Normal Forms and Equivalence of K-periodically Routed Graphs

Anthony Coadou, Robert De Simone

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Normal Forms and Equivalence of K-periodically Routed Graphs

Anthony Coadou — Robert de Simone

N° 7286

May 2010

Embedded and Real Time Systems

ISSN 0249-6399 ISRN INRIA/RR--7286--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SOPHIA ANTIPOLIS - MÉDITERRANÉE

Normal Forms and Equivalence of K-periodically Routed Graphs

Anthony Coadou , Robert de Simone

Theme : Embedded and Real Time Systems
Algorithmics, Programming, Software and Architecture
Équipe-Projet Aoste

Rapport de recherche n° 7286 — May 2010 — 36 pages

Abstract: We introduce K-periodically Routed Graphs, which are extensions of Marked Graphs with routing nodes, governed by ultimately periodic binary sequences. We study data relations and dependencies, as well as equational transformations of the network topology. We show the existence of expanded normal forms. We prove that some transformations preserve external flow equivalence. Issues arising from internal flow interleavings and permutations are also tackled.

Key-words: Process network, k-periodically routed graphs, marked graphs, SDF graphs, semantics, normal form

Centre de recherche INRIA Sophia Antipolis – Méditerranée
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Formes normales et équivalence de graphes à routage k-périodique

Résumé : Nous définissions la sémantique formelle des graphes à routage k-périodique, une extension des graphes marqués contenant des nœuds de routage dirigés par des séquences binaires ultimement périodiques. Nous étudions les relations et dépendances de données. Nous présentons également des transformations topologiques, et prouvons qu'elles préservent les équivalences de flots internes. Nous prouvons l'existance de formes normales. Enfin, nous abordons les problèmes d'entrelacements de flots et de permutations de jetons.

Mots-clés : Réseau de processus, graphe à routage k-périodique, graphe marqué, graphe SDF, sémantique, forme normale

1 Introduction

Dataflow Process Networks (PN) form an important class of formal models for the mathematical study of concurrency phenomena. They generally insist on conflict-freeness and predictivity of behaviors, by demanding that control switch decisions are taken according to internal component states only.

Simple PN models such as Marked Graphs (MG) [9] and their Synchronous Data Flow (SDF) [17] extension simply discard all control switch abilities, what is called pure dataflow. As a result, they provide a rich theory for regular static scheduling of computation node firings. By leaving the switching patterns of control nodes unspeficied, models such as Kahn PNs [13] or Boolean Data Flow (BDF) graphs [7] lose this static predictivity.

In previous works [5, 6], we have introduced a new PN model, named K-periodically Routed Graphs (KRG). The model introduces two kinds of routing nodes that operate as demuxes and muxes on channels, just as in BDF. But in our case, the switch conditions are made explicit, and additionally they are governed by ultimately periodic binary sequences.

The idea of having designers use explicitly infinite binary words to represent schedules and logical clocks was central in the theory of synchronous languages [3]. The specialization to the repetitive case, with ultimately periodic sequences, was introduced in the definition of n-synchronous formalisms [8].

KRGs can be shown to encompass other PN models, with a finer level of granularity though: one execution being represented by a sequence of KRG firings. Similarly, SDF can be expanded in MGs, while duplicating node occurrences [2]. They essentially express the fact that internal states can be expanded by unrolling the global systems according to various phases.

Our Contribution

After recalling in Section 2 the definitions of n-synchronous models and dataflow graphs, we introduce in Section 3 KRGs and the way their abstraction into SDF models allows to check for balanced throughput and various correctness properties

Then we show in Section 4 how the natural operational semantics leads to an expansion into quasi-Marked Graphs. Here the "quasi" prefix stands for the fact that the initial phase allowed for switching paterns in KRGs may differ for the ones in periodic phases, which is reflected here.

Finally, we present in Section 5 how this semantic interpretation is used when introducing algebraic graph transformations based on the merge and select nodes; these transformations are first proved sound, that is semantic-preserving as to the former expansion. Then we strive at defining a second normal form result, this time inside KRGs: because some of our transformations have a strong intuitive understanding as sharing/unsharing of links, a typical normal form characterization will consist in the one where channels are maximally distributed, as in a crossbar. We establish that our transformation rules are complete, in the sense that they allow to get from any KRGs to its normal form by succesive adequate applications of such transformations.

2 Preliminary Definitions

2.1 Binary Sequences and N-synchronous Theory

We first introduce some notations, borrowed from the n-synchronous theory and previous works on routed graphs [6, 8]:

- Let $\mathbb{B} = \{0, 1\}$ be the set of Boolean values, $\mathbb{B}^*$ and $\mathbb{B}^\omega$ respectively the sets of finite and infinite binary words. Let ε be the empty word.
- We note $|w|$ the length of w. Similarly, we note respectively $|w|_0$ and $|w|_1$ the number of 0's and 1's in w.
- w_i is the i^{th} letter of w. We also use w_{head} for w_1, and w_{tail} for the unique word such that $w = w_{head}.w_{tail}$.
- The prefix of length n of a word w is defined as $\mathrm{prf}\,(w, n) = w_1 \dots w_n$.
- We note $\mathrm{idx}_b\,(w, i)$ the position of the i^{th} $b \in \mathbb{B}$ in w. For instance, $\mathrm{idx}_1\,(010110\mathbf{1}10, 4) = 7$. By convention, $\mathrm{idx}_b\,(w, i) = +\infty$ when $|w|_b < i$.
- A sequence w in $\mathbb{B}^\omega$ is said to be *ultimately periodic*, or *k-periodic*, if it is of the form $u.v^\omega$, with $u, v \in \mathbb{B}^*$. We call u the *initial* part and v the *periodic* part. Let $\mathbb{P}$ be the set of all ultimately periodic words. We note $\mathbb{P}_p^k$ the set of k-periodic words with a steady part of length p containing k occurrences of 1. For such a word, we call k its *periodicity* and p its *period*.

Then we define the *On* and *When* operators. They are useful for routing and clock sampling.

Definition 1 (*On* Operator)**.** *The On operator is noted* $\blacktriangledown$*, and recursively defined on binary words as follows:* $\forall\, n \in \mathbb{N},\ \forall\, u \in \mathbb{B}^n,\ \forall\, v \in \mathbb{B}^{|u|_1}$,

$$\begin{aligned} \varepsilon \blacktriangledown \varepsilon &= \varepsilon \\ u \blacktriangledown v &= \begin{cases} 0.\,(u_{tail} \blacktriangledown v) & \text{if } u_{head} = 0 \\ v_{head}.\,(u_{tail} \blacktriangledown v_{tail}) & \text{if } u_{head} = 1 \end{cases} \end{aligned}$$

Definition 2 (*When* Operator)**.** *The When operator is noted* $\triangle$*, and recursively defined on binary words as follows:* $\forall\, n \in \mathbb{N},\ \forall\, u, v \in \mathbb{B}^n$,
if $n = 0$,

$$u \triangle v = \varepsilon \triangle \varepsilon = \varepsilon$$

otherwise,

$$u \triangle v = \begin{cases} u_{tail} \triangle v_{tail} & \text{if } v_{head} = 0 \\ u_{head}.\,(u_{tail} \triangle v_{tail}) & \text{if } v_{head} = 1 \end{cases}$$

Equivalently: $\forall\, n \in \mathbb{N},\ \forall\, u, v \in \mathbb{B}^n,\ \exists\, w \in \mathbb{B}^{|v|_1}$,

$$u \triangle v = w \ \Leftrightarrow\ \forall\, i \in [\![1, |w|]\!],\ w_i = u_{\mathrm{idx}_1(v,i)}$$

Despite *On* and *When* operators are defined over finite binary words, we can extend their definitions to infinite binary sequences, hence removing halting conditions. In particular, calculus over k-periodic words simply consists in studying the behavior over the initial part, and then a period.

2.2 Marked and SDF Graphs

A Process Network (PN) consists in a finite directed bigraph, with a set of *computation nodes* (or *transitions* in Petri Net terminology), and a set of *places* storing data. As the focus of PNs is more on the ordering of computations and communications than on the actual data values, these data values are abstracted as *tokens*. An integer assignment of tokens to places is called a *marking*.

Our model, presented in Section 3, is an extension of Commoner and Holt's Marked Graphs [9]; a special case of Petri Nets where each place has exactly one producer and one consumer. Each node consumes and produces respectively a single token from each input and output place respectively.

Given a place p, we note $\bullet p$ and $p\bullet$ its input and output node respectively (both are unique). Similarly for a node n, we note $\bullet n$ and $n\bullet$ its sets of input and output places.

Definition 3 (Marked Graph)**.** *A Marked Graph is a 4-tuple $\langle \mathcal{N}, \mathcal{P}, \mathcal{T}, M \rangle$ such that:*

- *$\mathcal{N}$ is a finite set of nodes,*
- *$\mathcal{P}$ is a finite set of places,*
- *$\mathcal{T} \subseteq (\mathcal{N} \times \mathcal{P}) \cup (\mathcal{P} \times \mathcal{N})$ is a finite set of edges, linking nodes and places,*
- *$M : \mathcal{P} \to \mathbb{N}$ is a function assigning an initial marking to places.*

and with the following constraint:

$$\forall\, p \in \mathcal{P},\ \ |\bullet p| = |p\bullet| = 1 \tag{1}$$

In *Synchronous Data Flow* (SDF) graphs [16, 17], weights are added: a fixed number of tokens is consumed and produced on its channels by a computation node. Systems are said to be *balanced* when production equals consumption over a *period* for all computation nodes, such that channels remain bounded during execution of computation nodes. The special case of Marked Graphs, where all weights equal 1, is called *homogeneous* SDF.

Definition 4 (SDF Graph)**.** *A SDF graph is a 5-tuple $\langle \mathcal{N}, \mathcal{P}, \mathcal{T}, M, W \rangle$ such that:*

- *$\langle \mathcal{N}, \mathcal{P}, \mathcal{T}, M \rangle$ is a Marked Graph.*
- *$W : \mathcal{P} \to (\mathbb{N} \times \mathbb{N})$ is a weight function, assigning to each place the number of tokens produced and consumed when firing $\bullet p$ and $p\bullet$ respectively.*

The main common feature of both previous models is *conflict-freeness*: once a local firing is enabled, it shall not be disabled unless the node is executed.

3 K-periodically Routed Graphs

Marked Graphs can only model systems without routing or branching patterns. We extend their expressivity as K-periodically Routed Graphs (KRGs), then introducing routing nodes driven by k-periodic binary sequences. They allow modeling of non-uniform routings and computations. First we define KRGs, as well as their semantics. They we tackle the issues of graph boundedness and liveness.

3.1 Definition and Semantics

So far Marked Graphs and SDF models consider uniform data paths. Boolean Data Flow (BDF) graphs [7] introduce two kinds of switching nodes, here called *merge* and *select*. A merge node connects to a single output channel and exactly two input places, identified by a 1 and and 0 label respectively. An internal oracle sequence, kept implicit, will indicate when tokens are to be consumed from the 0-labelled or 1-labelled place respectively; thus an oracle is an infinite binary word. Similarly, a select node will connect to a single input, and two output channels. This time the internal oracle sates whether the next token is redirected towards the 0-labelled or the 1-labelled input.

KRGs will essentially be homogeneous BDF models where the oracles are made explicit in a specific class of infinite binary words. We will thus first introduce these specific ultimately periodic binary words, then use them in our definition of KRGs.

