April 1990 LIDS-P-1964

Semi-Granules and Schielding for Off-line Scheduling

Bernard Le Goff
LIDS MIT, Cambridge, MA, USA
e_mail: legoff@lids.mit.edu

Paul Le Guernic
IRISA-INRIA, Université de Rennes 1, France

Julian Aráoz Durand
Universidad Simón Bolívar, Caracas, Venezuela

Abstract

In the framework of parallel programming, we use a type of directed graph, which we call the pin-graph, as a model for various applications: the vertices represent the elementary tasks; the arcs represent the internal dependences; and the pins represent communication with the outside. The connectivity of pin-graphs is studied in order to classify them into two classes. A pin-graph in the first class is called a semi-granule. The elementary tasks of a semi-granule can be ordered into a sequential scheduled such that no dead-lock due to this sequential schedule can appear; a schedule that has this last property is called a circuit-consistent schedule. A pin-graph that is not a semi-granule does not have any so scheduled. A partition into semi-granules of a pin-graph in this class is computed. The tasks of every so obtained semi-granule are well-ordered into a circuit-consistent schedule; and the composition of these schedules builds up a circuit-consistent schedule of the whole pin-graph, which thus cannot generate any dead-lock, whatever the context. Algorithms partitioning pin-graphs into semi-granules are surveyed.

Résumé

Dans le domaine de la programmation parallèle, nous utilisons comme modèle une variante du graphe orienté, nommée graphe à épingle. Les tâches élémentaires, dépendences entre tâches et communications avec l'extérieur de l'application considérée sont respectivement représentées par les sommets, arcs et épingles du graphe. La connexité des graphes à épingle est étudiée dans le but de les classer en deux catégories. Un graphe à épingle appartenant à la première de ces classes est appelé semi-granule. Pour une semi-granule, il existe un ordonnancement séquentiel de ses tâches élémentaires qui ne peut être la cause d'aucune étreinte fatale lors de l'exécution ; les ordonnancements ayant cette dernière propriété sont qualifiés d'ordonnancements acircogènes. Les graphes à épingle qui ne sont pas des semi-granules n'admettent nul tel ordonnancement. Une partition en semi-granules des graphes à épingle de cette seconde classe est calculée. Les tâches de chaque semi-granule ainsi obtenue sont alors totalement ordonnées en un ordonnancement acircogène ; et la composition de ces ordonnancements produit un ordonnancement acircogène du graphe tout entier, lequel ne peut ainsi être la cause d'aucune étreinte fatale, quelquesoit le contexte. Des algorithmes de partitionnement d'un graphe à épingle en semi-granules sont introduits.

Contents

List of Figures

Chapter 1

Introduction

The study we present in this paper may be useful to those who model real situations by directed graphs. In our case, the directed graphs formalize data dependences between tasks: the arcs represent the dependences and the vertices represent the tasks. From this model, many problems have been stated as NP-complete problems: the timing or scheduling, and the assignment or mapping of these tasks are very famous examples.

We have actually considered both in the context of the development of a compiler of a language for real-time applications or dynamic systems: the synchronous language SIGNAL [GLB87]. In short, this compiler generates a data dependences graph of the elementary tasks described in the program SIGNAL[BLL88]. The point is to implement efficiently the so obtained graph onto multiprocessors. For this, the two above mentioned problems have to be solved; algorithms that can map the graphs onto the chosen architectures, and that can compute the timing of execution of all the tasks have to be implemented.

1.1 Reducing the size when we cannot reduce the complexity

Our primary objective is to reduce the computational cost of both the scheduling and the mapping problems. Not being able to reduce the complexity of these problems, we could try to reduce their repective sizes, i. e., the sizes of the graphs we have to treat. For mapping, this reduction can be obtained from a partition of the graph such that each part can be mapped as a whole onto a processor. Thus, only the graph of the parts has to be mapped. The problem is to find the partitions that befit the assignment. For scheduling, the complexity can be reduced if it is possible to decide an order locally. The problem here is to find a local criterion that allows one to know whether a given order of several tasks is compatible with all the dependences of the whole graph or not. Once partitions have been formed and a local criterion has been found; we can then do the mapping and the scheduling. For this, the standard algorithms may be used.

The class of possible partitions has been formally defined. These are partitions such that their parts are what we call granules. A granule is a sub-graph which can be considered to be a black-box; it can therefore be assimilated into a regular vertex during the mapping. This aspect is developed in Le Goff 89 [Le 89], Figueira & al. 88 [FGLL88], Le Goff & Le Guernic 88 [LL88].

1.2 Off-line scheduling

This paper is devoted to the scheduling aspect of the problem. We assume the mapping is done and thus, for each processor, a schedule has to be computed.

