhibiting the conflict is found or the Boolean encoding becomes unsatisfiable.

In our future work, we are planning to further improve the conflict detection algorithm: the encoding refinement in step 3) excludes only one marking; by exploiting the causality hyper-graph more unreachable markings could be excluded in one step. Reachable conflicts are resolved by defining priorities between conflicting transitions. Dynamic priorities can only be resolved at run time. For static priorities, we are currently working on a cfNet transformation to statically eliminate the corresponding conflicts.

Finally, we are planning to implement the cfNet model in the JavaBIP [4, 5] component coordination framework and use it in conjunction with the JaCoP ² constraint solver.

References

- [1] M. K. Agarwal, K. Appleby, M. Gupta, G. Kar, A. Neogi, and A. Sailer. Problem determination using dependency graphs and run-time behavior models. In A. Sahai and F. Wu, editors, *Utility Computing: Proceedings of the 15th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2004)*, volume 3278 of *Lecture Notes in Computer Science*, pages 171–182. Springer Berlin Heidelberg, 2004.
- [2] C. Ansótegui, M. Bofill, M. Palahí, J. Suy, and M. Villaret. Satisfiability modulo theories: An efficient approach for the resource-constrained project scheduling problem. In *Symposium on Abstraction, Reformulation, and Approximation*, 2011.

²<http://jacop.osolpro.com/>

- [3] D. A. Berson, R. Gupta, and M. L. Soffa. GURRR: A global unified resource requirements representation. *SIGPLAN Not.*, 30(3):23–34, Mar. 1995.
- [4] S. Bliudze, A. Mavridou, R. Szymanek, and A. Zolotukhina. Coordination of software components with BIP: Application to OSGi. In *Proceedings of the 6th International Workshop on Modeling in Software Engineering*, MiSE 2014, pages 25–30, New York, NY, USA, 2014. ACM.
- [5] S. Bliudze, A. Mavridou, R. Szymanek, and A. Zolotukhina. Exogenous coordination of concurrent software components with JavaBIP. *Softw. Pract. Exper.*, 2016. Under review.
- [6] A. A. Chien, H. Casanova, Y.-s. Kee, and R. Huang. The virtual grid description language: vgDL. Technical Report CS2005-0817, Department of Computer Science and Engineering, University of California, San Diego, 2004.
- [7] J. M. Colom. The resource allocation problem in flexible manufacturing systems. In W. M. P. Aalst and E. Best, editors, *Applications and Theory of Petri Nets 2003: 24th International Conference*, volume 2679 of *Lecture Notes in Computer Science*, pages 23–35. Springer Berlin Heidelberg, June 2003.
- [8] Y. Cui and K. Nahrstedt. QoS-aware dependency management for component-based systems. In *High Performance Distributed Computing, 2001. Proceedings. 10th IEEE International Symposium on*, pages 127–138, 2001.
- [9] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke. A resource management architecture for metacomputing systems. In D. G. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing: IPPS/SPDP'98 Workshop*, volume 1459 of *Lecture Notes in Computer Science*, pages 62–82. Springer Berlin Heidelberg, 1998.
- [10] B. D. de Dinechin, D. van Amstel, M. Poulhiès, and G. Lager. Time-critical computing on a single-chip massively parallel processor. In *Proceedings of the Conference on Design, Automation & Test in Europe, DATE '14*, pages 97:1–97:6, 3001 Leuven, Belgium, Belgium, 2014. European Design and Automation Association.
- [11] C. Ensel and A. Keller. An approach for managing service dependencies with XML and the resource description framework. *Journal of Network and Systems Management*, 10(2):147–170, June 2002.
- [12] J. Ezpeleta, J. Colom, and J. Martínez. A Petri net based deadlock prevention policy for flexible manufacturing systems. *IEEE Transactions on Robotics and Automation*, 11:173–184, 04/1995 1995.
- [13] G. Giannopoulou, N. Stoimenov, P. Huang, L. Thiele, and B. de Dinechin. Mixed-criticality scheduling on cluster-based manycores with shared communication and storage resources. *Real-Time Systems*, pages 1–51, 2015.
- [14] Kalray. Kalray MPPA-256. http://www.kalray.eu/IMG/pdf/FLYER_MPPA_MANYCORE.pdf, Mar. 2015.
- [15] Y.-S. Kee, D. Logothetis, R. Y. Huang, H. Casanova, and A. A. Chien. Efficient resource description and high quality selection for virtual grids. In *CCGRID*, pages 598–606. IEEE Computer Society, 2005.

- [16] A. A. Kountouris and C. Wolinski. Hierarchical conditional dependency graphs for conditional resource sharing. In *Euromicro Conference, 1998. Proceedings. 24th*, volume 1, pages 313–316 vol.1, Aug 1998.
- [17] K. Krauter, R. Buyya, and M. Maheswaran. A taxonomy and survey of grid resource management systems for distributed computing. *Software: Practice and Experience*, 32(2):135–164, 2002.
- [18] O. Lassila and R. R. Swick. Resource description frame-work (RDF) model and syntax specification. Technical Report REC-rdf-syntax-19990222, World Wide Web Consortium (W3C), Feb. 1999.
- [19] J.-P. López-Grao and J.-M. Colom. A Petri net perspective on the resource allocation problem in software engineering. In K. Jensen, S. Donatelli, and J. Kleijn, editors, *Transactions on Petri Nets and Other Models of Concurrency V*, volume 6900 of *Lecture Notes in Computer Science*, pages 181–200. Springer Berlin Heidelberg, 2012.
- [20] R. Raman, M. Livny, and M. Solomon. Matchmaking: An extensible framework for distributed resource management. *Cluster Computing*, 2(2):129–138, Sept. 1999.
- [21] P. Senkul and I. H. Toroslu. An architecture for workflow scheduling under resource allocation constraints. *Information Systems*, 30(5):399 – 422, 2005.
- [22] P. Tendulkar, P. Poplavko, I. Galanommatis, and O. Maler. Many-core scheduling of data parallel applications using SMT solvers. In *Digital System Design (DSD), 2014 17th Euromicro Conference on*, pages 615–622. IEEE, 2014.
- [23] P. Tendulkar, P. Poplavko, J. Maselbas, I. Galanommatis, and O. Maler. A runtime environment for real-time streaming applications on clustered multi-cores. Technical report, Verimag, 2015.
- [24] F. Tricas, F. Garcia-Valles, J. M. Colom, and J. Ezpeleta. A Petri net structure-based deadlock prevention solution for sequential resource allocation systems. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 271–277, April 2005.
- [25] H. N. Van, F. D. Tran, and J. M. Menaud. SLA-aware virtual resource management for cloud infrastructures. In *Computer and Information Technology, 2009. CIT '09. Ninth IEEE International Conference on*, volume 1, pages 357–362, Oct 2009.
- [26] J. Vanderham, F. Dijkstra, F. Travostino, H. Andree, and C. Delaat. Using RDF to describe networks. *Future Generation Computer Systems*, 22(8):862–867, Oct. 2006.