Definition 5. *A* k-periodically routed graph *is a 5-tuple* $\langle \mathcal{N}, \mathcal{P}, \mathcal{T}, M, R \rangle$ *such that:*

- $\mathcal{N}$ *is a finite set of nodes, and composed of four distinct subsets:*
  - $\mathcal{N}_\lambda$*, set of* computation *nodes.*
  - $\mathcal{N}_c$*, set of* copy *nodes, such that:* $\forall\, n \in \mathcal{N}_c, |\bullet n| = 1$ *and* $|n\bullet| > 1$
  - $\mathcal{N}_s$*, set of* select *nodes, such that:* $\forall\, n \in \mathcal{N}_s, |\bullet n| = 1$ *and* $|n\bullet| = 2$
  - $\mathcal{N}_m$*, set of* merge *nodes, such that:* $\forall\, n \in \mathcal{N}_m, |\bullet n| = 2$ *and* $|n\bullet| = 1$
- $\mathcal{P}$ *is a finite set of* places*, such that:*

$$\forall\, p \in \mathcal{P}, \quad |\bullet p| = |p\bullet| = 1$$

- $\mathcal{T} \subseteq (\mathcal{N} \times \mathcal{P}) \cup (\mathcal{P} \times \mathcal{N})$ *is a finite set edges between nodes and places.*
- M *is a function assigning an initial marking to each place.* $M : \mathcal{P} \to \mathbb{N}$
- R *is a function assigning a routing sequence to select and merge nodes.* $R : \mathcal{N}_s \cup \mathcal{N}_m \to \mathbb{P}$

Copy nodes can be seen as a special case of computation nodes. However we singled them out, as they play a specific role: they specify data duplications without information altering. Places are simply point-to-point channels, as in usual Marked Graphs, temporarily buffering tokens. The routing patterns on merge (resp. select) nodes indicate on which place the next token is to be consumed (resp. produced) among the two node inputs (resp. outputs).

Considering a select node (resp. merge node), we note $n\bullet_0$ and $n\bullet_1$ its 0- and 1-labelled outputs (resp. $\bullet n_0$ and $\bullet n_1$ its 0- and 1-labelled inputs).

We will also need the notation $\lambda(n)$, for any computation node n, indicating an abstract *label* assigned to n. This is useful in order to model real-life designs; for instance, the label may represents the function or operator applied by n to its input data.

Rule 6 (Enabling)**.**

- *A computation, copy or select node is enabled if and only if each input place holds at least one token.*
- *A merge node is enabled if and only if its input place, given by the routing sequence, holds at least one token.*

Rule 7 (Firing)**.**

- *Computation or copy node firing consumes a token in each input place. It produces a token into each output place.*
- *Merge node firing consumes the first letter of the routing sequence, and a token in the corresponding input place. It produces a token into the output place.*
- *Select node firing consumes a token in its input place and the first letter of the routing sequence ; it produces a token in the corresponding output place.*

KRG is a conflict-free and confluent model, in which once enabled, any node remains so until it fires.

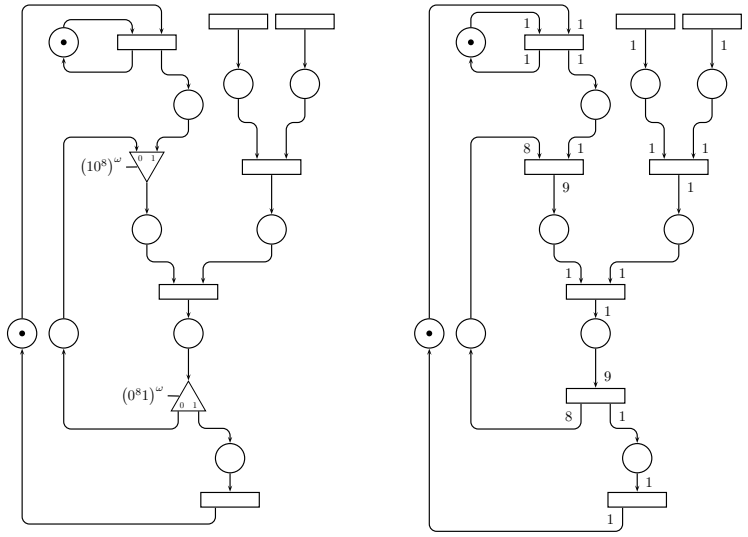
3.2 Boundedness

Boundedness is of crucial importance for real-life implementation of Process Networks: the problem consists in "balancing" token traffic in order that consumption equals production over all paths. It depends on both token routing and choice of orderings between fireable nodes, summarized as "*is there a periodic sequence of firings such that the number of tokens in each place remains bounded ?*"

This problem of determining place bounds and proper firing rates can be elegantly checked in SDF using the so-called *balance equations*. There exists a natural abstraction from KRGs to SDF graphs which makes these results available also for checking boundedness on KRGs, as follows:

Rule 8 (Abstracting KRG into SDF graph)**.** *Let $\langle \mathcal{N}, \mathcal{P}, \mathcal{T}, M, R\rangle$ be a KRG. Its SDF abstraction $\langle \mathcal{N}', \mathcal{P}', \mathcal{T}', M', W\rangle$ is such that:*

$$\begin{array}{rcl} \mathcal{N}' & \stackrel{def}{=} & \mathcal{N} \\ \mathcal{P}' & \stackrel{def}{=} & \mathcal{P} \\ \mathcal{T}' & \stackrel{def}{=} & \mathcal{T} \\ M' & \stackrel{def}{=} & M \end{array}$$



(a) (b)

Figure 1: Abstracting (a) a KRG into (b) a SDF graph.

$\forall\, p \in \mathcal{P}$, *let* $W(p) = (w_o, w_d)$ *be:*

$$w_o = \begin{cases} 1 & \text{if } \bullet p \in \mathcal{N}_\lambda \cup \mathcal{N}_c \\ |R(\bullet p)|_b & \text{if } \bullet p \in \mathcal{N}_s \\ |R(\bullet p)| & \text{if } \bullet p \in \mathcal{N}_m \end{cases} \qquad w_d = \begin{cases} 1 & \text{if } p\bullet \in \mathcal{N}_\lambda \cup \mathcal{N}_c \\ |R(p\bullet)| & \text{if } p\bullet \in \mathcal{N}_s \\ |R(p\bullet)|_b & \text{if } p\bullet \in \mathcal{N}_m \end{cases}$$

where $b \in \mathbb{B}$ *is the label of the select (resp. merge) node output (resp. input).*

The KRG to SDF graph transformation leaves graph topology and initial marking unchanged. However, merge and select nodes are assimilated to non-homogeneous SDF nodes, whose productions and consumptions equal the corresponding number of "0" or "1" of the routing sequence over a period.

Then, solving balance equations on the transformed SDF model allows to establish boundedness, and to compute buffer bounds as well as matching firing rates.

Proposition 9 (Abstraction correctness)**.** *KRG abstraction into SDF is correct.*

Corollary 10. *A KRG is bounded if and only if its SDF abstraction is.*

3.3 Liveness

Potential deadlock due to buffer emptyness is also an issue, related to Petri Net *liveness* property. For instance in Figure 1 (a), if the prefix "0" is concatened to the merge routing sequence, then a deadlock is introduced: a token flows down to the right path, while the merge node awaits it first from the left.

The intuitive solution for checking liveness in a bounded KRG consists in computing the exhaustive reachable state space: boundedness and k-periodicity

imply a finite reachable state space. Hence we can simulate the graph execution till reaching a starvation (deadlock) or an already-visited state. In the latter case, liveness is enforced.

We will see in deeper details how that can be formally defined and studied in Section 4.

4 Token Flows and Graph Equivalences

Because switching conditions evolve in time, succesive tokens do not follow identical routes. Connecting directly computation outputs to next compution nodes thus requires an unfolding of these nodes. Still, because the switching paterns are ultimately k-periodic, the same configurations are bound to eventually occur, so that the expansion can be limited to an initial MG, then followed by a repeated periodic MG, both finite. We call these *quasi-Marked Graphs*.

This expansion provide a semantic interpretation tha twiill be used later: two KRGs are considered as behaviorally equivelent if they translate by this expansion into equivalent quasi-MGs.

In this section, we study relations over token flows, modeled as dependence graphs. Then we define equivalence classes, allowing KRG comparisons. Finally, we define a first notion of KRG normal form.

4.1 Token Flows

We shall first introduce notations to express in formal terms how the successive tokens flowing through a place will later reach another place: possibly not all of them, duplicated and/or disordered.

Definition 11 (Token Flow)**.** *Let $\langle \mathcal{N}, \mathcal{P}, \mathcal{T}, M, R\rangle$ be a KRG, and $t = (x, y)$ an edge of $\mathcal{T}$. Let t_i, with $i \in \mathbb{N}^*$, be the i^{th} token flowing from x to y through t. For all i, t_i define a sequence of integers, corresponding to their indices in the output token sequence. We call this sequence the token flow of t.*

Definition 12 (Elementary Flow Relations)**.** *Let $\langle \mathcal{N}, \mathcal{P}, \mathcal{T}, M, R\rangle$ be a KRG, and $t = (x, y)$ and $t' = (y, z)$ be two edges of $\mathcal{T}$. Let us consider a token produced by x and flowing towards z, via y, through the path $t \rightsquigarrow t'$. There exists a set of flow relations of the form $\curvearrowright_y \subset \mathbb{N}^* \times \mathbb{N}^*$, such that $i \curvearrowright_y j$, where i is the token index in t flow, and j its index in t' flow:*

$$
\begin{array}{rll}
\forall\, p \in \mathcal{P},\ \forall\, i \in \mathbb{N}^*, & i \curvearrowright_p i + M(p) & \\
\forall\, n \in \mathcal{N}_\lambda \cup \mathcal{N}_c,\ \forall\, i \in \mathbb{N}^*, & i \curvearrowright_n i & \\
\forall\, s \in \mathcal{N}_s,\ \forall\, b \in \mathbb{B},\ \forall\, i \in \mathbb{N}^*, & \mathrm{idx}_b\left(R(s), i\right) \curvearrowright_s^{(b)} i & \\
\Leftrightarrow & i \curvearrowright_s^{(b)} \left|\mathrm{prf}\left(R(s), i\right)\right|_b \quad \wedge \quad \left(R(s)\right)_i = b & \\
\forall\, m \in \mathcal{N}_m,\ \forall\, b \in \mathbb{B},\ \forall\, i \in \mathbb{N}^*, & i \curvearrowright_m^{(b)} \mathrm{idx}_b\left(R(m), i\right) & \\
\Leftrightarrow & \left|\mathrm{prf}\left(R(m), i\right)\right|_b \curvearrowright_m^{(b)} i \quad \wedge \quad \left(R(m)\right)_i = b &
\end{array}
$$

where $b \in \mathbb{B}$ corresponds to the considered routing node input or output.

The first tokens to leave the place are the ones of the initial marking. Then by assumption the next tokens leave the place in the same order as they entered it, their rank shifted by the initial marking.

We now want to define, for every couple of places p, q, the set $\downarrow_q^p$, which is made of all couples (i, j) such that the token occurrence p_i at place p later becomes (possibly associated with others) the occurrence q_j. The definition is inductive, on the length of the longest path from p to q.

Definition 13 (Flow Relations over a Path)**.** *Let $\Sigma_{p,q}$ the set of all paths, in a KRG, from p output to q output. $\sigma \in \Sigma_{p,q}$ is of the form:*

$$\sigma = t_1 \underset{n}{\rightsquigarrow} t_2 \underset{p'}{\rightsquigarrow} \ldots \underset{n'}{\rightsquigarrow} t_{l-1} \underset{q}{\rightsquigarrow} t_l$$

with $n, ..., n' \in \mathcal{N}$ and $p, p', ..., q \in \mathcal{P}$. Relation $\curvearrowright_\sigma$ on path σ can be computed using elementary relation composition:

$$\curvearrowright_\sigma = \curvearrowright_q \circ \curvearrowright_{n'} \circ \ldots \circ \curvearrowright_{p'} \circ \curvearrowright_n$$

where $\curvearrowright_b \circ \curvearrowright_a = \{(i,k) \mid \exists\ (i,j) \in \curvearrowright_a,\ (j,k) \in \curvearrowright_b\}$.

