Dynamic Reconfigurations in Frequency Constrained Data Flow

Paul Dubrulle[1,(0000−0002−1158−6348)], Christophe Gaston[1,(0000−0001−6865−5108)], Nikolai Kosmatov[1,2,(0000−0003−1557−2813)], and Arnault Lapitre[1,(0000−0002−2185−4051)]

[1]CEA, List, 91191 Gif-sur-Yvette France
firstname.lastname@cea.fr
[2]Thales Research and Technology, Palaiseau, France

Abstract. In Cyber-Physical Systems, the software components are often distributed over several computing nodes, connected by a communication network. Depending on several factors, the behavior of these components may dynamically change during its execution. The existing data flow formalisms for the performance prediction of dynamic systems do not cover the real-time constraints of these systems, and suffer from complexity issues in the verification of mandatory model properties. To overcome these limitations, we propose a dynamic extension to Polygraph, a static data flow formalism covering the real-time behavior of the CPS components. We also propose a verification algorithm to determine if the transitions between different modes are well-defined for a given model. Initial experiments show that this algorithm can be efficiently applied in practice.

1 Introduction

Context. Cyber-Physical Systems (CPS) are increasingly present in everyday life. These systems are often distributed over several computing nodes, connected by a communication network. For example, the next generation of autonomous vehicles will heavily rely on sensor fusion systems to operate the car. Sensors and actuators are distributed over the car, in places where their measure or action makes sense, while fusion kernels operate on high-performance computation platforms. A network connects these elements together, and in some cases the computation kernels can even be off-loaded to remote servers over wireless connections.

Depending on several factors, the behavior of the software components of a CPS may change during its execution. The algorithms used to process data may change depending on the nature of the input data, and a component may even be deactivated due to an external factor. In an autonomous car for example, the components implementing a parking assistance functionality relying on a rear-view camera may operate at a lower resolution while driving on a highway, or even be deactivated completely.

When network communication is involved, an analysis of the communications between the components is necessary to determine the bandwidth and memory necessary to respect the application's real-time requirements. Dynamic variations

in execution time and bandwidth usage due to changes in the behavior of the components must be taken into account in the performance prediction.

Several extensions to static data flow formalisms [9, 2] can be used to perform this kind of performance analysis, taking dynamic reconfigurations into account [6, 7, 11, 14, 15]. In general for data flow formalisms, a prerequisite to analyze a model is the existence of a periodic schedule in bounded memory without deadlock, sometimes called an admissible schedule.

Motivation and Goals. Dynamic data flow models are adapted to capture reconfigurations in a CPS, but they lack expressiveness regarding the real-time synchronization of the components interfaced with the physical world. Recent research introduced Polygraph, a new static data flow formalism that covers such synchronous behavior [5], while allowing asynchronous behavior for computation kernels. Our goal is to extend Polygraph to support the expression of dynamic reconfigurations, describing more precisely the behavior of distributed CPS, thus allowing a refinement of the static performance analysis of the resulting models.

Approach and Main Results. This paper proposes to extend Polygraph models with the specification of different operational modes, in a way inspired by the well-known Scenario-Aware Data Flow (SADF) [14]. We rely on additional type information on the communication data to dynamically change the execution mode of the components that receive it, allowing for a distributed control of the current operational modes in the system. With our approach, if there is an admissible schedule for a polygraph model without any mode extension, this schedule is admissible for any of its extended versions. For a given model, this property allows to specify as many modes as required by the real-life system, without impacting the cost of the verification of the existence of such a schedule. In addition, we propose an algorithm to check that the dynamic changes in the modes of the components never lead to incoherent states where a component is supposed to execute in two different modes.

The contributions of this work include:

- an extension to the Polygraph data flow formalism, to support dynamic changes between static configurations of the modeled system;
- a proof of additional properties on the executions of non-extended polygraphs, and properties on their extended executions; these properties are proved in all generality, for an arbitrary reconfiguration strategy;
- an algorithm that, given an extended polygraph with an admissible schedule, checks that the current mode of the polygraph is always defined;
- an implementation of that algorithm in the DIVERSITY tool, and initial experiments to validate this approach.

Outline. The remainder of this paper is organized as follows. Section 2 gives an informal introduction to the proposed modeling approach. In Section 3, we remind the formalization of Polygraph, prove additional execution properties, and formalize the extension. Section 4 presents the verification algorithm and an initial evaluation of its implementation. In Section 5, we discuss related work, while Section 6 presents conclusion and perspectives.

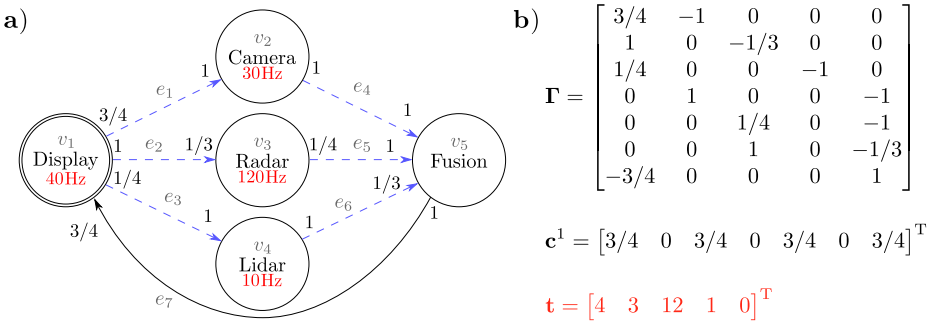


Fig. 1. a) A cockpit display system modeled as a polygraph denoted $\mathcal{P}_1$, and b) its topology matrix $\mathbf{\Gamma}$, initial channel state $\mathbf{c}^1$, and vector of synchronous constraints $\mathbf{t}$.

2 Motivation

To introduce the dynamic extension to Polygraph, we use a toy example of a data fusion system that could be integrated into the cockpit display of a car, depicted in Fig. 1. The system is composed of three sensors producing data samples to be used by a data fusion component, and a display component. The first sensor component is a video camera producing frames. The other two sensor components analyze radar and lidar based samples to produce a descriptor of the closest detected obstacles. The fusion component uses this information to draw the obstacle descriptors on the corresponding frame. The display component is a touch screen, and the driver can choose to activate or deactivate the rendering of the enhanced camera feed through an interface on the screen. The driver can also choose to deactivate only the lidar. This leads to three different configurations for the fusion sub-system: all the components are active (denoted *all*), all the components are active except the lidar (denoted *no lidar*), and the three sensor components are inactive (denoted *none*).

Existing semantic. The Polygraph language is a data flow formalism enabling static performance analysis of systems mixing real-time and compute intensive components. Compared to existing static data flow formalisms [9, 2, 10], Polygraph mixes synchronous and asynchronous constraints for the execution of a model, and is well-suited to capture the real-life constraints of the CPS.

The core element of Polygraph is a *system graph*, capturing data dependencies between the components. Each vertex of this graph models an *actor*, an abstract entity representing the function of a component. Each directed edge of the graph models a communication *channel*, the source actor being the producer of data consumed by the destination actor. The communication policy on the channels is First-In First-Out (FIFO), the write operation is non-blocking, and the read operation is blocking. The actors communicate by *firing*, an atomic process during which they consume and produce a certain number of data *tokens* on the connected channels. The number of tokens produced or consumed per firing of an actor on a channel is specified by a rational *rate*, which is adapted to capture resampling of data streams. Fig. 1a gives an example of a system graph with rational rates (shown near the ends of the edges) for our example.

Considering only the modeling elements mentioned so far, the model of Fig. 1a is equivalent to a Fractional Rate Data Flow graph [10]. In addition to these elements, Polygraph introduces a specific semantic for rational rates, whose goal is to approximate a non-linear behavior with integer rates by a linear behavior with rational rates, and thus to gain the good properties of a linear behavior. It allows rational initial conditions on the channels. The detailed semantic is recalled in Section 3.1.

The main advantage of Polygraph is the capability to label a subset of actors with *frequencies*, corresponding to the real-life constraints imposed for example by the sampling rates of the sensors (see the frequency labels on v_1, v_2 v_3, and v_4 in Fig. 1a). The actors that do not have a specified frequency correspond to computation kernels, which in real-life systems often compute as soon as enough input data is available (notice the absence of a frequency label on v_5 in Fig. 1a). A *global clock* provides *ticks* to synchronize the firings of frequency labeled actors, introducing global synchronous behavior in the data flow graph.

A prerequisite to analyze the performance of a Polygraph model is the existence of a periodic schedule with two properties. The first property, *consistency*, requires that the sizes of communication buffers remain bounded for an unbounded execution of the periodic schedule. In practice, if a model is not consistent, it is not possible to implement the communications without losing data samples. The second property, *liveness*, requires the absence of deadlocks in the schedule. The semantic of Polygraph is detailed in [5], including a proof that the existence of an *admissible schedule* with both consistency and liveness properties is decidable and can be checked in practice.