Let us discuss the choice of scheduling off-line, so-called static scheduling. One of the motivations for this choice is the limitation of overhead: providing a static schedule precludes the use of a dynamic scheduler. However, a static schedule is rigid, and, consequently, can generate delays that a dynamic scheduler could avoid. But the interest of static scheduling is not only at the level of the quantitative aspects of the performances of the applications, but at the level of the qualitative aspects of its performances, as well. Indeed, one of the most important difficulties of the implementation of parallel programs is to be able to obtain deterministic executions of them. This is the key point in parallel programming [BIM88]. However, if all the scheduling decisions are made beforehand, the executions will be deterministic [GIB89]. In fact, this seems to be the fundamental justification of the research of static scheduling algorithms.

Basically, a schedule of a set of tasks is an ordering of these tasks such that they can be performed in this order. By scheduling the tasks in advance, the ordering, which is so built, can be incompatible with the constraints imposed by the context of the execution; a dead-lock will appear during the run. Insuring that the ordering will not create dead-locks, and modifying the ordering each time it does so is the complex part of the scheduling.

In order to reduce the computation of this part, our idea is to consider a partition of the tasks such that the tasks of each part of the partition can be scheduled in an ordering that is compatible with the rest of the constraints. And, moreover, the schedule of each part can be computed without looking at any other part. The computation of a local schedule is local: only the information included in the considered part is used.

1.3 Paper organization

In chapter 2 the criterion that determines whether a graph, representing a set of tasks and their dependences, can be sequentially scheduled or not is formalized.

We assume that the mapping has been done. Thus, the set of tasks mapped onto a processor has to be scheduled into a well-ordering. The sub-class of the graphs that verify the criterion is characterized; the graphs of this class are called semi-granules.

This leads to the third chapter, in which a method of scheduling graphs that are not semi-granules is developed. We know that it is impossible, in this case, to schedule off-line all the vertices of the graph in a sequential ordering. The reason is that we cannot order communications with the context without risking the possibility of a dead-lock. So a partition of the graph into semi-granules may be computed. The properties of the semi-granules allow one to compute a schedule of the whole graph by computing sequential schedules of each semi-granules.

Chapter 2

Sequentialization and semi-granules

In the framework of the implementation onto multiprocessors, we consider, in this chapter, the problem of static sequentializations, and therefore, of course, the problem of the run-scheme for each of the processors.

2.1 Dead-locks and reinforcement

Let us consider the following basic problem: we have several tasks to perform, and we have only one resource. What do we have to do? Of course, we have to well-order the set of tasks. By doing so, the resource will be able to be assigned sequentially to every task, and every task will be able to be performed.

2.1.1 Internal dependences

Tasks can have relations among themselves. For example a task x can provide another, y, with information. If y needs this information to be performed, then we must consider that y depends on x. These dependences constrain the choice of the well-ordering of the set of tasks.

A directed graph is perfectly suitable to represent such a set of tasks and their dependences. The tasks are represented by the vertices, and the dependences by the arcs. We can consider only the directed graphs that do not include circuits: in our framework a circuit would express a dead-lock, thus we assume the circuits were detected before this stage. Consequently, Γ, the set of dependences of a directed graph, $G = (X, \Gamma)$, generates an ordering by reflexo-transitive closure; we denote the so obtained new set of dependences Γ^*, and the associated ordering $\leq_G$ or $\leq_\Gamma$.

Now we can introduce the basic operation we use to build the well-ordering of the set of tasks: the reinforcement.

Definition 2.1 (reinforcement) *Let $G = (X, \Gamma)$ and $G' = (X, \Gamma')$ be circuit-free directed graphs, G' is a* **reinforcement** *of G iff*

$$\Gamma^* \subset \Gamma'^*$$

Indeed, the well-ordering of the set of tasks that we build has to be compatible with the initial dependences between the tasks; in other words, the well-ordering $\leq_{\Gamma'^*}$ has to include the ordering $\leq_\Gamma$ ($x \leq_\Gamma y \Longrightarrow x \leq_{\Gamma'^*} y$).

Moreover, no circuit can be closed by building the well-ordering (($x \leq_{\Gamma'^*} y$ *and* $y \leq_{\Gamma'^*} x) \Longrightarrow x = y$): the well-ordering is an ordering!

Finally, let us notice that adding an arc is not sufficient to reinforce a graph: if we add an arc which is in the reflexo-transitive closure, this closure is not changed (($\Gamma \subset \Gamma'$ *and* $\Gamma' \subset \Gamma^*) \Longrightarrow \Gamma^* = \Gamma'^*$).

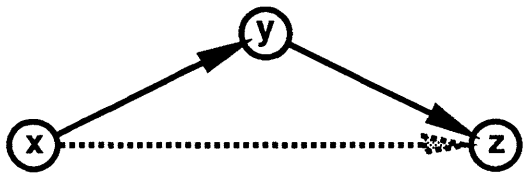

Figure 2.1: Adding an arc that does not result in a reinforcement

Figure 2.1 shows we can effectively add the arc (x, z) to this graph without changing the associated ordering: the addition of the dotted line arc does not modify Γ^*.