Definition 14. *In a given KRG, let p and q be two places of $\mathcal{P}$, such that there exists a path from p to q. Let $\downarrow_q^p$ be the set of relations from p to q such that:*

$$\downarrow_q^p = \bigcup_{\sigma \in \Sigma_{p,q}} \{(i,j) \mid i \curvearrowright_\sigma j\}$$

In other words, we define:

$$\downarrow_p^p = \{(i,i) \mid i \in \mathbb{N}^*\}$$

and the set $\downarrow_q^p$ is recursively defined as follows:

- *if $p\bullet \in (\mathcal{N}_\lambda \cup \mathcal{N}_c)$, then:*

$$\downarrow_q^p = \bigcup_{r \in p\bullet\bullet} \left\{(i,k) \mid i \curvearrowright_r \circ \curvearrowright_{p\bullet} j,\ (j,k) \in \downarrow_q^r\right\}$$

- *si $p\bullet \in (\mathcal{N}_s \cup \mathcal{N}_m)$, alors :*

$$\downarrow_q^p = \bigcup_{r \in p\bullet\bullet} \left\{(i,k) \mid i \curvearrowright_r \circ \curvearrowright_{p\bullet}^{(b)} j,\ (j,k) \in \downarrow_q^r\right\}$$

where $b \in \mathbb{B}$ is p or r label, according to $p\bullet$ type.

Definition 15. *A KRG is said order preserving if and only if, for all $p, q \in \mathcal{P}$, $\downarrow_q^p$ is monotone.*

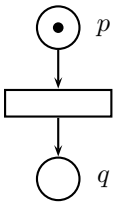
Note that in a subgraph consisting strictly of merge and select nodes, all relations $\downarrow_q^p$ are bijective: for all i, there exists a unique j such that $(i,j) \in \downarrow_q^p$, and vice versa.

4.2 Token Dependencies

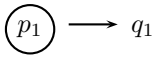
Let us consider a KRG $\langle \mathcal{N}, \mathcal{P}, \mathcal{T}, M, R \rangle$. Token p_i in place p may depend on token q_j in place q such that:

- a *flow dependence* is noted $p_i \rightarrow_f q_j$[1],
- a *sequence dependence* is noted $p_i \rightarrow_s q_j$.

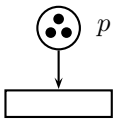
[1] We can possibly precise $p_i \overset{n}{\rightarrow}_f q_j$, in order to point out that the dependence relation is caused by the firing of n.



(a) Part of a KRG introducing a flow dependency.



(b) Flow dependency between p_i and q_j in (a).



(c) Part of a KRG introducing output dependencies.

(d) Output dependencies between tokens of p in (c).

Figure 2: Examples of different dependency types.

In the first case, a node n fires; it consummes token p_i and produces q_j; it is an operand-result relation. In the latter case, token sequentiality introduces a precedence relation: p_i has to be processed before q_j. Then, token dependences can be modeled as a *dependence graph*. Figure 2 presents the differences between these two kinds of dependences.

Because a KRG execution is possibly infinite, the number of tokens flowing through each place is possibly infinite as well. We first define the concept of *Expanded Dependence Graph* (EDG), that represents the exhaustive (thus possibly infinite) set of dependences.

Definition 16 (Expanded Dependence Graph)**.** *Let $\langle \mathcal{N}, \mathcal{P}, \mathcal{T}, M, R \rangle$ be a KRG. Its expanded dependence graph is a 4-tuple $\langle \mathcal{M}, \mathcal{D}, \mathcal{I}, \mathcal{Y} \rangle$ such that:*

- *$\mathcal{M}$ is a (possibly infinite) set of vertices, corresponding to the set of passings of tokens p_i in each place p.*
- *$\mathcal{D} \subseteq \mathcal{M} \times \mathcal{M}$ is a (possibly infinite) set of edges, or dependences. We note $\mathcal{D}_f$ the subset of flow dependences, and $\mathcal{D}_s$ the subset of sequence dependences, such that:*

$$\begin{aligned} \forall\, p, q \in \mathcal{P}, \quad & p\bullet = \bullet q,\ \forall\, p_i, q_j \in \mathcal{M},\ i \curvearrowright_q \circ \curvearrowright_{p\bullet} j \ \Leftrightarrow\ p_i \xrightarrow{p\bullet}_f q_j \\ \forall\, p_i, p_{i+1} \in \mathcal{M}, \quad & p_i \rightarrow_s p_{i+1} \end{aligned}$$

- *$\mathcal{I} \subseteq \mathcal{M}$ is the finite set of initial tokens:*

$$\mathcal{I} = \{\, p_i \mid p_i \in \mathcal{M},\ i \leq M\,(p)\}$$

- *$\mathcal{Y} \subseteq \mathcal{M}$ is the (possibly infinite) set of yieldable tokens; that is to say tokens that are not initially present, but that can be produced by graph inputs:*

$$\mathcal{Y} = \{\, p_i \mid p_i \in \mathcal{M},\ \bullet\bullet p = \emptyset\}$$

Remark 17. *For all $p_i \in \mathcal{M}$, if the indegree of p_i is null, then $p_i \in \mathcal{I} \cup \mathcal{Y}$.*

Remark 18. *An EDG is acyclic.*

Remark 19. *It is infinite if and only if its corresponding KRG is (quasi-)live.*

Definition 20 (Dependence Cone)**.** *A dependence cone in an expanded dependence graph $\langle \mathcal{M}, \mathcal{D}, \mathcal{I}, \mathcal{Y} \rangle$ is a subgraph $\langle \mathcal{M}_c, \mathcal{D}_c \rangle$:*

- *with an unique vertex of null outdegree (sink),*
- *where for all vertex, there exists a path from this vertex to the sink.*

Hence:

$$\forall\, p_i \in \mathcal{M}_c,\ \forall\, q_j \in \mathcal{M},\ \ (q_j \rightarrow p_i) \Rightarrow (q_j \in \mathcal{M}_c,\ q_j \rightarrow p_i)$$

4.3 Reducing Dependence Graphs

As mentionned above, EDG are possibly infinite graphs; they are useful to define the concept of dependence graph, but they are actually inappropriate for real-life studies and implementations. As a consequence, we want to define a *finite* form of a KRG dependence graph, so-called *Reduced Dependence Graph* (RDG).

To that aim, we first define an equivalence relation ($\overset{f}{\sim}$) that only consider flow dependences. The reader will notice that a parallel could be drawn between what follows, and Milner's work on equivalences and bisimulations in CCS [18, 19]; despite different models, both reasonings and goals are very similar.

Definition 21. *Let $\langle \mathcal{M}, \mathcal{D}, \mathcal{I}, \mathcal{Y} \rangle$ be an EDG, and $\mathcal{R} \in \mathcal{M} \times \mathcal{M}$ a relation over its vertices. For all $p_i, p'_{i'} \in \mathcal{M}$, if $p_i \mathcal{R} p'_{i'}$, then one of the following conditions is verified:*

1. *There exists a path σ in the EDG between p_i and $p'_{i'}$, such that all dependences in σ are flow dependences induced by routing or copy node firings, or*
2. *$p_i \xrightarrow{n}_f q_j \Leftrightarrow p'_{i'} \xrightarrow{n'}_f q'_{j'}$ and $\lambda(n) = \lambda(n')$ and $q_j \mathcal{R} q'_{j'}$*

Proposition 22. *Let $\mathcal{R}_n$ be a relation verifying definition 21. Following relations also verify definition 21:*

1. *$id_{\mathcal{M}}$*
2. *$\mathcal{R}_n^{-1}$*
3. *$\mathcal{R}_n \circ \mathcal{R}_{n'}$*
4. *$\bigcup_n \mathcal{R}_n$*

Let the relation $\overset{f}{\sim}$ be the union of relations $\mathcal{R}_n$ verifying definition 21. For all $p_i, p'_{i'} \in \mathcal{M}$, p_i and $p'_{i'}$ are said to be flow equivalent if and only if $p_i \overset{f}{\sim} p'_{i'}$.

Proposition 23. *$\overset{f}{\sim}$ is an equivalence relation.*

Proposition 24 (Flow Equivalence)**.** *Let $\langle \mathcal{M}, \mathcal{D}, \mathcal{I}, \mathcal{Y} \rangle$ be an EDG. For all $p_i, p'_{i'} \in \mathcal{M}$, p_i and $p'_{i'}$ are flow equivalent, noted $p_i \overset{f}{\sim} p'_{i'}$, if and only if one of the following conditions is verified:*

1. *There exists a path σ in the EDG from p_i to $p'_{i'}$, such that all dependences in σ are flow dependences induced by routing or copy node firings, or*

2. $p_i \xrightarrow{n}_f q_j \Leftrightarrow p'_{i'} \xrightarrow{n'}_f q'_{j'}$ *and* $\lambda(n) = \lambda(n')$ *and* $q_j \overset{f}{\sim} q'_{j'}$

Lemma 25. *Let $\langle \mathcal{N}, \mathcal{P}, \mathcal{T}, M, R \rangle$ be a bounded KRG with empty initial routing sequences. In other words:*

$$\forall\, n \in \mathcal{N}_s \cup \mathcal{N}_m,\ \exists\, v_n \in \mathbb{B}^*,\ R(n) = v_n^{\omega}$$

1. *For all $n \in \mathcal{N}_s \cup \mathcal{N}_m$, there exists $j_n \in \mathbb{N}^*$, such that for all $k \in \mathbb{N}$, $R(n) = \mathrm{suf}(R(n), kj_n)$*

2. *There exists a firing sequence in which each n is fired exactly j_n times.*

Theorem 26. *Let $\langle \mathcal{N}, \mathcal{P}, \mathcal{T}, M, R \rangle$ be a bounded KRG, and let $\langle \mathcal{M}, \mathcal{D}, \mathcal{I}, \mathcal{Y} \rangle$ be its infinite EDG. There exists $l, j \in \mathbb{N}$ $(j > 0)$ such that:*

$$\forall\, p \in \mathcal{P},\ \forall\, i, k,\ i < j,\ p_{l+i} \overset{f}{\sim} p_{l+i+kj}$$

Theorem 26 give us a necessary and sufficient condition for bounding dependency studies: graph behavior is eventually periodic, after an initial phase of length l. Then we can limit our study to the initial part and the first period.

However, relation $\overset{f}{\sim}$ do not consider sequence dependences. We would like to define another equivalence relation that preserve these informations, that may be required in order to detect deadlocks. Now we define another relation, co-called behavior equivalence, and noted $\overset{b}{\sim}$.