Dynamic extension. The different configurations of our fusion example could be captured by the scenario concept introduced in Scenario Aware Data Flow (SADF) [14], which encompasses the semantic of many other dynamic data flow formalisms (see Sec. 5). In SADF, some actors receive the role of detector, and are in charge of broadcasting the current scenario to other actors. Each actor impacted by the decisions of a detector has a different rate specification and execution time per possible scenario. Before firing, such an actor reads the scenario information received from the detector and then fires with the appropriate rates and execution time. Special care is required from the designer when specifying the alternative rates, to avoid situations where an actor consumes tokens produced in two different scenarios, that we call an *indecision*. An SADF model with no indecision is said to be *strongly consistent* (see [14] for more detail).

In our context, there are several issues that do not allow for a direct reuse of this concept. First of all, unlike Polygraph, SADF does not capture synchronous constraints for a subset of actors. Second, since the rates are the main parameter influencing the existence of an admissible schedule, a separate check is required for each possible combination of scenarios, which does not scale up well for real-life systems with many configurations. Finally, the broadcasting of the detected scenario has several disadvantages. Additional channels impede the model's readability, and increase the modeling complexity, since the designer needs to know the number of firings of each scenario-dependent actor in a schedule to spec-

ify appropriate output rates on the control channels. In addition, the idea of a centralized orchestrator implies additional synchronizations, with a negative impact on the overall performance for distributed architectures. Our proposal extends Polygraph with similar concepts, preserving the support of synchronous constraints, and trying to overcome these limitations.

Modes. We introduce the notion of *modes* for a polygraph. A mode is decided for each firing of an actor, and it is similar to a scenario. The main difference is that the rates do not change in different modes. Instead, a mode change implies a change in the contents of produced data tokens. For example, for the three aforementioned configurations in Fig. 1, the firing of each actor can have one of three user-defined modes λ_1 (*all*), λ_2 (*no lidar*), λ_3 (*none*), which can be used to enable a non-trivial behavior. For one firing of the lidar actor, it will produce 1 token regardless of the decided mode of that firing. The produced token will hold an obstacle descriptor if the firing has mode λ_1. For the other modes λ_2 and λ_3, since the lidar is disabled in the corresponding configurations, the produced token will hold an empty descriptor. With scenarios, a similar behavior would require rate 1 in scenario *all* and 0 in the other scenarios.

The fact that the rates are fixed, regardless of the mode of a firing, brings an important benefit. The existence of an admissible schedule for an extended polygraph can be verified once and for all, by considering the polygraph without modes. We show that by construction of the extension in Section 3.3.

Modes allow for a rich set of options: when the firing of an actor can have two or more different modes, configuration parameters (produced data type, execution time, placement, code version, etc.) may change depending on the mode, enabling static analysis of the modeled system with many different objectives. Our goal here being to provide a formal presentation of the modeling language, its mechanism for changing modes dynamically in an execution, and its properties, we do not detail how such parameters are associated to modes.

Mode propagation. Like in SADF, in our proposal, some actors, called *selectors*, are in charge of identifying reconfigurations and notifying the impacted actors, called *followers*. But unlike SADF, we do not require that selectors broadcast the selection information to all their followers. Instead, we define the propagation of modes transitively from a selector to its followers. In Fig. 1a, the double-circled display actor is a good candidate to be the selector responsible for the three user-defined modes λ_1, λ_2, and λ_3, since it reads the user input that will decide of the configuration. It does not require a specific channel connected to the fusion actor, the mode information will reach it through the channels represented with dashed arrows.

The advantage of this approach is that we do not need channels between a selector and its followers which are not its direct successors. This overcomes the issues caused by these channels. The propagation by transitivity also provides means to automatically check the absence of indecision in a model.

3 The Polygraph Modeling Language

We denote by $\mathbb{Z}$ the set of integers, by $\mathbb{N} = \{n \in \mathbb{Z} \,|\, n \geqslant 0\}$ the set of natural integers, and by $\mathbb{Q}$ the set of rational numbers. A number $r \in \mathbb{Q}$ rounded down (resp., up) to a closest integer is denoted by $\lfloor r \rfloor$ (resp., $\lceil r \rceil$), and the fractional part of r is denoted $\lfloor r \rceil = r - \lfloor r \rfloor$.

For a set A, we denote by A^* the set of all finite *words* over A (i.e. sequences of elements of A), and by A^+ the set of non-empty words.[1] The length of a word $w = a^1 \cdots a^n \in A^*$ is denoted $|w| = n$, and the i^{th} element of w is denoted $w[i] = a_i$. For any $a \in A$ and for any $n \in \mathbb{N}$ we denote $n * a$ the word composed of n occurrences of a. For any word $w \in A^*$ and $0 \leqslant l \leqslant |w|$, the suffix of w of length l is denoted $\text{suffix}(w, l)$. For any two words w and w', the concatenation of w and w' is denoted by $w \cdot w'$ or ww'.

3.1 Background

A *system graph* is a connected finite directed graph $G = (V, E)$ with set of vertices (or *actors*) V and set of edges (or *channels*) $E \subseteq V \times V$. We consider that V and E are indexed respectively by $\{1, \cdots, |V|\}$ and $\{1, \cdots, |E|\}$, and denote by v_j the actor of index j and by e_i the channel of index i. For an actor v_j, let $\text{in}(v_j) = \{\langle v_k, v_j \rangle \in E \,|\, v_k \in V\}$ denote the set of *input channels* of v_j; $\text{out}(v_j) = \{\langle v_j, v_k \rangle \in E \,|\, v_k \in V\}$ the set of *output channels* of v_j.

For any pair of a channel e_i and an actor v_j, we associate a *rate* γ_{ij} which is a rational number whose absolute value defines the partial production or consumption effect on e_i of each firing of v_j, and whose sign indicates if the effect is a partial production ($\gamma_{ij} > 0$) or consumption ($\gamma_{ij} < 0$). By convention, the rate γ_{ij} must be 0 if v_j is not connected to e_i, or connected to both ends of e_i. Indeed, for a *self-loop* $e_i = \langle v_j, v_j \rangle$ connecting v_j to itself, the global production/consumption effect of v_j on the channel must be 0 for the model to be consistent. Therefore the associated production and consumption rates must be equal. Their exact value does not matter and can be any integer. The rates are given by a matrix with one row per channel and one column per actor, as illustrated in Fig. 1b for $\mathcal{P}_1$.

Definition 1 (Topology matrix). *A matrix $\mathbf{\Gamma} = (\gamma_{ij}) \in \mathbb{Q}^{|E| \times |V|}$ is a* topology matrix *of a system graph G if for every channel $e_i = \langle v_k, v_l \rangle \in E$, we have:*

- *the rate $\gamma_{ij} = 0$ for all $j \neq k, l$;*
- *if $k \neq l$, then the rates $\gamma_{ik} > 0$ and $\gamma_{il} < 0$ are irreducible fractions, and at least one of them has a denominator equal to 1 (i.e. is an integer); let $q_i \geqslant 1$ be the greatest of their denominators , we define $r_i = 1/q_i$ the* smallest fraction portable by e_i*;*
- *if $k = l$, then $\gamma_{ik} = 0/1 = 0$, and we define $q_i = r_i = 1$.*

A *channel state* is a vector of rational numbers giving for each channel its state, tracking the partial production or consumption effect of the successive firings, which must thus be a multiple of its smallest portable fraction (c.f. Fig. 1b). The number of tokens in a channel is defined as the integer part of its rational state, and a token is actually produced (resp. consumed) by a firing when this integer part increases (resp. decreases) at this firing.

[1] In other words, A^* is the *free monoid* on A, and A^+ is the *free semigroup* on A.

Definition 2 (Channel state). *A vector* $\mathbf{c} = (c_i) \in \mathbb{Q}^{|E|\times 1}$ *is a* channel state *of a system graph* G *with topology matrix* $\mathbf{\Gamma}$ *if for every channel* $e_i = \langle v_j, v_k \rangle \in E$*, we have* $c_i = zr_i$ *for some* $z \in \mathbb{Z}$*. We say that* $\lfloor c_i \rfloor$ *is the* number of tokens occupying channel e_i*.*

A *polygraph* is composed[2] of a system graph, a topology matrix, and a subset of *timed actors* $V_F \subseteq V$ with certain *synchronous constraints* Θ. These constraints require that each timed actor fires at a given *frequency*, synchronously with respect to the ticks of a *global clock*. It is possible to choose a suitable time unit and a global clock with a suitable frequency, such that each $v_j \in V_F$ has to fire the same number of times $t_j \in \mathbb{N}$ during this time unit. In $\mathcal{P}_1$, with a time unit of 100ms and a global clock at 120Hz, the vector $\mathbf{t} = (t_j)$ gives that value t_j for each $v_j \in V_F$. The current tick of the global clock and the information about the timed actors which have already fired at this tick are represented[3] by a synchronous state θ. The detailed semantic of these synchronous constraints [5] is not essential for the comprehension of the extension we propose, as further discussed in Remark 5. For lack of space, we only recall basic notation and definitions that are mandatory to present the contributions of this paper.