Corollary 2.1.1 *Let G be a circuit-free directed graph, let G_R be a reinforcement of G, a reinforcement of G_R is a reinforcement of G.*

2.1.2 External dependences

The group of tasks and their dependences represented by a directed graph may communicate with the outside. Some tasks may either need information from the outside or provide the outside with information or even both.

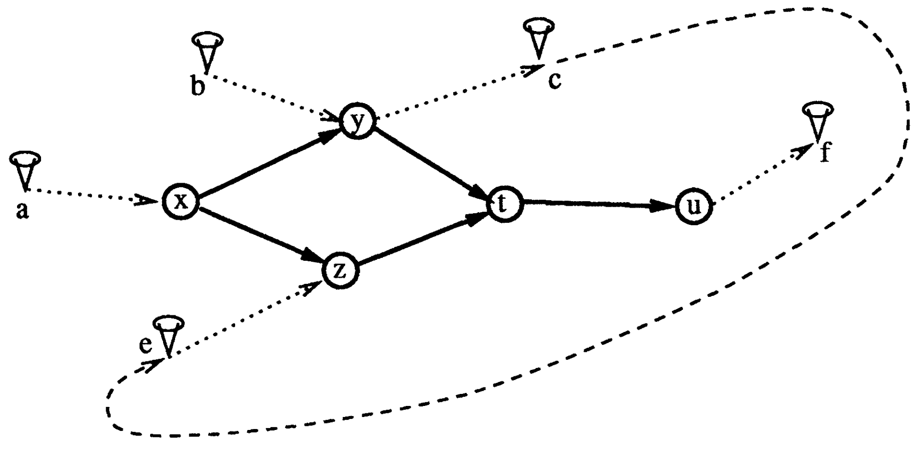

Figure 2.2: The reinforcement problem

The reinforcement of graphs that communicate creates a big problem. Indeed, the obtained new ordering can be incompatible with the context of the graph (see Figure 2.2): let us suppose a task z needs information from e to be performed, and the context needs information from c, which has to be provided by a task y, in order to provide e with information; then it is forbidden to add the arc (z, y) by reinforcing the graph representing the considered group of tasks, otherwise a dead-lock shall happen.

Consequently, it is necessary to include the potential for communication with the outside in our formal representation. In Lengauer 86 [Len86], such a case is considered. Lengauer uses a special kind of directed graph: the pin-graph. He simply added marks to the vertices that can communicate with the outside. These marks are called pins; there are input-pins (respectively output-pins) to mark vertices that can receive information from the context (respectively send to the context); a vertex can both receive and send information.

Definition 2.2 (pin-graph) *We call a* **pin-graph** *the quintuple* $G = (X, \Gamma, V, I, O)$ *where*
(X, Γ) *is a directed-graph,*
V *is a finite set,*
$I \subset V \times X$, *called* **input-pins** *set of* G,
$O \subset X \times V$, *called* **output-pins** *set of* G.

A vertex carrying an input-pin (respectively an output-pin) is called an input (respectively an output). A vertex can be both input and output.

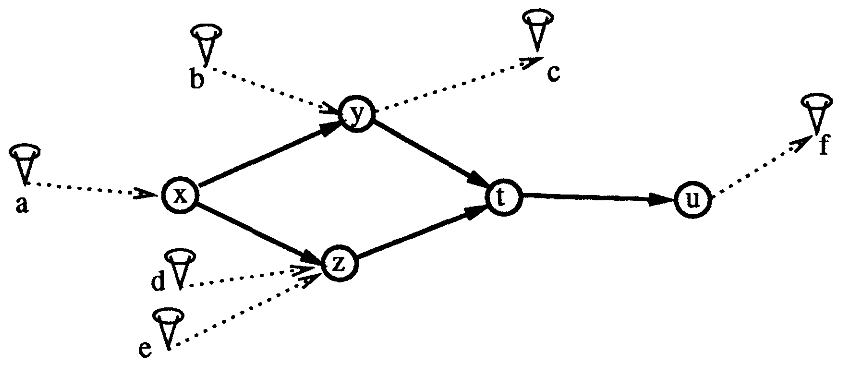

Figure 2.3: The pin-graph G_ℓ

Figures 2.3 and 2.4 show two pin-graphs. The following table gives their description.

	G_ℓ	G_r
X	$\{x, y, z, t, u\}$	$\{v, w, p, q, r\}$
Γ	$\{(x, y), (x, z), (y, t), (z, t), (t, u)\}$	$\{(q, p), (q, r), (v, w), (w, r)\}$
V	$\{a, b, c, d, e, f\}$	$\{c, e, f, g, h, k\}$
I	$\{(a, x), (b, y), (d, z), (e, z)\}$	$\{(c, v), (f, r), (k, q)\}$
O	$\{(y, c), (u, f)\}$	$\{(p, e), (r, h), (w, g)\}$