Definition 27 (Precedence)**.** *Let $\langle \mathcal{N}, \mathcal{P}, \mathcal{T}, M, R \rangle$ be a bounded KRG, and let $\langle \mathcal{M}, \mathcal{D}, \mathcal{I}, \mathcal{Y} \rangle$ be its EDG. For all $p_i, p'_{i'} \in \mathcal{M}$, p_i precedes $p'_{i'}$ if and only if $p_i = p'_{i'}$, or $\exists\, q_j, q_{j'},\ j < j'$:*

- $\exists\, n, \ldots, n' \in \mathcal{N}_c \cup \mathcal{N}_m \cup \mathcal{N}_s,\ p_i \xrightarrow{n}_f \circ \cdots \circ \xrightarrow{n'}_f q_j$
- $\exists\, n, \ldots, n' \in \mathcal{N}_c \cup \mathcal{N}_m \cup \mathcal{N}_s,\ p'_{i'} \xrightarrow{n}_f \circ \cdots \circ \xrightarrow{n'}_f q_{j'}$

Definition 28. *Let $\langle \mathcal{N}, \mathcal{P}, \mathcal{T}, M, R \rangle$ be a bounded KRG, and let $\langle \mathcal{M}, \mathcal{D}, \mathcal{I}, \mathcal{Y} \rangle$ be its EDG. We define the relation $\precsim$ as the transitive closure of the set of precedence relations, as defined in definition 27.*

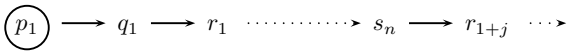
Proposition 29. *$\precsim$ is a preorder relation.*

Definition 30 (Behavioral Equivalence)**.** *Let $\langle \mathcal{N}, \mathcal{P}, \mathcal{T}, M, R \rangle$ be a bounded KRG, and let $\langle \mathcal{M}, \mathcal{D}, \mathcal{I}, \mathcal{Y} \rangle$ be its EDG. For all $p_i, p'_{i'} \in \mathcal{M}$, p_i and $p'_{i'}$ are behaviorally equivalent, noted $p_i \overset{b}{\sim} p'_{i'}$, if and only if:*

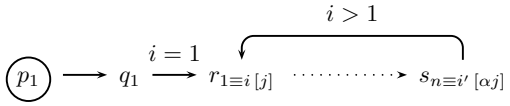
$$p_i \overset{f}{\sim} p'_{i'} \ \wedge\ p_i \precsim p'_{i'} \ \wedge\ p'_{i'} \precsim p_i$$

Proposition 31. *$\overset{b}{\sim}$ is an equivalence relation.*

Proposition 32. *For all $p_i, q_j \in \mathcal{M}$, if $p_i \overset{f}{\sim} q_j$, then $p_i \overset{b}{\sim} q_j$.*



(a) Part of an infinite EDG.



(b) Corresponding RDG: folding of the periodic part.

Figure 3: Relation between EDG and RDG.

We are now able to construct $\overset{f}{\sim}$- and $\overset{b}{\sim}$-RDGs from any EDG: equivalent nodes in an EDG are merged into a single node in the corresponding RDG.

Definition 33 (Reduced Dependence Graph)**.** *Let $\langle \mathcal{N}, \mathcal{P}, \mathcal{T}, M, R\rangle$ be a KRG, and let $\langle \mathcal{M}, \mathcal{D}, \mathcal{I}, \mathcal{Y}\rangle$ be its corresponding EDG. The related RDG $\langle \mathcal{M}', \mathcal{D}', \mathcal{I}', \mathcal{Y}'\rangle$ is an EDG, such that:*

- *$\mathcal{M}'$ and $\mathcal{D}'$ are both finite.*
- *there exist surjections from $\mathcal{M}$ to $\mathcal{M}'$ and from $\mathcal{D}$ to $\mathcal{D}'$ respectively. Indeed, for all $p_i, q_j \in \mathcal{M}$, and their corresponding nodes $p'_i, q'_j \in \mathcal{M}'$ respectively, $p'_{i'} = q'_{j'}$ if and only if $p_i \overset{f}{\sim} q_j$ (resp. $p_i \overset{b}{\sim} q_j$) for a $\overset{f}{\sim}$-RDG (resp. $\overset{b}{\sim}$-RDG).*
- *for all $p'_{i'} \in \mathcal{M}$,*
  - *if there exists $q'_{j'} \in \mathcal{M}'$, such that $\left(q'_{j'}, p'_{i'}\right) \in \mathcal{D}'$, and $q'_{j'}$ has an infinite number of inverse images in $\mathcal{M}$.*
  - *if there exists $r'_{k'} \in \mathcal{M}'$, such that $(r'_{k'}, p'_{i'}) \in \mathcal{D}'$, and $r'_{k'}$ has one and only one inverse image $r_k \in \mathcal{M}$.*

  then we assign validity flags to dependences. Let p_i be an inverse image of $p'_{i'}$, where $(r_k, p_i) \in \mathcal{D}$. Then $(r'_{k'}, p'_{i'})$ is valid if and only if $i' = i$, and $\left(q'_{j'}, p'_{i'}\right)$ is valid if and only if $i' > i$.
- *$\mathcal{I}'$ and $\mathcal{Y}'$ are the corresponding images of $\mathcal{I}$ and $\mathcal{Y}$ in the RDG.*

We also note $card\,(p'_{i'})$ the cardinal of inverse images of $p'_{i'}$ in $\mathcal{I}'$.

In other words, equivalent nodes of an EDG are merged into a single node in the corresponding RDG. An example is given in Figure 3. For instance, all token occurrences of the form r_{1+kj} are merged into a single node $r_{1\equiv i\,[j]}$. Therefore, this would add extra-dependencies to $r_{1\equiv i\,[j]}$, which would now depend on both q_1 and $s_{n\equiv i'\,[\alpha j]}$. For this reason, we add exclusive assertions to its entering edges: in the first case, we assert that the dependence is true if and only if $i = 1$; otherwise, the dependence is true if and only if $i > 1$.

4.4 First Normal Form: Graph Unfolding

Finally, we are able to back-transform a RDG into its *dual* form, as a quasi-MG. An acyclic MG produces tokens of the initial phase, while a cyclic MG represents the periodic part. Both MGs are linked by a layer of merge nodes: a merge node reads once on its "initial" input, then infinitely on its "periodic" input. Routing sequences are thus of the form $1.(0)^{\omega}$. This idea is depicted in Figure 4.

Rule 34 (Translating a RDG into a quasi-MG)**.** *Let $\langle \mathcal{M}, \mathcal{D}, \mathcal{I}, \mathcal{Y} \rangle$ be a RDG. Its corresponding quasi-MG $\langle \mathcal{N}, \mathcal{P}, \mathcal{T}, M \rangle$ is such that:*

- *There exists an injection from elements of $\mathcal{M}$ to $\mathcal{P}$: for all $q_j \in \mathcal{M}$, there exists one and only one corresponding place $p \in \mathcal{P}$, and vice versa. We note $place(q_j) = p$;*
- *For all $q_j \in \mathcal{M}$, $M(place(q_j)) = card(q_j)$;*
- *For all $q_j \in \mathcal{M}$, let $\mathcal{M}_{q_j}$ be its in-neighborhood, and let $\mathcal{P}_{q_j}$ be the corresponding subset of places ($\mathcal{P}_{q_j} \subset \mathcal{P}$ and $\forall q'_{j'} \in \mathcal{M}_{q_j}$, $place(q'_{j'}) \in \mathcal{P}_{q_j}$).*
  - *If in-dependencies, from any $q'_{j'}$ to q_j, do not have validity flags, then let l be the label of these in-dependencies. There exists a unique node $n \in \mathcal{N}$, with $\lambda(n) = l$ (possibly $\lambda(n) = \varepsilon$ in case of sequence dependence), such that $place(q_j) \in n\bullet$ and $\mathcal{P}_{q_j} = \bullet n$. See Figures 5 (a) and (b);*
  - *Otherwise, a layer of merge nodes are inserted in between, whose routing sequences are $1.(0)^{\omega}$. See Figures 5 (c) and (d);*
  - *Special case: if a token is consumed by computation nodes with different labels, then we insert an intermediary copy. See Figures 5 (e) and (f). We recall that copy nodes are special cases of computation nodes, hence they can be seen as general MG nodes.*
- *For all $n, n' \in \mathcal{N}_{\lambda}$, $n = n'$ if and only if $n\bullet = n'\bullet$, $\bullet n = \bullet n'$, and $\lambda(n) = \lambda(n')$.*
- *For all $q_j \in \mathcal{M}$, if the indegree of q_j is null, then there exists a node $n \in \mathcal{N}$ such that $place(q_j) \in n\bullet$:*
  - *If all inverse images of $place(q_j)$ are in $\mathcal{I}$, then there exists a place p in $\mathcal{P}$, with no inverse image in $\mathcal{M}$, such that $\bullet n = \{p\}$ and $n\bullet = \{place(q_j), p\}$;*
  - *Otherwise, $\bullet n = \emptyset$ and $n\bullet = \{p\}$.*

Remark 35. *A quasi-MG is a special case of KRG.*

Hence we have expressed KRG semantics according to the well-known and simpler MG semantics. This though process is similar to Lee's SDF expansion into MG [15]. Then, properties such as liveness can be proved either directly on KRGs or on their equivalent quasi-MG.

Theorem 36 (KRG Liveness)**.** *A KRG is quasi-live if and only if its $\overset{b}{\sim}$-quasi-MG is quasi-live. Likewise, a KRG is live if and only if its $\overset{b}{\sim}$-quasi-MG is live.*

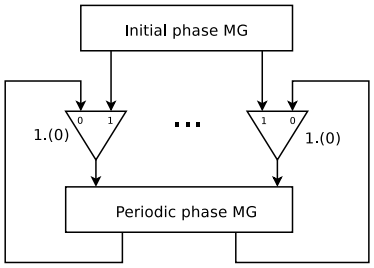


Figure 4: Shape of a quasi-MG.

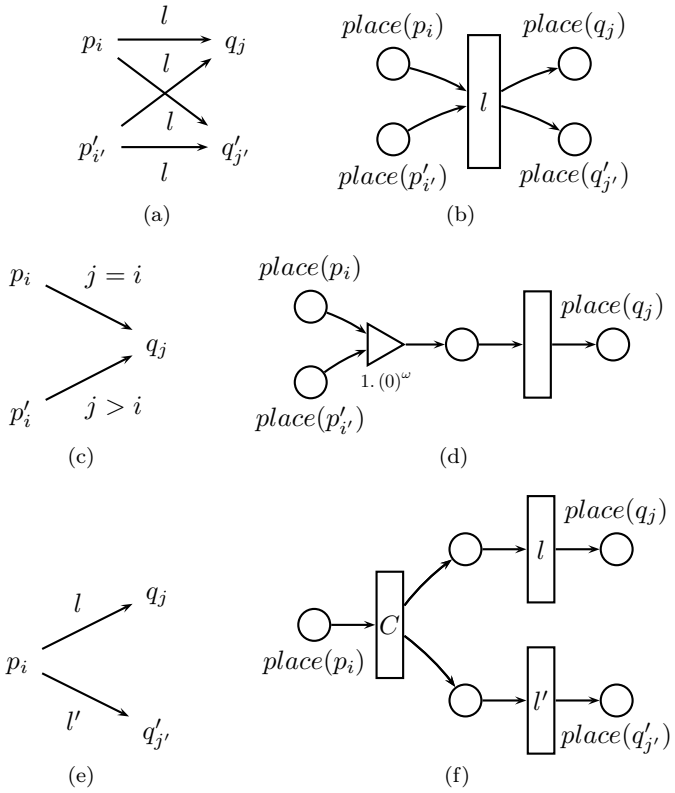
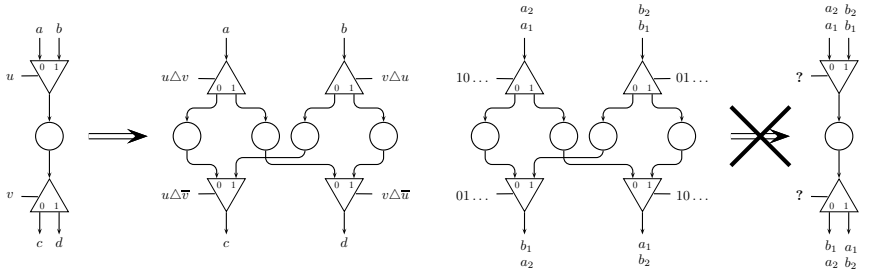


Figure 5: Constructing a quasi-MG *(right)* from a RDG *(left)*: constructing places and inserting computation nodes ((a) and (b)); linking initial and periodic parts with merge nodes ((c) and (d)); copying a token for multiple consumptions ((e) and (f)).