Definition 3 (Polygraph, state). *A* polygraph *is a tuple* $\mathcal{P} = \langle G, \mathbf{\Gamma}, \Theta \rangle$ *containing a system graph* G*, a topology matrix* $\mathbf{\Gamma}$*, and* synchronous constraints Θ*. A* state *of a polygraph* $\mathcal{P}$ *is a tuple* $s = \langle \mathbf{c}, \theta \rangle$ *containing a channel state* $\mathbf{c}$*, and a* synchronous state θ*. We denote by* S *the set of all possible states of* $\mathcal{P}$*.*

The only possible transitions from one state to another are the firing of an actor or a tick of the global clock. Starting from an *initial state*, a sequence of states resulting from such successive transitions is called an *execution*.

Definition 4 (Fire, Tick). *For a polygraph* $\mathcal{P}$*, the mapping* $\text{fire} : V \times S \longrightarrow S$ *maps an actor* v_j *and a state* $s = \langle \mathbf{c}, \theta \rangle$ *to the state* $s' = \langle \mathbf{c}', \theta' \rangle$ *such that for each* $e_i \in E$ *we have* $c'_i = c_i + \gamma_{ij}$*, and* θ' *is the resulting synchronous state. The mapping* $\text{tick} : S \longrightarrow S$ *maps a state* $s = \langle \mathbf{c}, \theta \rangle$ *to the state* $s' = \langle \mathbf{c}', \theta' \rangle$ *such that we have* $\mathbf{c}' = \mathbf{c}$*, and* θ' *is the resulting synchronous state (see [5] for detail).*

Remark 1. Let $\delta_i^+(s) \in \mathbb{N}$ denote the amount of tokens produced on a channel $e_i = \langle v_j, v_k \rangle$ by a firing of v_j in state $s = \langle \mathbf{c}, \theta \rangle$. By Def. 2 and 4, we have $\delta_i^+(s) = \lfloor c_i + \gamma_{ij} \rfloor - \lfloor c_i \rfloor$. Similarly, for the number of tokens consumed on e_i by a firing of v_k in state s, denoted $\delta_i^-(s) \in \mathbb{N}$, we have $\delta_i^-(s) = \lfloor c_i \rfloor - \lfloor c_i + \gamma_{ik} \rfloor$.

Not all transitions from a state $s = \langle \mathbf{c}, \theta \rangle$ to a state $s' = \langle \mathbf{c}', \theta' \rangle$ are valid. First, the synchronous constraints impose an order on some firing and tick transitions. We write $\theta \vdash \theta'$ when these synchronous constraints (formally defined in [5]) are satisfied for the transition from s to s'. In addition, the policy to read from a channel e_i is read-blocking. Thus, the transition from s to s' is called *valid*, and denoted $s \vdash s'$, if $\theta \vdash \theta'$ and $\forall e_i \in E, c_i \geqslant 0 \wedge c'_i \geqslant 0$.

[2] Θ corresponds to ω and φ in [5, Def. 4], while initial marking $\mathbf{m}$ is not integrated into the polygraph definition in this paper.

[3] θ corresponds to τ and $\mathbf{a}$ in [5, Def. 5].

Definition 5 (Execution). *An* execution *of a polygraph $\mathcal{P}$ is a sequence of states $\sigma = s^1 \cdots s^n \in S^+$, such that s^1 is the* initial state *of σ, and for each $1 \leqslant l < n$, we have either $s^{l+1} = \mathrm{fire}(v_j, s^l)$ for some $v_j \in V$, or $s^{l+1} = \mathrm{tick}(s^l)$. An execution σ is said to be* valid *if $s^l \vdash s^{l+1}$ for all $1 \leqslant l < n$.*

Remark 2. The number of firings of actors in an execution $\sigma = s^1 \cdots s^n$ can be represented by a *tracking vector* $\mathbf{x}^\sigma = (x_j^\sigma) \in \mathbb{N}^{|V|\times 1}$, such that for each v_j, the component x_j^σ gives the number of l such that $1 \leqslant l \leqslant n$ and $s^{l+1} = \mathrm{fire}(v_j, s^l)$. We say that the f^{th} such firing is of *rank* f.

To perform a static performance analysis of a polygraph, there should be a valid periodic behavior of the system. In other words, only the valid executions $\sigma = s^1 \cdots s^n \in S^+$ returning to their initial state $s^1 = s^n$ are relevant. From [5, Th. 1,2], the existence of such executions can be decided in general. In Fig. 2, a small example polygraph $\mathcal{P}_2$ and an execution σ_2 are represented.

Example 1. Fig.2a presents a polygraph $\mathcal{P}_2$ with 3 actors and 3 channels. For simplicity, it has no synchronous constraints. Fig.2b illustrates, step-by-step, the states of an execution σ_2 with consecutive firings of v_3, v_2, v_1, v_2. The first five columns give the firing actor, the number of its firings so far, and the states of the channels. The latter show the states c_i of channels e_i and illustrate by circles the tokens occupying each channel. The last four columns will be explained later. The first row provides the initial state. For instance, the first firing of v_2 consumes one token from e_2 (since its state changes from $2/2 = 1$ to $1/2$) and produces one token on e_3. Note that σ_2 is valid and returns to its initial state after the first 4 steps, and can thus be repeated infinitely.

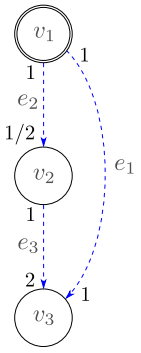
Definition 6 (Live execution). *For a polygraph $\mathcal{P}$, a valid execution $\sigma = s^1 \cdots s^n \in S^+$ is called* live *if $s^1 = s^n$. In this case, polygraph $\mathcal{P}$ is said to be* live from s^1.

Remark 3. We say that two valid executions $\sigma, \sigma' \in S^+$ of a polygraph $\mathcal{P}$ are equivalent, denoted by $\sigma \simeq \sigma'$, if their tracking vectors and initial states are equal, that is, $\mathbf{x}^\sigma = \mathbf{x}^{\sigma'}$ and $\sigma[1] = \sigma'[1]$, and the number of ticks is the same. If $\mathcal{P}$ is live from a state s^1, by Th. 2 in [5, 4] there is a minimal live execution σ, such that $\sigma[1] = s^1$. Moreover, any other live execution σ' with $\sigma'[1] = s^1$ is equivalent to l repetitions of σ (for some $l \geqslant 1$), that is, $\mathbf{x}^{\sigma'} = l \cdot \mathbf{x}^\sigma$ [4, Cor. 2]. Futhermore, any valid execution σ'' with $\sigma''[1] = s^1$ can be extended to a live execution σ' [4, Th. 2], thus, equivalent to a certain number of repetitions of a minimal live execution σ. This property will be used to justify the algorithm in Sec. 4.