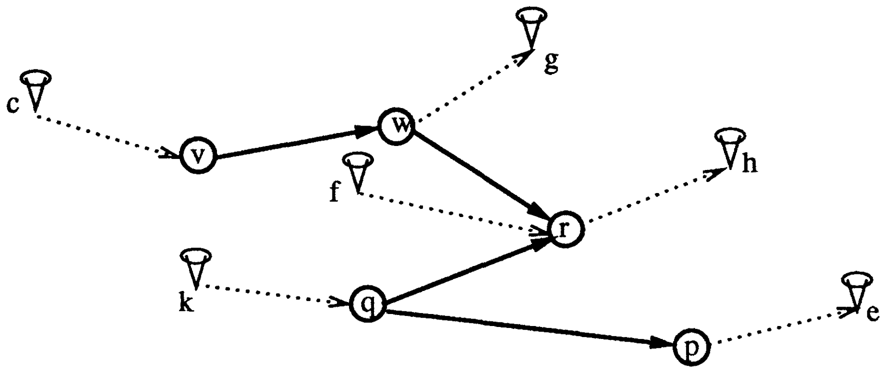

Figure 2.4: The pin-graph G_r

In order to simplify the theory, we state as an assumption **every vertex belongs to a path that goes from an input to an output** of the considered pin-graph. All pin-graph describing a reality can be reduced to one of these so defined sub-classes of pin-graphs without losing any real modeling ability. Indeed, such reductions can be done by removing some vertices: those that represent tasks that may never be performed, for these tasks do not have any input, and those that represent tasks whose the results may never be used, since they do not have any output.

2.1.3 Circuit consistency

The notion of reinforcement of a directed graph is generalized to a notion of reinforcement of a pin-graph. Let us consider the pin-graph G_ι. It is clear that any reinforcement of G_ι including the arc (z, y) creates the previously examined situation: such a reinforcement may be incompatible with the context and may provide a dead-lock (see Figure 2.2). There is, in fact, only one way to reinforce G_ι in order to obtain a "for all context compatible" well-ordering of its vertices: we add the arc (y, z). The following figure (Fig. 2.5) shows the result of this reinforcement.

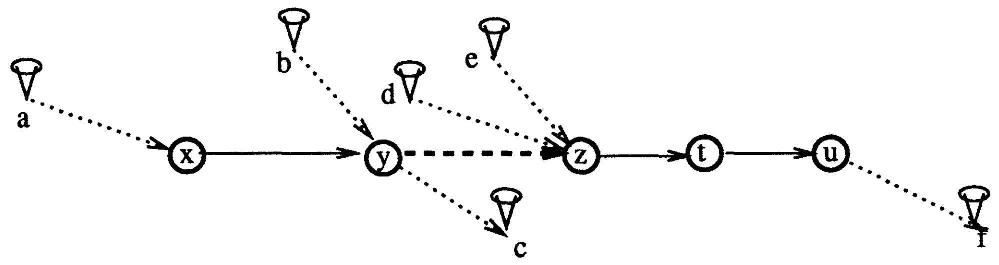

Figure 2.5: G_{R_1}: a sequential schedule of G_ι

The case of the pin-graph G_τ is more complex. Actually, three additional arcs can induce dead-locks in bad contexts: (q, w), (r, p) and (v, p). Therefore, we have to avoid reinforcing G_τ by adding one of these arcs. In reality, the problem is more difficult than this, because avoiding these arcs is not sufficient. They must not be represented by a path in the reinforcement. The following figure (Fig. 2.6) shows such a case.

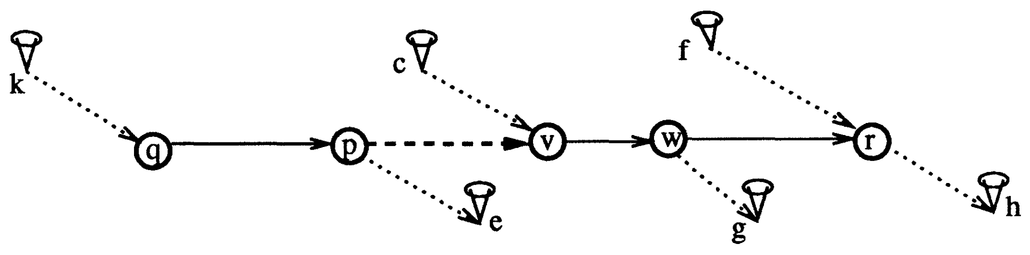

Figure 2.6: G_{R_2}: a sequential schedule of G_τ

In the reinforcement shown in this figure, only the arc (p, v) has been added. This arc is not in the list of arcs to be avoided ((q, w), (r, p) and (v, p)), but, one of them, the arc (q, w), is represented by the path (q, p, v, w). The schedule so represented requires reading the information from k before providing g with information. Of course, a context (a bad one) that would need information from g in order to provide k with information would create a dead-lock.