Building $\overset{f}{\sim}$- and $\overset{b}{\sim}$-quasi-MGs from a KRG gives us semantic reference points: all its equivalent graphs shall be expanded into the same MGs, depending on whether sequence dependencies shall be enforced or not.



(a) Place expansion *(PlaceExp)*.

(b) Example of graph where reverse factorization *(PlaceFact)* is impossible.

Figure 6: Expanding and factorizing places.

5 Graph Transformations

Optimizing interconnect traffic (sharing or unsharing places and buffers) is again an important issue in System-on-Chip design, where interconnects represent on-chip network structures.

We introduce local algebraic transformations on interconnect nodes, which we prove *sound* according to the previous behavior equivalence.

5.1 Local Graph Transformations

Places can be shared by different token flows: for instance if data are serialized through a shared communication medium. Expanding such place, as shown in Figure 6 (a), is equivalent to replace the shared medium by as many point-to-point links as required. This transformation may introduce more concurrency in the KRG but still preserves token order over each flow.

Lemma 37. $\forall\, u, v \in \mathbb{P}, \forall\, i \in \mathbb{N}^*$,

$$v_{[u]_i} = 1 \quad \Rightarrow \quad |\mathrm{prf}\,(v, \mathrm{idx}_1\,(u, i))|_1 = \mathrm{idx}_1\,(u \triangle v, |\mathrm{prf}\,(v \triangle u, i)|_1)$$

Proposition 38 (Expanding a Place)**.** *Expanding a place (PlaceExp) preserves order relations over token flows.*

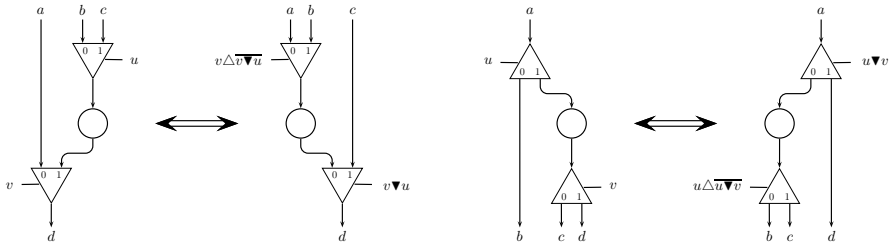
Notice that factorizing places is not always possible, as illustrated in Figure 6 (b). This is because some token orders, allowed by point-to-point connections are incompatible with token sequentialization that generates a total order.

Proposition 39 (Permuting Merge Nodes)**.** *Permuting merge nodes (MergePerm) preserves order relations over token flows.*

Proposition 40 (Permuting Select Nodes)**.** *Permuting select nodes (SelectPerm) preserves order relations over token flows.*

Corollary 41 (Transformation Correctness)**.** *Expanding a place, and permuting select and merge nodes alter neither graph boundedness, nor liveness, nor deadlock-freeness properties.*

Using previous local properties, we can generalize such local transformations to trees, and directed acyclic graphs. At each local transformation, we can check its correctness through a check of the equivalence of the expansion of the original and the expansion of the transformed KRG.



(a) Permuting merge nodes *(MergePerm)*. (b) Permuting select nodes *(SelectPerm)*.

Figure 7: Permuting routing nodes.

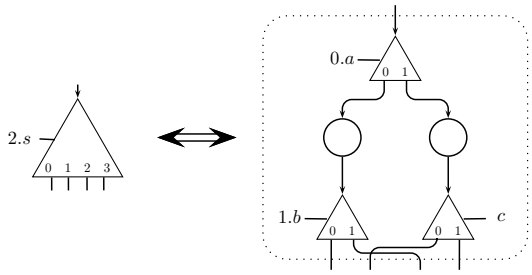


Figure 8: Example of 4-select block, and routing of the first token towards output #2.

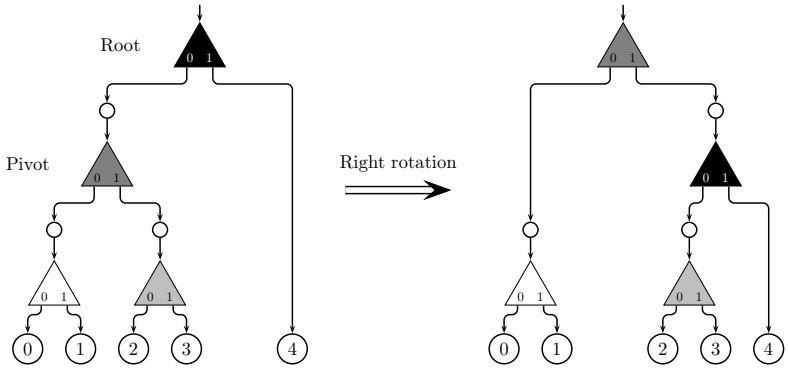


Figure 9: Balancing a 5-select tree.

5.2 Select and Merge Trees

Definition 42 (n-Select Block)**.** *A n-select block is a tree of $n-1$ select nodes; it has exactly one input and n outputs, labelled from 0 to $n-1$. By analogy with select nodes, its routing sequence is an ultimately periodic sequence over $[\![0, n-1]\!]$.*

Theorem 43 (Adelson-Velskii and Landis)**.** *The height h of an AVL tree with n inner nodes is bounded as follows:*

$$\log_2(n+1) \leq h < \log_\phi(n+2) + \log_\phi\left(\sqrt{5}\right) - 2$$

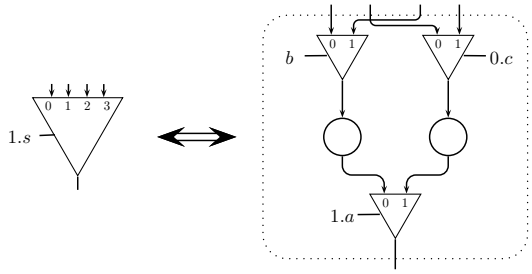
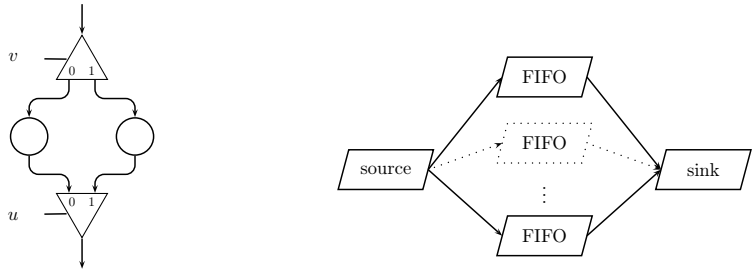


Figure 10: Example of 4-merge block, and routing of the first token from input #1.



(a) Elementary sorter.

(b) Parallel FIFO network.

Figure 11: Exemple of patterns allowing token permutations.

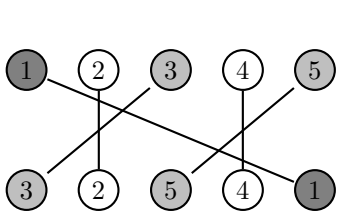
Definition 44 (n-Merge Block)**.** *A n-merge block is a tree of $n-1$ merge nodes; it has exactly n inputs, labelled from 0 to $n-1$, and one output. By analogy with merge nodes, its routing sequence is an ultimately periodic sequence over $[\![0, n-1]\!]$.*

5.3 Permuting Tokens

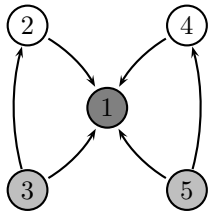
Up to this point, we have only considered cases where tokens are consumed in the same order as they are produced. Now we encompass the case of non-monotonic relations $\downarrow_q^p$. Token permutations are seldom discussed in PN literature; however they are of prime importance while considering token flow interleavings. We characterize permutations, and compute how much resources are strictly necessary for realizing a given permutation.

Tokens flowing through a single path are totally ordered. Bypasses are only possible if tokens follow different routes. We can define the concept of *sorter*: a subgraph consisting in a tree of select nodes, linked by the leafs to its symmetric tree of merge nodes; token permutations depend on both select tree dispatching, and reading priorities assigned to merge nodes. A first solution, based on *permutation graphs*, is proposed in Golumbic [11].

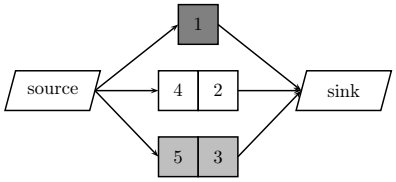
Definition 45 (Permutation Graph)**.** *Let $\pi = [\pi_1, \pi_2, ..., \pi_n]$ be a permutation of the n-tuple $[1, 2, ..., n]$. We note π_i^{-1} the position in the a permutation of the*



(a) Matching diagram.



(b) Permutation graph $G[\pi]$.



(c) Permutating tokens through a parallel FIFO network.

Figure 12: Example of permutation $\pi = [3, 2, 5, 4, 1]$.

i^{th} element of the given n-tuple. The permutation graph $G[\pi] = \langle V, E \rangle$ is a digraph[2] *as follows:*

- *Each vertex corresponds to an element of the permutation:*

$$V \stackrel{def}{=} \{1, 2, ..., n\}$$

- *There exists an edge from a vertex i to a vertex j if and only if their corresponding are mutually permuted:*

$$E \stackrel{def}{=} \left\{(i, j) \mid i, j \in V,\ i > j,\ \pi_i^{-1} < \pi_j^{-1}\right\}$$

  In other words, the i^{th} token enters the sorter after the j^{th} and exists before.

Therefore, the indegree $d^-(v)$ of a vertex v equals the number of times that the corresponding token is bypassed other tokens. Conversely, the outdegree $d^+(v)$ equals the number of times that that corresponding token bypasses other tokens. Notice that such a graph is acyclic.

Proposition 46. *Let π be a permutation. The following numbers are equal:*

- *the chromatic number χ of $G[\pi]$,*
- *the minimum number of queues required to sort π,*
- *the length of a longest decreasing subsequence of π.*

[2] In references, permutation graphs are defined as undirected graphs such that $E = \left\{(i, j) \mid (i - j)\left(\pi_i^{-1} - \pi_j^{-1}\right) < 0\right\}$.

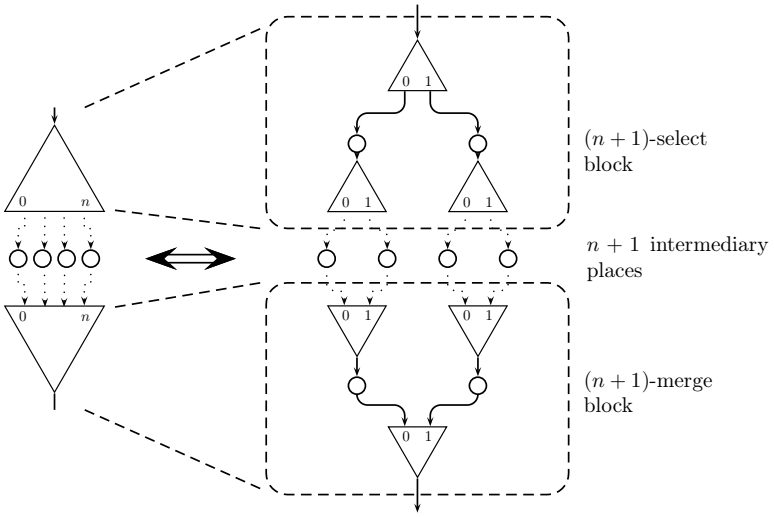


Figure 13: Example of parallel sorter.

Let us consider the example of permutation $\pi = [3, 2, 5, 4, 1]$ in Figure 12 (a). Its corresponding permutation graph is shown in Figure 12 (b). Such token permutation can be realised using the sorter of Figure 12 (c); three paths are necessary and sufficent, since $G[\pi]$ is 3-chromatic.