3.2 Additional Properties of Polygraph Executions

For a given valid execution $\sigma = \langle \mathbf{c}^1, \theta^1 \rangle \cdots \langle \mathbf{c}^n, \theta^n \rangle$ and a channel $e_i = \langle v_j, v_k \rangle$ with $j \neq k$, we can consider the total number of tokens produced (resp. consumed) on e_i by the firings of v_j (resp. v_k) along σ, denoted $\delta_i^+(\sigma)$ (resp. $\delta_i^-(\sigma)$). Formally, by Def. 2, 4 and 5, we have:

a)



b)

	v_j	y_j^ϵ	$\mathbf{c}_1^l$	$\mathbf{c}_2^l$	$\mathbf{c}_3^l$	$L_j(\overline{s}^l)$	$d_j(\overline{s}^l)$	b_1^l	b_2^l	b_3^l
σ_2	-	-	2;①②	2/2;①	2;①②	-	-	$\lambda_1\lambda_2$	λ_2	$\lambda_1\lambda_1$
	v_3	1	1;②	2/2;①	0;	$\{\lambda_1\}$	λ_1	λ_2	λ_2	
	v_2	1	1;②	1/2;	1;③	$\{\lambda_2\}$	λ_2	λ_2		λ_2
	v_1	1	2;②③	3/2;②	1;③	$\emptyset$	α	$\lambda_2\lambda_1$	λ_1	λ_2
	v_2	2	2;②③	2/2;②	2;③④	$\emptyset$	λ_2	$\lambda_2\lambda_1$	λ_1	$\lambda_2\lambda_2$
σ_2'	-	-	2;①②	2/2;①	2;①②	-	-	$\lambda_1\lambda_2$	λ_2	$\lambda_1\lambda_2$
	v_2	1	2;①②	1/2;	3;①②③	$\{\lambda_2\}$	λ_2	$\lambda_1\lambda_2$		$\lambda_1\lambda_2\lambda_2$
	v_3	1	1;②	1/2;	1;③	$\{\lambda_1,\lambda_2\}$	υ	λ_2		λ_2
	v_2	2	1;②	0/2;	2;③④	$\emptyset$	λ_2	λ_2		$\lambda_2\lambda_2$
	v_1	1	2;②③	2/2;②	2;③④	$\emptyset$	α	$\lambda_2\lambda_1$	λ_1	$\lambda_2\lambda_2$

Fig. 2. a) A polygraph $\mathcal{P}_2$, with v_1 the unique selector of modes $\Lambda_1 = \{\lambda_1, \lambda_2\}$ and its followers v_2 and v_3, and b) two step-by-step live executions σ_2 and σ_2' of $\mathcal{P}_2$.

$$\delta_i^+(\sigma) = \sum_{\substack{l\,:\,v_j \text{ fires on} \\ \langle \mathbf{c}^l, \theta^l\rangle \text{ in } \sigma}} (\lfloor c_i^{l+1}\rfloor - \lfloor c_i^l\rfloor), \quad \delta_i^-(\sigma) = -\sum_{\substack{l\,:\,v_k \text{ fires on} \\ \langle \mathbf{c}^l, \theta^l\rangle \text{ in } \sigma}} (\lfloor c_i^{l+1}\rfloor - \lfloor c_i^l\rfloor). \tag{1}$$

The number of tokens occupying e_i along σ is modified only by productions (by v_j) and consumptions (by v_k), thus using (1) we have:

$$\lfloor c_i^n\rfloor - \lfloor c_i^1\rfloor = \sum_{1\leqslant l<n} (\lfloor c_i^{l+1}\rfloor - \lfloor c_i^l\rfloor) = \delta_i^+(\sigma) - \delta_i^-(\sigma).$$

It follows that $\lfloor c_i^1\rfloor + \delta_i^+(\sigma) = \delta_i^-(\sigma) + \lfloor c_i^n\rfloor$. These two expressions compute the total *number of tokens transiting through channel e_i along* σ, that we denote by $\eta_i^\sigma \in \mathbb{N}$. We call these tokens the *footprint* of σ on channel e_i, and each of these tokens is identified by a *rank* $1 \leqslant l \leqslant \eta_i^\sigma$. They can be seen as the tokens initially occupying e_i and the $\delta_i^+(\sigma)$ tokens produced along σ. On the other hand, the same tokens can be seen as the $\delta_i^-(\sigma)$ tokens consumed along σ and the tokens occupying e_i after σ. In execution σ_2 of Fig. 2, four tokens transit through e_3 along σ_2, thus the footprint has $\eta_1^\sigma = 4$ tokens, and each token is shown as ①–④ for the states when they are occupying e_3.

The next result shows that η_i^σ, $\delta_i^+(\sigma)$, $\delta_i^-(\sigma)$ depend only on the number of firings of v_j and v_k in σ and the initial state of e_i, and not on the order of transitions in σ.

Proposition 1. *Let $\mathcal{P}$ be a polygraph, and $\sigma = \langle \mathbf{c}^1, \theta^1\rangle \cdots \langle \mathbf{c}^n, \theta^n\rangle \in S^+$ be a valid execution of $\mathcal{P}$ with tracking vector $\mathbf{x}^\sigma$. Let $e_i = \langle v_j, v_k\rangle \in E$ be a channel with $j \neq k$, and denote $r = \lfloor c_i\rfloor$. Then we have: $\eta_i^\sigma = \lfloor c_i^1\rfloor + \delta_i^+(\sigma) = \delta_i^-(\sigma) + \lfloor c_i^n\rfloor$, $\delta_i^+(\sigma) = \lfloor x_j^\sigma \gamma_{ij} + r\rfloor$, $\delta_i^-(\sigma) = \lceil x_k^\sigma |\gamma_{ik}| - r\rceil$.*

Proof. The first fact was shown above. We claim that the sums of $\delta_i^+(\sigma)$, $\delta_i^-(\sigma)$ in (1) can be rewritten as follows: $\delta_i^+(\sigma) = \sum_{f=1}^{x_j^\sigma} (\lfloor c_i^1 + f\gamma_{ij}\rfloor - \lfloor c_i^1 + (f-1)\gamma_{ij}\rfloor)$ and $\delta_i^-(\sigma) = -\sum_{f'=1}^{x_k^\sigma} (\lfloor c_i^1 + f'\gamma_{ik}\rfloor - \lfloor c_i^1 + (f'-1)\gamma_{ik}\rfloor)$.

Indeed, at each step, $c_i^{l+1} = c_i^1 + f\gamma_{ij} + f'\gamma_{ik}$ where f and f' are the number of firings, resp., of v_j and v_k in the prefix of length $l+1$ in σ. At most one of γ_{ij} and γ_{ik} is not an integer[4]; by symmetry, we can assume $\gamma_{ik} \in \mathbb{Z}$. Then in the first sum we have $\lfloor c_i^{l+1} \rfloor = \lfloor c_i^1 + f\gamma_{ij} \rfloor$ for all l, that implies the proposed rewriting. In the second sum, the rewriting follows from the fact: $\forall p, p' \in \mathbb{Q}, \forall m, m' \in \mathbb{Z}, \lfloor p+m \rfloor - \lfloor p+m' \rfloor = \lfloor p'+m \rfloor - \lfloor p'+m' \rfloor$.

We simplify the first sum $\delta_i^+(\sigma) = \lfloor c_i^1 + x_j^\sigma \gamma_{ij} \rfloor - \lfloor c_i^1 \rfloor = \lfloor x_j^\sigma \gamma_{ij} + r \rfloor$ as required since $r = \lfloor c_i \rfloor = \lfloor c_i^1 \rfloor - c_i^1$, and the second sum $\delta_i^-(\sigma) = \lfloor c_i^1 \rfloor - \lfloor c_i^1 + x_k^\sigma \gamma_{ik} \rfloor = -\lfloor x_k^\sigma \gamma_{ik} + r \rfloor$. The third formula follows from the fact: $\forall p \in \mathbb{Q}, -\lfloor p \rfloor = \lceil -p \rceil$. □

For the footprint of σ on e_i, we can consider a mapping $\mathrm{o}_i^\sigma : \{1, \ldots, \eta_i^\sigma\} \to \{0, \ldots, \delta_i^+(\sigma)\}$ (resp. $\iota_i^\sigma : \{1, \ldots, \eta_i^\sigma\} \to \{0, \ldots, \delta_i^-(\sigma)\}$) associating to each token in the footprint the rank of the firing of v_j (resp. v_k) that produced (resp. consumed) that token in σ. We call that rank the *production* (resp. *consumption*) *rank* of the token. By convention, a rank 0 is assigned to a token that was not produced (resp. consumed) by a firing in σ. In Fig. 2, since the 1st firing of v_2 consumes ① on e_2 and produces ③ on e_3, we have $\iota_2^\sigma(1) = 1$ and $\mathrm{o}_2^\sigma(3) = 1$.

Proposition 2. *In the assumptions of Prop. 1, we have:*

1. $\forall 1 \leqslant l \leqslant \lfloor c_i^1 \rfloor,\ \mathrm{o}_i^\sigma(l) = 0; \quad \forall \lfloor c_i^1 \rfloor < l \leqslant \eta_i^\sigma,\ \mathrm{o}_i^\sigma(l) = \lceil (l - c_i^1) / \gamma_{ij} \rceil;$
2. $\forall 1 \leqslant l \leqslant \delta_i^-(\sigma),\ \iota_i^\sigma(l) = 1 + \lfloor (l-1+r) / |\gamma_{ik}| \rfloor; \quad \forall \delta_i^-(\sigma) < l \leqslant \eta_i^\sigma,\ \iota_i^\sigma(l) = 0.$

Proof. The formulas with rank 0 follow from the definition. Given an index l with $\lfloor c_i^1 \rfloor < l \leqslant \eta_i^\sigma$, let us compute the rank f of firing of v_j producing the l^{th} token of the footprint. Considering the prefixes σ' of σ with $f = x_j^{\sigma'} \in \{1, 2 \ldots, x_j^\sigma\}$ firings of v_j we have to find the shortest prefix σ' producing that token. In other words, by Prop. 1, we have to find the smallest f such that $l \leqslant \delta_i^+(\sigma') = \lfloor c_i^1 + f\gamma_{ij} \rfloor$. Since $l \in \mathbb{Z}$, we look for the smallest f such that $l \leqslant c_i^1 + f\gamma_{ij}$, or equivalently, $f \geqslant (l - c_i^1)/\gamma_{ij}$. Thus, $\mathrm{o}_i^\sigma(l) = \lceil (l - c_i^1) / \gamma_{ij} \rceil$.