Let us consider a pin-graph G, and G_R, one of its reinforcements. And let there be a context that closes a circuit C through G_R. This circuit may be due to the initial constraints represented in G. For example, if a context of G_τ needs information from h in order to provide f with information, then a circuit is closed, and no reinforcement is responsible for that; that simply means the described application is impossible. But the circuit may be due to the aditional arcs of G_R; in this case the reinforcement is responsible for the dead-lock. Actually, the reinforcements can be classified into two types. Those of the first type cannot be responsible for a circuit's being closed by a context (think of G_{R_1} for example); we call them circuit-consistent reinforcements: if there is a circuit, then it is not due to the additional arcs. Those of the second type can be responsible for a circuit's being closed by a context (think of G_{R_2} for example).

Definition 2.3 (circuit-consistency) *A reinforcement G_R of a circuit-free pin-graph G is* **circuit-consistent** *iff*

$$\left(\begin{array}{c} \forall(a,x) \in I \\ \forall(y,b) \in O \end{array} \right) \left(x \leq_{G_R} y \implies \left(\begin{array}{c} \exists(a,x') \in I \\ \exists(y',b) \in O \end{array} \; x' \leq_G y' \right) \right) \tag{2.1}$$

This formalizes the following idea: if the reinforcement requires reading information from a before providing b with information, then this constraint existed in the initial graph.

The following figure shows why the condition (2.1) of Definition 2.3 is so complex.

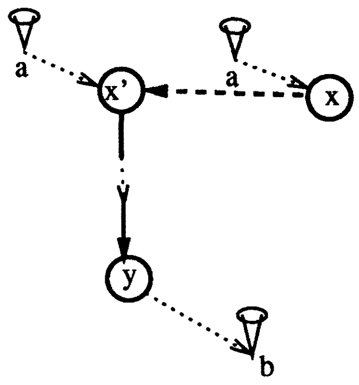

Figure 2.7: A pathological case concerning circuit-consistency

Indeed, the same information can be read at several vertices of the pin-graph. For example, Figure 2.7 shows a pin-graph in which information from a is read at both the vertices x and x'. Let us suppose this graph has been reinforced by adding the arc (x, x'). A context that would need information from b in order to provide a with information would close a circuit through $(x, x', \cdots, y)$, but this context would have closed another circuit through $(x', \cdots, y)$, which is a path of the initial graph. Therefore, though a circuit due to the reinforcement appears, this reinforcement is circuit consistent, for, simultaneously, another circuit, which is not due to the reinforcement but only to the initial graph itself, appears, too.

We propose to simplify these technical aspects by considering a stronger property.

Corollary 2.3.1 (strong circuit-consistency) *A reinforcement G_R of a circuit-free pin-graph G such that for all input x of G and for all ouput y of G:*

$$x \leq_{G_R} y \implies x \leq_G y \tag{2.2}$$

is circuit-consistent. Such a reinforcement is called **strongly circuit consistent.**

The sub-class of strongly circuit-consistent reinforcements is sufficient. Indeed, without losing the power of representation of the pin-graphs, we can map any pin-graph onto another for which a circuit-consistent reinforcement is necessarily a strongly circuit-consistent reinforcement. The following figure shows such a mapping.

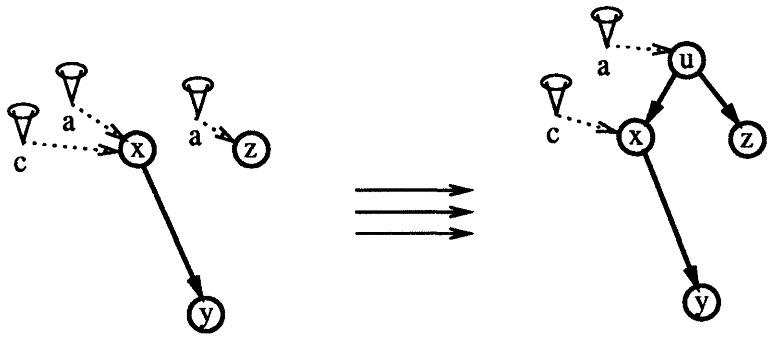

Figure 2.8: Equivalent cases between circuit-consistency and strong circuit-consistency

In the pin-graph shown in this figure, u can represent a fictitious task; in this sense we can consider the power of representation of the two (the pin-graph on the left and the one on the right) to be exactly the same.

2.1.4 Conclusion

So, in this section, reinforcement, which is what we do by scheduling, and its main problem, i. e., the increase of the risk of dead-lock, are formalized. The reinforcements that do not increase the risk of dead-lock are characterized as reinforcements that do not modify the dependences between the inputs and the outputs: the circuit-consistent reinforcements. Thus the problem is reduced to the level of the input-output pairs: the reinforcements must not modify the input-output connectivity.

2.2 Avoiding dead-lock by shielding

The problem having been formally stated, the question now is how to solve it. The first task is to state a constructive characterization of the circuit-consistent reinforcement.