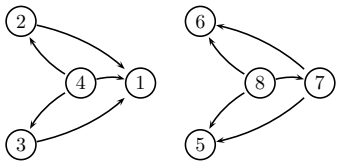
Definition 47 (Parallel Sorter)**.** *We call parallel sorter a part of a KRG dedicated to token permutations. Its nodes are either select or merge nodes, and they are linked as follows:*

- *select nodes form a n-select block,*
- *merge nodes form a n-merge block,*
- *the i-labelled output of the n-select block is linked to the i-labelled input of the n-merge block.*

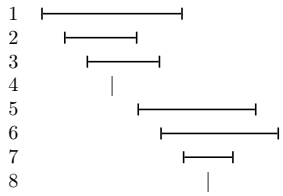
However, first-fit coloring algorithms are not suited for computing minimum place bounds. Indeed, this problem is similar to register allocation [10]; it requires to consider path assignments as well as token lifetimes inside the sorter. The second aspect can be modeled using an *interference graph* [12], which is an interval graph whose intervals are bounded by memory grants and releases. An example is shown in Figure 14.

Definition 48 (Interval Graph)**.** *Let $\mathcal{I}_1, \mathcal{I}_2, ..., \mathcal{I}_n \subset \mathcal{R}$ be a set of intervals over the ordered set $\mathcal{R}$. The corresponding interval graph $g = \langle V, E \rangle$ is an unoriented graph; it has one vertex for each interval in the set, and an edge between each pair of vertices corresponding to intervals that intersect.*

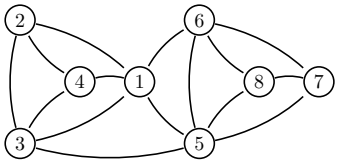
$$\begin{aligned} V &\stackrel{def}{=} \{I_1, I_2, ..., I_n\} \\ E &\stackrel{def}{=} \{I_i \cap I_j \neq \emptyset \mid \forall i, j \in [\![1..n]\!], i \neq j\} \end{aligned}$$



(a) Permutation graph.



(b) Permutation interferences.



(c) Interval graph.

Figure 14: Example of permutation $\pi = [4, 2, 3, 1, 8, 7, 5, 6]$.

Let s be the schedule of incoming tokens. Then the i^{th} token enters the sorter at time $in\,(i) = \text{idx}_1\,(s, i)$, and we can recursively compute its exit time (according to an ASAP firing rule) such that:

$$out\,(i) = \max\left(\text{idx}_1\,(s, i)\,, out\left(\pi_{\left(\pi_i^{-1}-1\right)}\right) + 1\right)$$

Now we formulate the integer quadratic program for computing minimal place bounds in the sorter. Let $G_P\,[\pi] = (V_P, E_P)$ and $G_I\,[\pi] = (V_I, E_I)$ be respectively the permutation and interference graphs of a given permutation π. $G_P\,[\pi]$ is X-chromatic. We set parameters as follows:

- $C_i = 1$ if $\alpha_i = \delta_i$, 0 otherwise.
- $I_{i,j} = 1$ if $(i, j) \in E_I$, 0 otherwise.
- $P_{i,j} = 1$ if $(i, j) \in E_P$, 0 otherwise.

Then we define the unknowns of our system:

- $\chi_{i,k} = 1$ if node i has color k, 0 otherwise.
- c_k is the capacity of k^{th} FIFO, corresponding to color k.

Proposition 49 (Minimum Capacity of a Parallel Sorter)**.** *The minimum capacity of the parallel sorter of n tokens using X paths is given by the following integer quadratic program:*

$$\text{minimize } \textit{capacity} = \sum_{k=1}^{X} c_k$$

with the following constraints:

$$\begin{array}{cc}
\chi_{i,k} \in \{0,1\}\,, \quad c_k \in \mathbb{N} & \forall\, i \in [\![1..n]\!], \forall\, k \in [\![1..X]\!] \\
\sum_{k=1}^{X} \chi_{i,k} = 1 & \forall\, i \in [\![1..n]\!] \\
(\chi_{i,k} + \chi_{j,k})\, P_{i,j} \leq 1 & \forall\, i, j \in [\![1..n]\!], \forall\, k \in [\![1..X]\!] \\
\left(\left(\sum_{j=1}^{i-1} \chi_{j,k} I_{j,i}\right) + 1 - C_i\right) \chi_{i,k} \leq c_k & \forall\, i \in [\![1..n]\!], \forall\, k \in [\![1..X]\!]
\end{array}$$

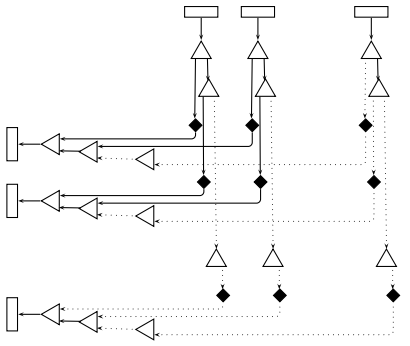


Figure 15: Shape of a normal form. Computation nodes are depicted both on the left side and at the top. Links show the normalized interconnect, with merge and select trees. Sorters are drawn in black.

Corollary 50 (Parallel Sorter Routings)**.** *Let $R(s)$ et $R(m)$ be respectively the routing sequences of the X-select block s and X-merge block m. They are computed using Proposition 49 such that:*

$$R(s) = [\chi_1, \chi_2, ...\chi_n]$$
$$R(f) = [\chi_{\pi_1}, \chi_{\pi_2}, ...\chi_{\pi_n}]$$

5.4 Second Normal Form: Interconnect Expansion

To sum up, we are now able to both expand interconnects and reorder tokens between a producer and a consumer. Hence we can normalize any KRG routing via a sequence of elementary transformations.

For simplicity, we first consider the case without token copies; the normal form shape is depicted in Figure 15. It consists mostly in connecting each computation node output to a fan-out of select nodes, in a comb shape. This structure splits the token flow according to the next computation node destination. Symmetrically, each computation node input is connected to a similar fan-in comb shape, gathering the flows from all potential computation node source. In the middle, links are thus separated as in a crossbar, with a separate channel for each couple producer/consumer. There, sorters take care of potential out-of-order token flows.

Remark 51. *The pattern shown in Figure 11 (a) is equivalent to a simple edge if and only if $u = v$. The same property holds if paths are crossed and if $u = \overline{v}$.*

Remark 52. *In Figure 11 (a), replacing u by $\overline{u}$ and v by $\overline{v}$ respectively preserve the KRG behavior.*

Remark 53. *Between a select and a merge node, paths can be uncrossed by replacing u by $\overline{u}$ or v by $\overline{v}$ exclusively.*

Theorem 54. *There exists a KRG normal form expansion, for which* SelectPerm*,* MergePerm*, and* PlaceExp *transformation rules are complete.*

Concerning token copies, the normal form is globally the same. However, select nodes in Figure 15 are replaced by more complex structures involving

copy nodes. They may produce token copies on both outputs, and they may even produce multiple copies towards a consumer. However, the corresponding transformation requires a factorization rule *PlaceFact* (cf. Figure 6 (b)). We conjecture that such *PlaceFact* transformations preserve behavioral equivalence in that case.

Conjecture 55. *There exists a KRG normal form expansion, for which* SelectPerm*,* MergePerm*,* PlaceExp *and* PlaceFact *transformation rules are complete.*

6 Conclusion

We introduced K-periodically Routed Graphs, a model which essentially retains the simplicity of Marked Graphs and Boolean Data Flow graphs. The explicit, finitely presented condition patterns for switching nodes allow to consider the topological modifications on the switching interconnect network, which preserve the computational semantics. These practical goals are achieved at the expense of a number of theoretical developments, to define exactly what "preservation" means here. This included the definition of proper algebraic graph transformations, and canonical normal forms amongst equivalent KRG forms. Thes results are currently being implemented into our K-PASSA tool [4] for analysis of Process Networks.

Further works should include the combinaison of (k-periodic) routing with former k-periodic scheduling on the one hand, and a better connection of these routing patterns to the kind of control achievable as nested loop forms of code, already used extensively for automatic parallelization. Extensions of KRG with parameters in the sense of HPDF [20] could also be an interesting direction.

A Proofs

A.1 Proofs of Section 3

Proof of Proposition 9. Property 1 of [6]. □

A.2 Proofs of Section 4

Proof of Proposition 22.

1. There cannot exist a self-dependence in an EDG, hence point 1 is false with respect to $id_{\mathcal{M}}$. However, point 2 is verified: it is obvious that $p_i \xrightarrow{n}_f q_j \Leftrightarrow p_i \xrightarrow{n}_f q_j$ and $\lambda(n) = \lambda(n)$ and $q_j \mathcal{R}_n q_j$. Hence $id_{\mathcal{M}}$ verifies the definition.

2. Point 1 is obviously true. Also, equality and equivalence are symmetric, such that:

$$p'_{i'} \mathcal{R}_n p_i \quad \Rightarrow \quad p'_{i'} \xrightarrow{n'}_f q'_{j'} \Leftrightarrow p_i \xrightarrow{n}_f q_j \ \wedge \ \lambda(n') = \lambda(n) \ \wedge \ q'_{j'} \mathcal{R}_n q_j$$

   hence point 2 is true and $\mathcal{R}_n^{-1}$ verifies the definition.

3. If there exists paths between p_i and $p'_{i'}$, and between $p'_{i'}$ and $p''_{i''}$, then there exists a path between p_i and $p''_{i''}$, hence point 1 is true. Now, let us suppose that $p_i \mathcal{R}_m p'_{i'}$ and $p'_{i'} \mathcal{R}_{m'} p''_{i''}$. Then:

$$p_i \xrightarrow{n}_f q_j \Leftrightarrow p'_{i'} \xrightarrow{n'}_f q'_{j'} \ \wedge \ \lambda(n) = \lambda(n') \ \wedge \ q_j \mathcal{R}_m q'_{j'}$$
$$p'_{i'} \xrightarrow{n'}_f q'_{j'} \Leftrightarrow p''_{i''} \xrightarrow{n''}_f q''_{j''} \ \wedge \ \lambda(n') = \lambda(n'') \ \wedge \ q'_{j'} \mathcal{R}_{m'} q''_{j''}$$

   Therefore:

$$p_i \xrightarrow{n}_f q_j \Leftrightarrow p''_{i''} \xrightarrow{n''}_f q''_{j''} \ \wedge \ \lambda(n) = \lambda(n'') \ \wedge \ q_j \mathcal{R}_{m''} \circ \mathcal{R}_m q''_{j''}$$

4. It is obvious that if each $\mathcal{R}_n$ individually verifies the definition, then their union also verifies it.

□

Proof of Proposition 23.
Reflexivity: For all $p_i \in \mathcal{M}$, $p_i \overset{f}{\sim} p_i$, by Proposition 22 (1).
Symmetry: For all $p_i, p_j \in \mathcal{M}$, if $p_i \overset{f}{\sim} p_j$, then $p_j \overset{f}{\sim} p_i$, by Proposition 22 (2).
Transitivity: For all $p_i, p'_{i'}, p''_{i''} \in \mathcal{M}$, if $p_i \overset{f}{\sim} p'_{i'}$ and $p'_{i'} \overset{f}{\sim} p''_{i''}$, then $p_i \overset{f}{\sim} p''_{i''}$, by Proposition 22 (3). □

Proof of Proposition 24. By definition 21, we know that half of the proposition holds. We now want to prove the reverse implication. For instance, we can define a relation $\mathcal{R}$ such that $p_i \mathcal{R} p'_{i'}$ if and only if:

$$p_i \xrightarrow{n}_f q_j \Leftrightarrow p'_{i'} \xrightarrow{n'}_f q'_{j'} \ \wedge \ \lambda(n) = \lambda(n') \ \wedge \ q_j \overset{f}{\sim} q'_{j'}$$

Hence:

$$p_i \overset{f}{\sim} p'_{i'} \ \Rightarrow \ p_i \mathcal{R} p'_{i'} \tag{2}$$

Let $p_i \xrightarrow{n}_f q_j$. By the definition of $\mathcal{R}$, we can find $q'_{j'}$, such that $p'_{i'} \xrightarrow{n'}_f q'_{j'}$ and $\lambda(n) = \lambda(n')$ and $q_j \overset{f}{\sim} q'_{j'}$. Then by (2), $q_j \mathcal{R} q'_{j'}$. □

Proof of Lemma 25. This lemma is a direct consequence of abstracting a KRG into a SDF graph (by Rule 8 and Proposition 9). When solving balance equations, we compute, for all routing node $n \in \mathcal{N}_m \cup \mathcal{N}_s$, the number of periods i_n that n shall be fired, so that the whole period of the SDF graph is balanced. Moreover, n is executed $|R(n)|$ times over its period (as a KRG node). Let $j_n = i_n \times |R(n)|$ be the number of times n shall be fired over a balanced KRG period.