Following the same logic, for $1 \leqslant l \leqslant \delta_i^-(\sigma)$, we look for the shortest prefix σ' of σ consuming the l^{th} token of the footprint. By Prop. 1, we have to find the smallest f such that $l \leqslant \delta_i^-(\sigma') = \lceil f|\gamma_{ik}| - r \rceil$. Equivalently, we look for the smallest f such that $l - 1 < f|\gamma_{ik}| - r$, i.e. $f > (l - 1 + r)/|\gamma_{ik}|$. In other words, we look for the biggest f such that $f - 1 \leqslant (l - 1 + r)/|\gamma_{ik}|$. Thus, $f - 1 = \lfloor (l - 1 + r)/|\gamma_{ik}| \rfloor$. It follows that $\iota_i^\sigma(l) = 1 + \lfloor (l - 1 + r) / |\gamma_{ik}| \rfloor$. □

Remark 4. For two valid executions $\sigma, \sigma' \in S^+$ that are equivalent (i.e. $\sigma \simeq \sigma'$, cf. Remark 3), by Prop. 1, for all channels[5] $e_i = \langle v_j, v_k \rangle \in E$ with $j \neq k$, we have $\eta_i^\sigma = \eta_i^{\sigma'}$, $\mathrm{o}_i^\sigma = \mathrm{o}_i^{\sigma'}$, and $\iota_i^\sigma = \iota_i^{\sigma'}$. For example in Fig. 2, we have $\sigma_2' \simeq \sigma_2$, and the footprints and production/consumption ranks are the same in these executions.

[4] We see here the reason for that condition in Def. 1: if both $\gamma_{ij}, \gamma_{ik} \notin \mathbb{Z}$, this and the following results do not hold, and the order of transitions can be important.

[5] For the case of self-loops, excluded in the proposition, a similar result can be proved by separately considering matrices $\mathbf{\Gamma}^+, \mathbf{\Gamma}^-$ with production and consumption rates.

3.3 Mode Extension of the Polygraph Modeling Language

In the rest of the paper, when there is no risk of confusion, we will use the term *polygraph* for an extended polygraph for short.

Extended polygraph. A *mode* identifies a reconfiguration of a polygraph's behavior. When firing, an actor has a mode for that firing, called *decided mode*. In the channels, the tokens are labeled with modes. An extended polygraph has a *nominal mode* denoted α, an *undefined mode* denoted υ, and a set of *user modes* denoted Λ_M with $\Lambda_M \cap \{\alpha, \upsilon\} = \varnothing$. The *mode set* is the set $\Lambda = \Lambda_M \cup \{\alpha, \upsilon\}$.

Every actor v_j is associated with a subset of user modes $\Lambda_j \subseteq \Lambda_M$. The set Λ_j contains the modes selectable by actor v_j, and is called its *selection set*. An actor v_j with $\Lambda_j \neq \varnothing$ is called a *selector*, and the subset of such actors is denoted by $V_M \subseteq V$. When firing, a selector v_j chooses a *selected mode* $\lambda \in \Lambda_j$ with which it labels the tokens it produces. A non-selector labels the tokens it produces with its decided mode. We assume that the Λ_j form a partition of Λ_M (cf. (i) in Def. 7). Hence a given user mode can be selected by one and only one selector. In addition, a selection makes sense only if there are at least two elements to select from (cf. (i)).

Every actor v_k is associated with a non-empty subset of *enabling modes* $\mathrm{M}_k \subseteq \Lambda$. The decided mode for a firing of v_k should always belong to M_k, unless it is undefined. We assume (cf. (ii) in Def. 7) that either $\mathrm{M}_k = \{\alpha\}$, or there exists a unique selector $v_j \in V_M$ with $\mathrm{M}_k = \Lambda_j$. In the latter case, v_j is said to be the *selector of* v_k, and v_k is said to be a *follower of* v_j, denoted $v_j \rightsquigarrow v_k$, and the decided mode of v_k can only be a mode selectable by v_j. The only exception is the undefined mode υ, which becomes the decided mode of v_k when its mode cannot be decided. Note that a selector can be a follower of another selector.

The decided mode of a firing of actor v_k is determined based on the labels of the tokens consumed by this firing from a subset of its input channels, denoted $\Psi_k \subseteq \mathrm{in}(v_k)$, called the *deciding set of* v_k, and whose elements are called *deciding channels* of v_k. If $\mathrm{M}_k = \{\alpha\}$, we require (cf. (ii) in Def. 7) that $\Psi_k = \varnothing$, since v_k does not need information to decide its mode. If $v_j \rightsquigarrow v_k$, to determine its decided mode, v_k must have at least one deciding channel. If all the tokens it consumes from its deciding channels are labeled with the same user mode, this mode becomes its decided mode. If conflicting modes are read, the decided mode of v_k is undefined. In order to obtain user modes in its enabling set $\mathrm{M}_k = \Lambda_j$, v_k's deciding channels come either from v_j or another follower v_l of v_j (cf. (ii)).

To ensure a follower receives tokens labeled with a mode selected by its selector, there must be a directed path from that selector to that follower, composed exclusively of deciding channels (cf. (iii) in Def. 7). Moreover, if there is a cycle of followers composed of deciding channels, some follower can receive conflicting modes from (the shortest path from) its selector and from its predecessor in the cycle. We exclude such a backward propagation of mode selections (cf. (iv)).

Definition 7 (Extended polygraph). *An* extended polygraph *is a tuple* $\overline{\mathcal{P}} = \langle \mathcal{P}, \Lambda, \{\langle \Lambda_j, \mathrm{M}_j, \Psi_j \rangle\}_j \rangle$ *where* $\mathcal{P}$ *is a polygraph,* $\Lambda = \Lambda_M \cup \{\alpha, \upsilon\}$ *is a mode set, and the tuples* $\langle \Lambda_j, \mathrm{M}_j, \Psi_j \rangle$ *contain respectively the selection set, enabling set,*

and deciding set of actor $v_j \in V$, such that:
(i) $\Lambda_M = \coprod_{v_j \in V} \Lambda_j$; $\forall v_j \in V, |\Lambda_j| = 0$ *or* $|\Lambda_j| \geqslant 2$; $V_M = \{v_j \in V \mid \Lambda_j \neq \varnothing\}$;
(ii) for any v_k, *either* $\mathrm{M}_k = \{\alpha\} \wedge \Psi_k = \varnothing$,
or $\exists v_j \in V_M$, $\mathrm{M}_k = \Lambda_j \wedge \varnothing \neq \Psi_k \subseteq \{\langle v_l, v_k\rangle \in E \mid j = l \vee \mathrm{M}_l = \Lambda_j\}$;
(iii) if $\mathrm{M}_k = \Lambda_j$, *there is a path* $v_j = v_{l_1}, \cdots, v_{l_n} = v_k$ *of deciding channels;*
(iv) for any $v_j \in V_M$, *there is no cycle* $v_{k_1}, \cdots, v_{k_p} = v_{k_1}$ *in which* $\mathrm{M}_{k_l} = \Lambda_j$ *and all channels are deciding.*

In addition to a state of a polygraph (cf. Def. 3), a state of an extended polygraph $\overline{\mathcal{P}}$ contains an *actor mode mapping* $m : V \longrightarrow \Lambda$, which stores for an actor v_j the decided mode of its last firing, denoted $m_j = m(v_j)$. To capture the mode labels associated to tokens in the channels, the state of $\overline{\mathcal{P}}$ also has a *token labeling* $b : E \longrightarrow \Lambda^*$ mapping a channel e_i to a sequence of modes, denoted $b_i = b(e_i)$. In a given state $s = \langle \mathbf{c}, \theta \rangle$, in channel e_i, there are $\lfloor c_i \rfloor$ tokens (cf. Def. 2), so there is a mode label for each of them in FIFO order (see for example the three rightmost columns for the executions of Fig. 2).