2.2.1 Shielding (first characterization of the semi-granule)

As we have just shown the problem is at the level of the input-output pairs. Let us consider an input-output pair (x, y) of a circuit-free pin-graph G; there are three cases to consider:

1. $x \leq_G y$; in this case the context can close a circuit through (x, y), but it will be due to G itself, and thus we shall not worry about it;
2. $y \leq_G x$; in this case no circuit can be closed through (x, y) by a context;

3. $x \not\leq_G y$ and $y \not\leq_G x$; in this case also, no circuit can be closed by a context, if we consider G by itself; but some reinforcement G_R can change the situation.

This third case is the one we have to regard with caution. And special care must be taken about the reinforcements that include $x \leq_{G_R} y$, because, then, a (bad) context may close a circuit for which the considered reinforcement is responsible.

Definition 2.4 (communication and shielding) *Let G be a pin-graph, x be an input and y be an output of G.*

- *(x, y) is a* **communication pair** *iff $x \not\leq_G y$.*
- *(y, x) is a* **shielding-arc** *iff $x \not\leq_G y$ and $y \not\leq_G x$.*

We denote $COM(G)$ the set of pairs of communication of G, and $SHI(G)$ the set of shielding-arcs of G.

The idea of avoiding the dead-locks due to reinforcements is to reinforce, in a good way, and in advance, the pin-graphs that include communication pairs. In order to do so, we add all the shielding-arcs; this gives a new pin-graph $G_T = (X, \Gamma \cup SHI(G), V, I, O)$, called the shielded-graph of G, which is a reinforcement of $G = (X, \Gamma, V, I, O)$.

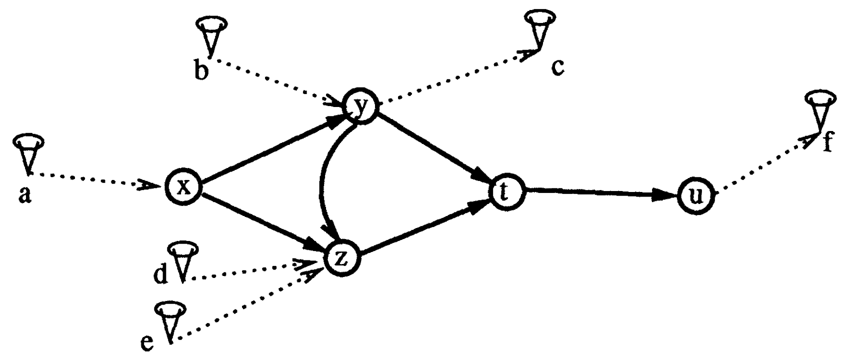

Figure 2.9: The shielded-graph of G_ι is circuit-free

Figure 2.9 and 2.10 show the shielded-graphs of G_ι and G_r respectively.

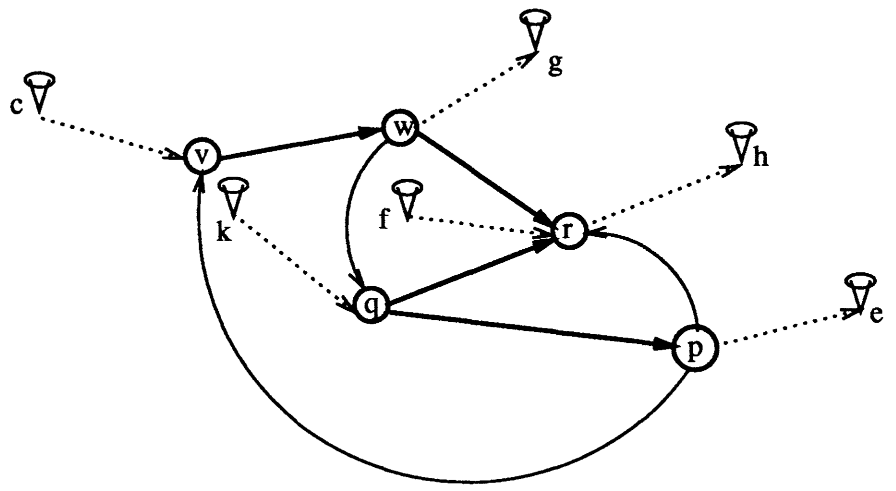

Figure 2.10: The shielded-graph of G_T includes a circuit

Figure 2.10 shows that the shielded-graph of G_T includes the circuit (v, w, q, p, v); therefore, it is not a reinforcement of G_T. The existence of a circuit in the shielded-graphs is a key to the solution of our problem.

Proposition 2.1 *Any reinforcement of a shielded-graph of a pin-graph G is a strongly circuit-consistent reinforcement of G.*

Proof

Obviously, if the shielded-graph of G includes a circuit, then no reinforcement exists, and the proposition is true. This is the case if G is not circuit-free.

Let us consider the other cases. Let G be a circuit-free pin-graph and G_T be its shielded-graph, which is assumed circuit-free, too. Let G_{T_R} be a reinforcement of G_T. Let x be an input of G and y be an ouput of G, such that $x \leq_{G_{T_R}} y$.