By hypothesis, $R(n)$ is k-periodic of period $|R(n)|$. Then it is obvious that $R(n)$ without its $i_n \times |R(n)|$ first letters equals itself. Hence $R(n) = \mathrm{suf}(R(n), j_n)$. From then on, equation $R(n) = \mathrm{suf}(R(n), kj_n)$ is proved by induction over k. □

Proof of Theorem 26. By definition 5, each routing nodes has an initial phase. Let us call l_n the length of the initial phase of routing node n. If each routing node n can be fired at least l_n times, then the whole graph enters its steady routing. Otherwise, if any node n cannot be fired l_n times, then all paths passing through n are dead paths, and can be ignored if we consider the behavior of the graph after a huge number of firings. Hence there exists $l > l_n$, for all n, such that if each node is fired at least l times, then it enters its periodic phase.

Now we consider the behavior of the graph in its periodic phase: there is either none or an infinite number of tokens flowing through each place. Using Lemma 25, we determine the length of a period, and we can bound our study to a single period of length j. According to Proposition 24, $p_{l+i} \overset{f}{\sim} p_{l+i+kj}$ if and only if:

$$p_{l+i} \to q_x \Leftrightarrow p_{l+i+kj} \to q_y \quad \text{and} \quad q_x \overset{f}{\sim} q_y \tag{3}$$

Then, according to Definition 16, there are two kinds of dependencies: flow dependencies and sequence dependencies. We have to prove (3) in both cases.

Sequence dependency: $\forall p_i, p_{i+1} \in \mathcal{M},\ p_i \to_s p_{i+1}$. Let $q = p$, $x = l + i + 1$ and $y = x + kj$. We show by induction that the equivalence is true for all j and k.

Flow dependency: $\forall p, q \in \mathcal{P},\ p\bullet = \bullet q,\ i \curvearrowright_q \circ \curvearrowright_{p\bullet} j \ \Leftrightarrow\ p_i \xrightarrow{p\bullet}_f q_j$. Then, according to $p\bullet$:

- if $p\bullet \in \mathcal{N}_\lambda \cup \mathcal{N}_c$, then for all i, $l + i \curvearrowright_q \circ \curvearrowright_{p\bullet} l + i + M(q)$, and $p_{l+i} \to_f q_{l+i+M(q)}$. Let $x = l + i + M(q)$ and $y = x + kj$, and we prove by induction that the equivalence is true for all j and k.
- if $p\bullet \in \mathcal{N}_m$, then for all i:

  $$l + i \curvearrowright_q \circ \curvearrowright^{(b)}_{p\bullet} |\mathrm{prf}(R(p\bullet), l + i)|_b + M(q)$$

  where $b \in \mathbb{B}$ depends on which input of $p\bullet$ is linked to p. Let $i = k' |R(p\bullet)| + i'$ and $i' < |R(p\bullet)|$. We can split $R(p\bullet)$ so that:

  $$|\mathrm{prf}(R(p\bullet), l + i)|_b = |\mathrm{prf}(R(p\bullet), l)|_b + k' |(p\bullet)|_b + |w|_b$$

where w is the suffix of $\mathrm{prf}\left(R\left(p\bullet\right), l+i\right)$ of length i'. Let:

$$\begin{aligned} x &= \left|\mathrm{prf}\left(R\left(p\bullet\right), l\right)\right|_b + k'\left|(p\bullet)\right|_b + \left|w\right|_b + M\left(q\right) \\ y &= x + \frac{\left|R\left(p\bullet\right)\right|_b}{\left|R\left(p\bullet\right)\right|} kj \end{aligned}$$

We show by induction that the equivalence is verified for all k and for all j of the form $k''\left|R\left(p\bullet\right)\right|$.

- In a similar way, if $p\bullet \in \mathcal{N}_s$, then for all i:

$$l+i \curvearrowright_q \circ \curvearrowright_{p\bullet}^{(b)} \mathrm{idx}_b\left(R\left(p\bullet\right), l+i\right)$$

Let $i = k'\left|R\left(p\bullet\right)\right|_1 + i'$ and $i' < \left|R\left(p\bullet\right)\right|_1$. We can split the expression so that:

$$\mathrm{idx}_b\left(R\left(p\bullet\right), l+i\right) = \mathrm{idx}_b\left(R\left(p\bullet\right), l\right) + k'\left|R\left(p\bullet\right)\right| + y$$

where y is the length of the corresponding shortest suffix of $R\left(p\bullet\right)$ containing the last i'^{th} b's. Then we pose:

$$\begin{aligned} x &= \mathrm{idx}_b\left(R\left(p\bullet\right), l\right) + k'\left|R\left(p\bullet\right)\right| + y + M\left(q\right) \\ y &= x + \frac{\left|R\left(p\bullet\right)\right|}{\left|R\left(p\bullet\right)\right|_b} kj \end{aligned}$$

and we show by induction that the equivalence is verified for all k and for all j of the form $k''\left|R\left(p\bullet\right)\right|_b$.

As a consequence, let:

$$j = \prod_{n \in \mathcal{N}_s \cup \mathcal{N}_m} \max\left(1, \left|R\left(n\right)\right| \times \left|R\left(n\right)\right|_0 \times \left|R\left(n\right)\right|_1\right) \tag{4}$$

Then j is a multiple of all $\left|R\left(n\right)\right|$, $\left|R\left(n\right)\right|_0$ and $\left|R\left(n\right)\right|_1$; j verifies previous equivalence requirements. j is also a multiple of the graph period (least common multiple of node periods). When j tokens have been consummed in place p, αj tokens have been produced into place q, where α is either 1, $\frac{|R(p\bullet)|_b}{|R(p\bullet)|}$ or $\frac{|R(p\bullet)|}{|R(p\bullet)|_b}$, depending on $p\bullet$. Routings are periodic, and because we have proved the equivalence for all p, now we can prove by induction that $p_{l+i} \overset{f}{\sim} p_{l+i+kj} \Leftrightarrow q_x \overset{f}{\sim} q_y$ holds. □

Proof of Proposition 29.
Reflexivity: $p_i = p_i$, hence $p_i \precsim p_i$
Transitivity: By definition, $\precsim$ is a transitive closure, hence it is transitive. □

Proof of Proposition 31.
Reflexivity: $p_i \overset{f}{\sim} p_i$ and $p_i \precsim p_i$, hence $p_i \overset{b}{\sim} p_i$, by Propositions 23 and 29.
Symmetry: $\overset{f}{\sim}$ is symmetric by Proposition 23, and $p_i \precsim q_j \wedge q_j \precsim p_i$ is symmetric too. Hence $p_i \overset{b}{\sim} q_j \Leftrightarrow q_j \overset{b}{\sim} p_i$.
Transitivity: $\overset{f}{\sim}$ and $\precsim$ are both transitive, hence $\overset{b}{\sim}$ is transitive too. □

Proof of Proposition 32. This proposition is directly induced by Definition 30. □

Proof of Theorem 36. By definition, a KRG or a quasi-MG is quasi-live if there exists a node that can fire infinitely often; morever, it is live if any node can fire infinitely often. Conversely one can say, for instance, that a KRG or a quasi-MG is quasi-live if there exists a node that will not eventually starve. These property can be transposed to EDGs and RDGs: for instance, liveness in an RDG means that all its tokens can be computed.

However, $\overset{f}{\sim}$-RDGs preserve informations about enabling and firing rules, but they do not strictly preserve informations on token ordering. Thus, given a KRG, we only consider in this theorem its $\overset{b}{\sim}$-RDG and $\overset{b}{\sim}$-quasi-MG.

As stated in Remark 35, a quasi-MG is actually a special case of KRG. Given a $\overset{b}{\sim}$-RDG, we can build its corresponding $\overset{b}{\sim}$-quasi-MG, then we can abstract it back as a $\overset{b}{\sim}$-RDG: the resulting RDG is equivalent to the initial one, where some sequence dependences have been turned into flow dependences.

Then, given a KRG and its equivalent $\overset{b}{\sim}$-quasi-MG, their $\overset{b}{\sim}$-RDGs are topologically equivalent. We can thus conclude that when the one suffer starvation, the other also does. □

A.3 Proofs of Section 5

Proof of Lemma 37. We illustrate the proof by the example of Figure 6 (a). Left-hand side of the implication asserts that the $\mathrm{idx}_1(u,i)^{th}$ token going through the merge node is routed onto the 1^{st} output of the select node, *i.e.* the i^{th} token in flow b is routed towards flow d. We show that both operands of the equality are different manners to write the index in flow d of this i^{th} token of flow b.

On the left-hand side: $\mathrm{idx}_1(u,i)$ is the position, in the merge output flow, of the i^{th} token in b (position of the i^{th} "1" in the routing sequence). $\mathrm{prf}(v, \mathrm{idx}_1(u,i))$ corresponds to the routing by select node of tokens, up to the i^{th} from b. We have assumed that the i^{th} letter of the select sequence is "1", so its position in d equals the number of "1" in this prefix.

On the right-hand side: $v \triangle u$ is a sampling by the routing sequence of the select node by the one of the merge node; it corresponds to the routing, by the select node, of all tokens issued from b. Then, $|\mathrm{prf}(v \triangle u, i)|_1$ is the number of those tokens, up to the i^{th} one, routed to d. Conversely, $u \triangle v$ corresponds to origins of tokens in d. Finally, $\mathrm{idx}_1(u \triangle v, |\mathrm{prf}(v \triangle u, i)|_1)$ is the position, among tokens in d, of the i^{th} coming from b. □

Proof of Proposition 38. We infer the following relations between input and output flows, using and composing flow relations:

Figure 6 (a) *(Left)*:

$$a \curvearrowright_s^{(0)} \circ \curvearrowright_p \circ \curvearrowright_m^{(0)} |\mathrm{prf}(\overline{v}, \mathrm{idx}_1(\overline{u}, a))|_1$$
$$a \curvearrowright_s^{(1)} \circ \curvearrowright_p \circ \curvearrowright_m^{(0)} |\mathrm{prf}(v, \mathrm{idx}_1(\overline{u}, a))|_1$$
$$b \curvearrowright_s^{(0)} \circ \curvearrowright_p \circ \curvearrowright_m^{(1)} |\mathrm{prf}(\overline{v}, \mathrm{idx}_1(u, b))|_1$$
$$b \curvearrowright_s^{(1)} \circ \curvearrowright_p \circ \curvearrowright_m^{(1)} |\mathrm{prf}(v, \mathrm{idx}_1(u, b))|_1$$

Figure 6 (a) *(Right)*:

$$\begin{aligned}
a \curvearrowright_m^{(0)} \circ \curvearrowright_p \circ \curvearrowright_s^{(0)} &\ \mathrm{idx}_1\left(\overline{u}\triangle\overline{v}, |\mathrm{prf}\left(\overline{v}\triangle\overline{u}, a\right)|_1\right)\\
a \curvearrowright_m^{(0)} \circ \curvearrowright_p \circ \curvearrowright_s^{(1)} &\ \mathrm{idx}_1\left(\overline{u}\triangle v, |\mathrm{prf}\left(v\triangle\overline{u}, a\right)|_1\right)\\
b \curvearrowright_m^{(1)} \circ \curvearrowright_p \circ \curvearrowright_s^{(0)} &\ \mathrm{idx}_1\left(u\triangle\overline{v}, |\mathrm{prf}\left(\overline{v}\triangle u, b\right)|_1\right)\\
b \curvearrowright_m^{(1)} \circ \curvearrowright_p \circ \curvearrowright_s^{(1)} &\ \mathrm{idx}_1\left(u\triangle v, |\mathrm{prf}\left(v\triangle u, b\right)|_1\right)
\end{aligned}$$

Then, equalities between left and right relations are shown by direct application of Lemma 37. □

Proof of Proposition 39.