Definition 8 (State). *A* state *of an extended polygraph $\overline{\mathcal{P}}$ is a tuple $\overline{s} = \langle s, m, b\rangle$ where $s = \langle \mathbf{c}, \theta\rangle \in S$ is a state of $\mathcal{P}$, m is an actor mode mapping such that $\forall v_j \in V$ we have $m_j \in \mathrm{M}_j \cup \{\upsilon\}$, and b is a token labeling for s such that $\forall e_i \in E$ we have $|b_i| = \lfloor c_i \rfloor$. We denote by $\overline{S}$ the set of all possible states for $\overline{\mathcal{P}}$, and by $\triangledown : \overline{S} \longrightarrow S$ the forgetful mapping that maps a state $\overline{s} = \langle s, m, b\rangle$ to $\triangledown(\overline{s}) = s$.*

Our next goal is to extend the fire and tick transitions between states of $\mathcal{P}$ to transitions between states of $\overline{\mathcal{P}}$. We first define, given a state $\overline{s} = \langle s, m, b\rangle \in \overline{S}$, a new state $\overline{s}' = \langle s', m', b'\rangle \in \overline{S}$ resulting from the firing of an actor v_j in state $\overline{s}$. We assume that $s = \langle \mathbf{c}, \theta\rangle$ and $\overline{s} = \langle \mathbf{c}', \theta'\rangle$. As mentioned in Sec. 3.1, only valid executions are relevant. For this reason, we only define partial mappings for the transitions in $\overline{\mathcal{P}}$ such that $\triangledown(\overline{s}) \vdash \triangledown(\overline{s}')$.

Decided Mode. We first define how the decided mode for a firing of v_j in $\overline{s}$ is determined. Only the mode labels of the $\delta_i^-(s)$ tokens consumed from each deciding channel e_i will influence the decision. The set of the relevant modes is thus defined by $L_j(\overline{s}) = \{b_i[k] \mid e_i \in \Psi_j, 1 \leqslant k \leqslant \delta_i^-(s)\}$.

If the set $L_j(\overline{s})$ is restricted to a singleton $\{\lambda\}$, then λ is the decided mode of v_j (Case 2 in the following Def. 9). If $L_j(\overline{s}) = \varnothing$, it means that $L_j(\overline{s})$ does not provide information to decide, and v_j will keep its last mode m_j (Case 1). The last possible case is that $|L_j(\overline{s})| \geqslant 2$, which means that $L_j(\overline{s})$ provides incoherent information, since several modes are possible. As explained in Sec. 2, this is an *indecision*, and v_j will switch to the undefined mode υ (Case 3).

For a non-follower v_j, since $\Psi_j = \varnothing$ (cf. (ii) in Def. 7), the set $L_j(\overline{s})$ is empty (Case 1 in Def. 9), and the decided mode always remains the same (nominal if the previous mode was nominal). Finally, if a predecessor of v_j propagated to v_j an undefined mode via one of its deciding channels, the decision for v_j is taken either by Case 2 or Case 3, and in both situations v_j also enters an undefined mode. Hence, the undefined mode is propagated to successors.

Definition 9 (Decided mode). *Let $\overline{\mathcal{P}}$ be an extended polygraph, $v_j \in V$ an actor, and $\overline{s} = \langle s, m, b\rangle \in \overline{S}$ a state with $s = \langle \mathbf{c}, \theta\rangle \in S$ such that $\forall e_i \in \mathrm{in}(v_j)$, $c_i \geqslant |\gamma_{ij}|$. Given the set $L_j(\overline{s})$, the* decided mode *$d_j(\overline{s})$ of v_j for its firing in state $\overline{s}$ is defined as follows:*

1. *if $L_j(\overline{s}) = \varnothing$, then $d_j(\overline{s}) = m_j$;*
2. *if $L_j(\overline{s}) = \{\lambda\}$ for some λ, then $d_j(\overline{s}) = \lambda$;*
3. *if $|L_j(\overline{s})| \geqslant 2$, then $d_j(\overline{s}) = \upsilon$.*

Extended transitions. We can now define the resulting state $\overline{s}'$ after the firing of v_j in state $\overline{s}$ as an extension of a firing in $\mathcal{P}$ (cf. Case 1 in Def. 10 below). The mode of v_j is set to its decided mode $m'_j = d_j(\overline{s})$, while for the other actors, their mode is unchanged (cf. Case 2).

By Def. 2 and Remark 1, for every input channel e_i of v_j, the firing of v_j consumes the first $\delta_i^-(s)$ tokens, so the token labeling b'_i for the remaining tokens is the suffix of b_i of length $|b_i| - \delta_i^-(s)$ (cf. Case 3). Since we only define a partial mapping for states where the firing of an actor does not result in a negative channel state, the resulting token labeling is always well defined.

When firing, v_j arbitrarily chooses a mode, and labels all the produced tokens with that mode. If v_j is not a selector, then it can only choose its decided mode. Otherwise, v_j is a selector, and can select any mode from its selection set (cf. (ii)). In Def. 10, we make the choice to represent the arbitrarily chosen mode λ as an additional parameter of the partial firing mapping, so that different choices can lead to different resulting states. Then for each output channel e_i, since the number of tokens produced is $\delta_i^+(s)$, a suffix $(\delta_i^+(s) * \lambda)$ is added to the token labeling sequence (cf. Case 4).

Definition 10 (Extended firing). *For an extended polygraph $\overline{\mathcal{P}}$, the partial mapping* $\mathrm{fire} : V \times \overline{S} \times \Lambda \nrightarrow \overline{S}$ *is defined for the tuples $\langle v_j, \overline{s}, \lambda\rangle$ such that (i) $\nabla(\overline{s}) \vdash \mathrm{fire}(v_j, \nabla(\overline{s}))$, and (ii) $\lambda = d_j(\overline{s})$ if $v_j \notin V_M$ or $\lambda \in \Lambda_j$ if $v_j \in V_M$. In this case, if we denote $\overline{s} = \langle s, m, b\rangle$ and $\langle s', m', b'\rangle = \mathrm{fire}(v_j, \overline{s}, \lambda)$, we have:*

1. *$s' = \mathrm{fire}(v_j, s)$;*
2. *$m'_j = d_j(\overline{s})$, and $m'_k = m_k$ for any $k \neq j$;*
3. *$\forall e_i \in \mathrm{in}(v_j)$, $b'_i = \mathrm{suffix}(b_i, |b_i| - \delta_i^-(s))$;*
4. *$\forall e_i \in \mathrm{out}(v_j)$, $b'_i = b_i \cdot (\delta_i^+(s) * \lambda)$.*

Definition 11 (Extended tick). *For an extended polygraph $\overline{\mathcal{P}}$, the partial mapping* $\mathrm{tick} : \overline{S} \nrightarrow \overline{S}$ *is defined for the tuples $\langle v_j, \overline{s}, \lambda\rangle$ such that $\nabla(\overline{s}) \vdash \mathrm{tick}(\nabla(\overline{s}))$. In this case, if we denote $\overline{s} = \langle s, m, b\rangle$ and $\langle s', m', b'\rangle = \mathrm{tick}(\overline{s})$, we have $s' = \mathrm{tick}(s)$, $m' = m$, and $b' = b$.*

Remark 5. Def. 9, 10, 11 show that the extended transitions in $\overline{\mathcal{P}}$ impact, or depend on the synchronous constraints θ in the same way as the underlying transitions in $\mathcal{P}$ do. There is no additional dependence or impact on synchronous constraints introduced by the mode extension. An extended firing only relies on the channel states $\mathbf{c}$ in $s = \langle \mathbf{c}, \theta\rangle \in S$ to determine the mode changes. Therefore, the mode extension is orthogonal to synchronous constraints. We thus chose not to detail them here. In other words, the mode extension is about *which* tokens are consumed or produced by a firing, not *when* they are.

Extended execution. An execution of $\overline{\mathcal{P}}$ relies on the extended firing and the extended tick. By construction, the underlying execution in $\mathcal{P}$ is valid.

Definition 12 (Extended execution). *An* execution *of an extended polygraph $\overline{\mathcal{P}}$ is a sequence $\overline{\sigma} = \overline{s}^1 \cdots \overline{s}^n \in \overline{S}^+$, such that $\forall 1 \leqslant k < n$ we have either $\overline{s}^{k+1} = \mathrm{fire}(v_j, \overline{s}^k, \lambda)$ for some $v_j \in V$ and $\lambda \in \Lambda$, or $\overline{s}^{k+1} = \mathrm{tick}(\overline{s}^k)$. The forgetful mapping is extended to any execution $\overline{\sigma} = \overline{s}^1 \cdots \overline{s}^n$ as follows: $\triangledown(\overline{\sigma}) = \triangledown(\overline{s}^1) \cdots \triangledown(\overline{s}^n)$. In addition, if $\sigma = \triangledown(\overline{\sigma})$, for all channels $e_i \in E$ we denote $\eta_i^{\overline{\sigma}} = \eta_i^{\sigma}$, $\mathrm{o}_i^{\overline{\sigma}} = \mathrm{o}_i^{\sigma}$, and $\iota_i^{\overline{\sigma}} = \iota_i^{\sigma}$.*

Coherence. As explained above, the decision of the next mode captures a drift in mode propagation by assigning an undefined mode to actors. We propose in the next section an algorithm to verify that, given a polygraph $\overline{\mathcal{P}}$ with an initial state $\overline{s}^1$, the decided mode is never undefined in any execution starting from $\overline{s}^1$. To show its soundness, we need to show (cf. Th. 1 below) that the decided modes of all actors are pre-determined by the initial state and the modes selected by the selectors, even if the order of transitions is changed.