A reinforcement is circuit-free; therefore G_{T_R} is so, too, and $x \not\geq_{G_T} y$. This means (y, x) is not a shielding-arc of G, and (x, y) is not a communication pair; therefore $x \leq_G y$.

This proves that, for any given input-output-pair (x, y) of G, $x \leq_{G_{T_R}} y \implies x \leq_G y$, which is the definition of the strong circuit-consistency of G_{T_R}.

□

Proposition 2.2 (semi-granule) *Let G be a pin-graph, and G_T be its shielded-graph. The two following properties are equivalent:*

1. *G_T is circuit-free;*
2. *there is a strongly circuit-consistent reinforcement of G that well-orders the vertices of G.*

A pin-graph G for which 1 and 2 are true is called a **semi-granule**.

A reinforcement of a pin-graph well-ordering its vertices is called a sequential schedule (may be denoted 1-schedule).

Proof

(1) $\Longrightarrow$ (2) ((1) $\Longrightarrow$ G_T is a reinforcement of G) is a corollary of the definitions of a reinforcement and a shielded-graph.

(G_T is a reinforcement of G $\Longrightarrow$ (2)) is a corollary of the proposition 2.1, by taking a reinforcement of G_T that well-orders the vertices of G (a 1-schedule of G).

(2) $\Longrightarrow$ (1) Let G_S be a strongly circuit-consistent 1-schedule of G. We are going to prove that $\Gamma_T^* \subset \Gamma_S^*$, which implies that, since G_S is circuit-free, G_T is also circuit-free.

Let (y, x) be a shielding-arc of G ($(y,x) \in \Gamma_T \subset \Gamma_T^*$). Then $x \not\leq_G y$ and $x \not\leq_{G_S} y$, because of the strong circuit-consistency of G_S. Finaly, since G_S is a 1-schedule, all the pairs of vertices are ordered; therefore $y \leq_{G_S} x$ ($(y,x) \in \Gamma_S^*$).

□

So, the pin-graph G_ι is a semi-granule, for its shielded-graph (see Fig. 2.9) is circuit-free, while the pin-graph G_τ is not so, since its shielded-graph includes the circuit (v, w, q, p, v) (see Fig. 2.10).

2.2.2 Second characterization of the semi-granules

The former definition of the semi-granule is derived from the property of sequentialization. It does not give a test to detect whether a pin-graph is a semi-granule or not. Moreover, from this definition, how to build a reasonable complexity test is not clear. There exists another characterization of the semi-granule which provides immediately a test of polynomial complexity. The following lemma is the first step towards this new definition of the semi-granules. It is illustrated by Figure 2.11, which follows.

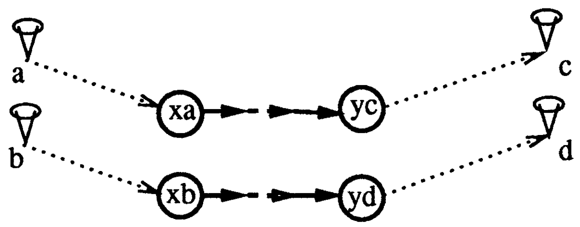

Figure 2.11: This situation may not be in a semi-granule

Lemma 2.1 *Let $G = (X, \Gamma, V, I, O)$ be a circuit-free pin-graph. The following properties are equivalent:*

1. *G is a semi-granule;*
2. $\forall (a, x_a) \in I\ \forall (b, x_b) \in I\ \forall (y_c, c) \in O\ \forall (y_d, d) \in O$
   $(x_a \leq_G y_c)$ *and* $(x_b \leq_G y_d)$ $\Longrightarrow$ $(x_b \leq_G y_c$ *or* $x_a \leq_G y_d)$.

Proof

(1) $\Longrightarrow$ (2) G is a semi-granule; therefore G_T, its shielded-graph, is circuit-free (by definition). Moreover, for all input-output pair (x, y) of G, $x \leq_{G_T} y$ or $y \leq_{G_T} x$; therefore necessarily, $x_a \leq_G y_d$ or $x_b \leq_G y_c$, otherwise we would have, in G_T, $y_d \leq_{G_T} x_a$ and $y_c \leq_{G_T} x_b$, which would close the circuit $(x_a, y_c, x_b, y_d, x_a)$ in G_T.

(2) $\Longrightarrow$ (1) (by contradiction) That is illustrated by the next figure. Now, let us suppose that G_T has a circuit C. We assume that C is elementary (it does not include any smaller circuit). G itself is circuit-free; therefore, C includes at least a shielding-arc. Let (y, x) be this arc. (y, x) is a shielding-arc; therefore, (x, y) is a communication pair and there is no path from x to y in G. This implies that C includes another shielding-arc, let (y', x') be the first shielding-arc that we can find along the circuit C from x to y ($x \leq_G y'$).

Either the x' through y part of C is completely included in G ($x' \leq_G y$) and x, x', y and y' build a counter-example to (2), or there exists another shielding-arc between x' and y. Then (x', y) is a communication pair and (y, x') is a shielding-arc; therefore, (y, x') is a shortcut which closes a circuit in G_T, and which is included in C. But, since C is elementary, this is impossible.