Figure 7 (a) *(Left)*:

$$\begin{aligned}
a &\curvearrowright_m^{(0)} \mathrm{idx}_1\left(\overline{v}, a\right)\\
b \curvearrowright_m^{(1)} \circ \curvearrowright_p \circ \curvearrowright_m^{(0)} &\ \mathrm{idx}_1\left(v, \mathrm{idx}_1\left(\overline{u}, b\right)\right)\\
c \curvearrowright_m^{(1)} \circ \curvearrowright_p \circ \curvearrowright_m^{(1)} &\ \mathrm{idx}_1\left(v, \mathrm{idx}_1\left(u, c\right)\right)
\end{aligned}$$

Figure 7 (a) *(Right)*:

$$\begin{aligned}
a \curvearrowright_m^{(0)} \circ \curvearrowright_p \circ \curvearrowright_m^{(0)} &\ \mathrm{idx}_1\left(\overline{v\blacktriangledown u}, \mathrm{idx}_1\left(\overline{v\triangle\overline{v\blacktriangledown u}}, a\right)\right)\\
b \curvearrowright_m^{(0)} \circ \curvearrowright_p \circ \curvearrowright_m^{(1)} &\ \mathrm{idx}_1\left(\overline{v\blacktriangledown u}, \mathrm{idx}_1\left(v\triangle\overline{v\blacktriangledown u}, b\right)\right)\\
c &\curvearrowright_m^{(1)} \mathrm{idx}_1\left(v\blacktriangledown u, c\right)
\end{aligned}$$

Then, we have:

(a)

$$\begin{aligned}
\overline{v\blacktriangledown u}\blacktriangledown\overline{v\triangle\overline{v\blacktriangledown u}} &= (v\blacktriangledown u) \oplus \overline{\overline{v\blacktriangledown u}\blacktriangledown\left(v\triangle\overline{v\blacktriangledown u}\right)}\\
&= (v\blacktriangledown u) \oplus \overline{v \wedge \overline{v\blacktriangledown u}}\\
&= (v\blacktriangledown u) \oplus (\overline{v} \vee (v\blacktriangledown u))\\
&= \overline{v}
\end{aligned}$$

(b)

$$\begin{aligned}
\overline{v\blacktriangledown u}\blacktriangledown\left(v\triangle\overline{v\blacktriangledown u}\right) &= v \wedge \overline{v\blacktriangledown u}\\
&= v \wedge (\overline{v} \oplus (v\blacktriangledown\overline{u}))\\
&= v\blacktriangledown\overline{u}
\end{aligned}$$

(c)

$$v\blacktriangledown u = v\blacktriangledown u$$

□

Proof of Proposition 40.

Figure 7 (b) *(Left)*:

$$\begin{aligned}
\mathrm{idx}_1\left(\overline{u}, b\right) &\curvearrowright_s^{(0)} b\\
\mathrm{idx}_1\left(u, \mathrm{idx}_1\left(\overline{v}, c\right)\right) &\curvearrowright_s^{(0)} \circ \curvearrowright_p \circ \curvearrowright_s^{(1)} c\\
\mathrm{idx}_1\left(u, \mathrm{idx}_1\left(v, d\right)\right) &\curvearrowright_s^{(1)} \circ \curvearrowright_p \circ \curvearrowright_s^{(1)} d
\end{aligned}$$

Figure 7 (b) *(Right)*:

$$\begin{aligned}\mathrm{idx}_1\left(\overline{u\blacktriangledown v}, \mathrm{idx}_1\left(\overline{u\triangle\overline{u\blacktriangledown v}}, b\right)\right) &\curvearrowright_s^{(0)} \circ \curvearrowright_p \circ \curvearrowright_s^{(0)} b\\ \mathrm{idx}_1\left(\overline{u\blacktriangledown v}, \mathrm{idx}_1\left(u\triangle\overline{u\blacktriangledown v}, c\right)\right) &\curvearrowright_s^{(1)} \circ \curvearrowright_p \circ \curvearrowright_s^{(0)} c\\ \mathrm{idx}_1\left(u\blacktriangledown v, d\right) &\curvearrowright_s^{(1)} d\end{aligned}$$

Then, the proof is similar to the one of Proposition 39. □

Proof of Theorem 43. Given in [1] and [14]. □

Proof of Proposition 46. Corollary 7.4 of [11]. □

Proof of Proposition 49. We want to minimize the total capacity of the sorter, that is to say, the sum of place capacities. Here we detail the meaning of each constraint:

- $\chi_{i,k} \in \{0,1\}$ and $c_k \in \mathbb{N}$, as previously defined.
- $\sum_{k=1}^{X} \chi_{i,k} = 1$ asserts that each node i has one and only one color k, hence a token is assigned to exactly one intermediary place.
- $(\chi_{i,k} + \chi_{j,k})\, P_{i,j} \leq 1$ asserts that if i and j are adjacent, then at most one is k-colored; hence two permuted tokens do not goes through the same path.

Up to this point, these constraints equal Grund-Hack's or Appel-George's ones. The last constraint define the lower bound of capacity c_k, according to node colors (token routing).

$\sum_{j=1}^{i-1} \chi_{j,k} I_{j,i}$ gives the number of tokens stored in place k when the i^{th} arrives ($j < i$). We add 1 if this token does not exit the sorter immediately and has to be stored too ($+1 - C_i$). Then we multipley the whole parenthesis by $\chi_{i,k}$, only considering the case where i is k-colored (zero otherwise). Finally, we get the number of token in place k at a given instant, including token i. □

Proof of Theorem 54. The proof goes by induction on the number of computation nodes. Let us consider two computation nodes n and n', such that there exists a path σ from n to n' going through places, select and merge nodes only (neither computation nor copy nodes).

First, we can apply transformation PlaceExp over σ as long as there are merge nodes *above* select nodes. Thus, select nodes are pushed upward, and merge nodes are pushed downward. Then, interconnect is transformed so that any path σ goes through a n-select tree, *then* a n-merge tree.

The next step consists in reordering n-select and n-merge trees. We apply either MergePerm, or SelectPerm, or modifying routing with respect to Remarks 52 and 53; we iterate till we get a parallel sorter.

This parallel sorter may be not minimal, as in Section 5.3. Establishing the sorter scheme requires knowledge of the target, in order to decide on the required sequence of graph transformations. Thus, we can split token flows with respect to Remark 51, in order to separate tokens assigned to different channels; then we reorder select and merge nodes using transformations MergePerm and SelectPerm; finally we merge them back, because tokens assigned to a common channel are not permuted (merge routing equals select routing). □

Contents

References

[1] Georgy M. Adelson-Velskii and Yevgeniy M. Landis. An algorithm for the organization of information. *Proceedings of the USSR Academy of Sciences*, 146:263–266, 1962. English translation by Myron J. Ricci in Soviet Math. 3, pp. 1259–1263.

[2] Shuvra S. Battacharyya, Edward A. Lee, and Praveen K. Murthy. *Software Synthesis from Dataflow Graphs*. Kluwer Academic Publishers, Norwell, MA, USA, 1996.

[3] Albert Benveniste, Paul Caspi, Stephen A. Edwards, Nicolas Halbwachs, Paul Le Guernic, and Robert de Simone. The synchronous languages 12 years later. *Proceedings of the IEEE*, 91(1):64–83, 2003.

[4] Julien Boucaron, Anthony Coadou, and Robert de Simone. KPASSA: A tool for simulating periodically scheduled SoCs. Presentation at DATE Conference, April 2009. http://www-sop.inria.fr/aoste/?r=9&s=26.

[5] Julien Boucaron, Anthony Coadou, and Robert de Simone. *Synthesis of Embedded Software - Frameworks and Methodologies for Correctness by Construction Software Design*, chapter Formal Modelling of Embedded Systems with Explicit Schedules and Routes. Springer, 2010. To appear.

[6] Julien Boucaron, Anthony Coadou, Benoît Ferrero, Jean-Vivien Millo, and Robert de Simone. Kahn-extended event graphs. Research Report RR-6541, INRIA, May 2008.

[7] Joseph T. Buck. *Scheduling Dynamic Dataflow Graphs with Bounded Memory Using the Token Flow Model*. PhD thesis, University of California, Berkeley, CA, USA, 1993.

[8] Albert Cohen, Marc Duranton, Christine Eisenbeis, Claire Pagetti, Florence Plateau, and Marc Pouzet. N-synchronous kahn networks: a relaxed model of synchrony for real-time systems. In *Conference Record of the 33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'06)*, pages 180–193, New York, NY, USA, January 2006. ACM.

[9] Frederic Commoner, Anatol W. Holt, Shimon Even, and Amir Pnueli. Marked directed graph. *Journal of Computer and System Sciences*, 5:511–523, October 1971.

[10] Janet Fabri. Automatic storage optimization. In *Proceedings of the 1979 SIGPLAN Symposium on Compiler Construction*, pages 83–91. ACM, 1979.

[11] Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Number 57 in Annals of Discrete Mathematics. Elsevier, 2004.

[12] Daniel Grund and Sebastian Hack. A fast cutting-plane algorithm for optimal coalescing. In Shriram Krishnamurthi and Martin Odersky, editors, *Compiler Construction 2007*, volume 4420 of *Lecture Notes In Computer Science*, pages 111 – 125. Springer, 2007.

[13] Gilles Kahn. The semantics of a simple language for parallel programming. In Jack L. Rosenfeld, editor, *Information Processing '74: Proceedings of the IFIP Congress*, pages 471–475, New York, NY, USA, 1974. North-Holland.

[14] Donald E. Knuth. *The Art of Computer Programming*, volume 3 (Sorting and Searching). Addison-Wesley Publishing Co., 2nd edition, 1969.

[15] Edward A. Lee. *A Coupled Hardware and Software Architecture for Programmable Digital Signal Processors*. PhD thesis, University of California, Berkeley, June 1986.

[16] Edward A. Lee and David G. Messerschmitt. Static scheduling of synchronous data flow programs for digital signal processing. *IEEE Transactions on Computers*, C-36(1):24–35, 1987.

[17] Edward A. Lee and David G. Messerschmitt. Synchronous data flow. *Proceeding of the IEEE*, 75(9):1235–1245, 1987.

[18] Robin Milner. *A Calculus of Communicating Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982.

[19] Robin Milner. *Communication and Concurrency*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[20] Mainak Sen, Ivan Corretjer, Fiorella Haim, Sankalita Saha, Jason Schlessman, Tiehan Lv, Shuvra S. Bhattacharyya, and Wayne Wolf. Dataflow-based mapping of computer vision algorithms onto fpgas. *EURASIP Journal on Embedded Systems*, 2007(1):29–29, 2007.

Centre de recherche INRIA Sophia Antipolis – Méditerranée
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399