To formalize this idea, for an execution $\overline{\sigma} = \overline{s}^1 \cdots \overline{s}^n \in \overline{S}^+$ and for each actor v_j, we define two mappings $\mu_j^{\overline{\sigma}}$ and $\chi_j^{\overline{\sigma}}$ giving for each $l \in \{1, \ldots, x_j^{\sigma}\}$ the decided mode $\mu_j^{\overline{\sigma}}(l)$ and the selected mode $\chi_j^{\overline{\sigma}}(l)$ for the l^{th} firing of v_j in $\overline{\sigma}$. Hence, if the f^{th} firing of v_j occurs in state $\overline{s}^l$ such that $\overline{s}^{l+1} = \mathrm{fire}(v_j, \overline{s}^l, \lambda)$, we have $\mu_j^{\overline{\sigma}}(f) = d_j(\overline{s}^l)$ and $\chi_j^{\overline{\sigma}}(f) = \lambda$. By Def. 10, they are equal for non-selectors.

Theorem 1. *Let $\overline{\mathcal{P}}$ be an extended polygraph, and $\overline{\sigma} \in \overline{S}^+$, $\overline{\sigma}' \in \overline{S}^+$ be two executions of $\mathcal{P}$ such that $\triangledown(\overline{\sigma}) \simeq \triangledown(\overline{\sigma}')$ and $\overline{\sigma}[1] = \overline{\sigma}'[1]$. Assume that for any selector $v_j \in V_M$ we have $\chi_j^{\overline{\sigma}} = \chi_j^{\overline{\sigma}'}$. Then for any actor $v_k \in V$ we have $\mu_k^{\overline{\sigma}} = \mu_k^{\overline{\sigma}'}$.*

Sketch of proof. By definition of $\simeq$ (see Remark 3), the tracking vectors of $\overline{\sigma}$ and $\overline{\sigma}'$ are equal, so each actor v_j fires the same number of times $x_j^{\sigma} = x_j^{\sigma'}$ in $\overline{\sigma}$ and $\overline{\sigma}'$, while the order of firings can be different. Assume the result does not hold. We can then choose the very first firing of an actor in $\overline{\sigma}$ for which the property does not hold. Assume this is the f^{th} firing of actor v_k (referred to below as *problematic*) for which the decided mode is not the same: $\mu_k^{\overline{\sigma}}(f) \neq \mu_k^{\overline{\sigma}'}(f)$. Hence for all previous firings (of all actors) in $\overline{\sigma}$, the required property holds.

To choose the decided mode of its f^{th} firing, v_k considers the tokens either initially present in the channel or produced by a firing of some actor occurring *before* this firing of v_k in $\overline{\sigma}$. By Prop. 2, a given token of the footprint of a channel is produced by the firings of the same rank of the producer, and consumed by the firings of the same rank of the consumer of the channel in both executions $\triangledown(\overline{\sigma})$ and $\triangledown(\overline{\sigma}')$ (and thus in $\overline{\sigma}$ and $\overline{\sigma}'$). Thus, the f^{th} firing of v_k in $\overline{\sigma}$ and $\overline{\sigma}'$ consumes the initial tokens of the same rank in the footprint, that is, exactly the same number and on the same position. Since $\overline{\sigma}[1] = \overline{\sigma}'[1]$, there cannot be any difference of modes for these tokens between $\overline{\sigma}$ and $\overline{\sigma}'$. Regarding the tokens produced by a firing of a selector in $\overline{\sigma}$, since the ranks of such tokens in the

footprint are the same in $\overline{\sigma}$ and $\overline{\sigma}'$ and since the selector's choice is the same in $\overline{\sigma}$ and $\overline{\sigma}'$, there cannot be any difference of token modes either. Regarding the tokens produced by a firing of a non-selector in $\overline{\sigma}$ *before* the problematic firing of v_k, there cannot be any difference since each such token is produced by a firing for which the property holds and therefore the token was labeled with the same decided mode in $\overline{\sigma}$ and $\overline{\sigma}'$. The contradiction finishes the proof. □

4 Method and Tool Support to Check Coherence

In this work, we design and implement in DIVERSITY an algorithm to check whether or not an indecision can occur in a live execution of a polygraph. DIVERSITY is a customizable model analysis tool based on symbolic execution, available in the *Eclipse Formal Modeling Project* [12].

The input of our algorithm is a polygraph $\overline{\mathcal{P}}$ and an initial state $\overline{s}^1$ such that $\mathcal{P}$ is live from $\nabla(\overline{s}^1)$. Assume σ_0 is a minimal live execution of $\mathcal{P}$ from $\nabla(\overline{s}^1)$. The algorithm performs symbolic execution of all executions $\overline{\sigma}$ of $\overline{\mathcal{P}}$ with initial state $\overline{s}^1$ such that $\nabla(\overline{\sigma})$ is a repeated execution of σ_0 (up to a certain number of times). This exploration is based on a straightforward implementation of Def. 10, 11, 12. We call *current state* the state $\overline{s}^l$ resulting from the last such application, with $\overline{s}^l = \overline{s}^1$ initially. In order to represent all possible choices modeled by the selected mode arguments of the extended fire transitions, for any firing of rank f of a selector v_j in these executions, we use a symbolic parameter a_{jf} representing that mode and on which we compute constraints (the initial constraint being that $a_{jf} \in \Lambda_j$). The produced tokens are labelled with a_{jf}. The resulting symbolic state represents all possible choices of selected modes and resulting constraints (stating that the mode labels of some tokens—having the same a_{jf}—are equal).

Each time an actor fires, we compute the conditions given in Def. 9. By construction at least one of them is satisfiable. If the condition corresponding to Item 3 is satisfiable, it means that there exists a valuation of the formal parameters for which the decided mode is undefined. In this case the computation ends with a verdict *no* stating that there is at least one live execution with an indecision. Otherwise, the exploration continues.

From Prop. 1, for any actor v_j and channel $e_i \in \text{in}(v_j)$, the $\lfloor c_i^1 \rfloor$ tokens initially occupying e_i are consumed after a known number of firings of v_j. Hence, from Remark 3, it is possible to successively execute σ_0 (with all possible choices of decided modes) a number of times $l > 0$ such that all tokens initially occupying input channels are consumed. Assume we reach some state $\overline{s}^n$; as σ_0 is live, we have $\nabla(\overline{s}^n) = \nabla(\overline{s}^1)$, so the number of tokens in the channels is the same as in the initial state. The labels of all tokens present in the channels at that stage are expressed by some a_{jf}. Finally, our algorithm executes σ_0 one last time, overall to a repetition of $l + 1$ minimal live executions σ_0. This last step, starting from some symbolic state $\overline{s}^n$, necessarily comes to a symbolic state equivalent to $\overline{s}^n$ (since by Prop. 2 the constraints on the mode labels of tokens—having the same a_{jf}—will be semantically the same, even if the indices j of a_{jf} will be shifted). If no indecision is detected by executing an extension of σ_0 from state $\overline{s}^n$, and since it leads to an equivalent state, we can stop iterations: any additional step will not detect an indecision. The computation ends with a verdict *yes*.

N°	example	♯ actors (♯ timed act.)	min live exec. len.	verdict	♯ firings simulated	♯ min. live execs.	time
1	Fig. 1	5 (4)	22	yes	44	2	95ms
2	Fig. 1(†)	5 (4)	220	yes	440	2	432ms
3	Fig. 2	3 (2)	8	yes	16	2	9ms
4	Fig. 4	15 (0)	1358	yes	2716	2	4s668ms
5	Fig. 4(†)	15 (0)	2702	yes	5404	2	14s716s
6	Fig. 4(†)	15 (0)	13580	yes	27160	2	1m41s
7	Fig. 4(‡)	15 (0)	1358	no	8	0	6ms
8	MP4-SP	5 (2)	299	yes	598	2	557ms
9	MP4-SP(†)	5 (2)	598	yes	6008	2	7s451ms
10	MP4-SP(§)	5 (2)	299	no	237	0	189ms

Fig. 3. Experiments on examples, where (†) denotes token rate modification preserving consistency, (‡) denotes a modified number of initial tokens, and (§) denotes marking an existing channel as deciding. The last three columns show the verdict, the number of firings and the number of full live executions executed by DIVERSITY.

The proposed technique is *sound* by construction: if an indecision is detected, it really occurs since the algorithm simulates a possible execution of $\overline{\mathcal{P}}$. To show its *completeness*, we should check that if an indecision can occur for some execution of $\overline{\mathcal{P}}$, the *no* verdict will be returned.

We claim that it is indeed sufficient to explore extensions of live executions as per our algorithm, and check that no indecision occurs in them. First, from Th. 1, the only parameters influencing the decided modes are the labels of the tokens initially occupying the channels, and the modes selected by the firings of selectors. The labels of the initial tokens are an input of the problem and do not change in the executions to consider. The symbolic parameters used to label the tokens produced by the firing of selectors cover all possible choices. Hence, all possible executions $\overline{\sigma}$ such that $\nabla(\overline{\sigma})$ is a repeated execution of σ_0 (any finite number of times, as argued above) are covered by our exploration. By Remark 3 and Th. 1, we deduce that a possible indecision in any execution of $\overline{\mathcal{P}}$ will be thus detected by our technique on an execution $\overline{\sigma}$ such that $\nabla(\overline{\sigma})$ is equivalent to a repeated execution of σ_0

We have applied our algorithm to different examples and summarized the results in Fig. 3. The three examples introduced in this paper were analyzed, and are referenced by Figure number (for Fig. 2 the tested model received synchronous constraints). The example denoted MP4-SP is a translation to Polygraph of the classical MPEG4-SP SADF decoder (see [13] for original graph). For some examples, transformations were applied to the initial model, in order to show how execution time increases linearly with the number of firings, or to voluntarily introduce an indecision. Correctness of verdicts was checked manually. Experiments were run on an Intel core i7-7920HQ@3.10GHz, 32GB RAM.

5 Discussion and Related Work

In [9], the authors introduced Synchronous Data Flow (SDF), a restriction of Kahn Process Networks [8], overcoming the undecidability of the existence of

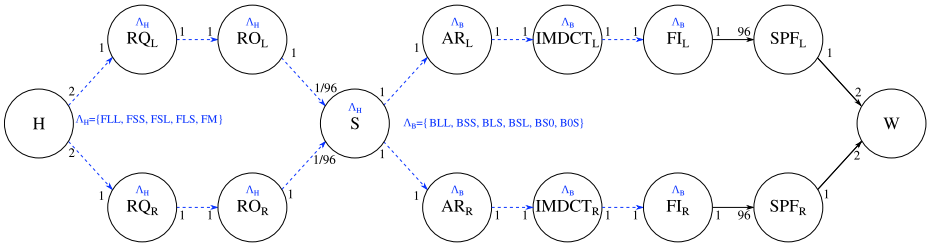


Fig. 4. A MP3 decoder modeled as a polygraph.

an admissible schedule (in the general case), and allowing static performance analysis. The key was the introduction of a linear behavior using static integer rates. For a more precise performance prediction, it is useful to allow rates to change dynamically during an execution to find tighter bounds on the evaluated memory footprint, throughput, and latency. The main difficulty to introduce such dynamic behavior resides in the capability to approximate the desired non-linear behavior while preserving the good properties of a linear equation system.

Our proposition is one of many other approaches attempting to tackle this issue [3, 2, 1, 10]. In Sec. 2, we mentioned SADF [14], and referenced a recent and extensive survey of similar approaches [6], showing that SADF is the most expressive while retaining the scheduling properties. Compared to SADF, Polygraph can express global synchronous constraints, simplifies the reconfiguration mechanisms by removing control channels, and offers a different approach to capture the dynamic changes in communication. Since the rational rates are the same in all modes, consistency and liveness can be checked as for static models. By adding configuration information per mode on the tokens, the existing examples with variable rates provided in [13] can be modeled in Polygraph without loss of behavioral information.

For example, in the translation of the MP3 decoder of Fig. 4, the actor H is a selector and determines the frame type for the left and right channels, and produces 2 granules of 576 components labeled with the determined frame types (*e.g.* FSL stands for short frame on left channel and long frame on right channel). Actor S is a follower of H, and it is also a selector, determining the amount and type of blocks to distribute to the block processing pipelines starting with actors AR_i. For example, if S consumes a granule labeled FSL on its 1^{st} firing, it will not consume another granule for its next 95 firings (input rates 1/96), and from the 1^{st} firing to the 96^{th}, the left channel will receive 96 short blocks each of size 6, and the right channel will receive 32 long blocks each of size 18. In the model, both channels receive 96 tokens, the first 32 are labeled BSL for short block on left and long block on right, and the remaining 64 tokens are labeled $BS0$ for short block on left and empty block on right. As in SADF, depending on the mode for a firing, the execution time can change and be set to 0 for actors that do not process for the determined frame and block types.

6 Conclusion

To cover the needs of modeling distributed and reconfigurable CPS, we have introduced dynamic behavior in Polygraph, such that for an extended polygraph, the consistency and liveness properties of the underlying static polygraph are not impacted by the extension. This allows for a single verification of these properties for a static model, and the static performance analysis of many alternative extended versions.

An adaptation of the existing approaches for the static performance analysis of similar languages will be the main part of our future work. In addition, we want to consider hierarchical and composable modeling of polygraphs, allowing the design of large scale complex distributed CPS.

Acknowledgement. Part of this work has been realized in the FACE/OPTEEM projects, involving CEA List and Renault. The Polygraph formalism has been used as a theoretical foundation for the software methodology in the project.

References

1. Bhattacharya, B., Bhattacharyya, S.S.: Parameterized dataflow modeling for DSP systems. IEEE Transactions on Signal Processing 49(10), 2408–2421 (2001)
2. Bilsen, G., Engels, M., Lauwereins, R., Peperstraete, J.A.: Cyclo-static data flow. In: Proc. of ICASSP. vol. 5, pp. 3255–3258 (1995)
3. Buck, J.T.: Static scheduling and code gen. from dynamic dataflow graphs with integer-valued control streams. In: Proc. of ACSSC. vol. 1, pp. 508–513 (1994)
4. Dubrulle, P., Gaston, C., Kosmatov, N., Lapitre, A., Louise, S.: Polygraph: A data flow model with frequency arithmetic, submitted
5. Dubrulle, P., Gaston, C., Kosmatov, N., Lapitre, A., Louise, S.: A data flow model with frequency arithmetic. In: Proc. of FASE. LNCS, vol. 11424, pp. 369–385 (2019)
6. Geilen, M., Falk, J., Haubelt, C., Basten, T., Theelen, B., Stuijk, S.: Performance analysis of weakly-consistent scenario-aware dataflow graphs. Journal of Signal Processing Systems 87(1), 157–175 (2017)
7. Geilen, M., Stuijk, S.: Worst-case performance analysis of synchronous dataflow scenarios. In: Proc. of CODES+ISSS. pp. 125–134. ACM (2010)
8. Kahn, G., MacQueen, D., Laboria, I.: Coroutines and Networks of Parallel Processes. IRIA Research Report, IRIA laboria (1976)
9. Lee, E.A., Messerschmitt, D.G.: Static scheduling of SDF programs for digital signal processing. IEEE Transactions on Computers C-36(1), 24–35 (87)
10. Oh, H., Ha, S.: Fractional rate dataflow model for efficient code synthesis. Journal of VLSI Signal Process. Syst. Signal Image Video Technol. 37(1), 41–51 (2004)
11. Plishker, W., Sane, N., Kiemb, M., Anand, K., Bhattacharyya, S.S.: Functional DIF for rapid prototyping. In: Proc. of Int. Symp. RSP. pp. 17–23 (2008)
12. The List Institute, CEA Tech: The DIVERSITY tool, `http://projects.eclipse.org/proposals/eclipse-formal-modeling-project/`
13. Theelen, B.D., et al.: Scenario-aware dataflow. Tech. Rep. ESR-2008-08, TUE (2008)
14. Theelen, B.D., Geilen, M.C., Basten, T., Voeten, J.P., Gheorghita, S.V., Stuijk, S.: A scenario-aware data flow model for combined long-run average and worst-case performance analysis. In: Proc. of MEMOCODE. pp. 185–194. IEEE (2006)
15. Wiggers, M.H., Bekooij, M.J., Smit, G.J.: Buffer capacity computation for throughput constrained streaming applications with data-dependent inter-task communication. In: Proc. of RTAS. pp. 183–194. IEEE (2008)