□

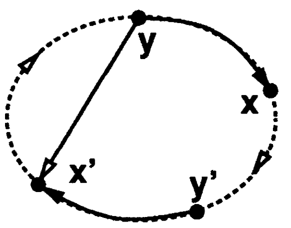

Figure 2.12: Illustration of Lemma 1's proof

Actually, Lemma 2.1 defines precisely how the circuits are created during the shielding. The reader can look at Figure 2.10, showing the shielded-graph of G_T, in order to confirm that there are two couples of input-output pairs that contradict the second property of the Lemma:

1. (v, w) with (q, r);
2. (v, w) with (q, p).

So, detecting the eventual apparence of circuits during the shielding is equivalent to finding these couples of input-output pairs. This can be done from a summary of the internal communications between the inputs and the outputs. This summary is called a profile.

Definition 2.5 (profile) *The profile of a pin-graph G is the pair* $(\{I_i \ / \ 1 \leq i \leq n\}, \{O_i \ / \ 1 \leq i \leq n\})$.

$\{I_i \ / \ 1 \leq i \leq n\}$ *is the partition of the inputs of G defined by the following equivalence relation: two inputs x_1 and x_2 are equivalent iff for all ouput y, $x_1 \leq_G y \iff x_2 \leq_G y$.*

$\{O_i \ / \ 1 \leq i \leq n\}$ *is the spanning of the outputs of G defined such as: an output y of G is in a part O_i of the spanning iff for all input x in I_i, $x \leq_G y$.*

$\{O_i \ / \ 1 \leq i \leq n\}$ is actually a spanning because any vertex of the considered pin-graphs belongs to a path from an input to an output. Moreover, it is clear that $(\{O_i \ / \ 1 \leq i \leq n\}, \subset)$ is an ordering, which may be partial.

The following theorem characterizes the class of pin-graphs that include bad couples of input-output pairs.

Theorem 1 *Let G be a circuit-free pin-graph. The following properties are equivalent:*

1. *G is a semi-granule;*
2. *$(\{O_i \ / \ 1 \leq i \leq n\}, \subset)$ is a well-ordering.*

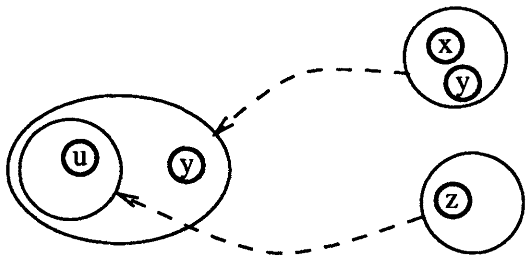

Figure 2.13: The profile of the semi-granule G_ι

Figure 2.13 shows the profile of the pin-graph G_ι, which is a semi-granule, and effectively its output spanning is well-ordered. On the other hand, this is not the case for G_τ, which is shown in Figure 2.14; indeed, G_τ is not a semi-granule.

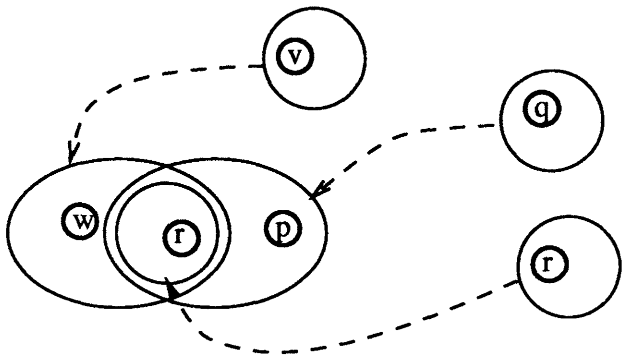

Figure 2.14: The profile of the pin-graph G_τ

Theorem's proof

We use Lemma 2.1.

(1) $\Longrightarrow$ (2) (by contradiction)

The proof's assumption is then: $(\{O_i \ / \ 1 \leq i \leq n\}, \subset)$ is not a well-ordering. Therefore, there exist i and j such that $O_i \not\subset O_j$ and $O_j \not\subset O_i$. Let us consider $(y_i, b_i) \in O_i \setminus O_j$ and $(y_j, b_j) \in O_j \setminus O_i$. There exists $(a_i, x_i) \in I_i$ such that $x_i \not\leq_G y_j$, and there exists $(a_j, x_j) \in I_j$ such that $x_j \not\leq_G y_i$. The couple (x_i, y_i), (x_j, y_j) of input-oupt pairs is a bad one; it contradicts Statement 2 of Lemma 2.1; therefore; it contradicts 1.

(2) $\Longrightarrow$ (1) Let us consider (a, x_a), (b, x_b), (y_c, c) and (y_d, d), input or ouput pins of G such that $x_a \leq_G y_c$ and $x_b \leq_G y_d$. We are going to prove that they are necessarily good (good for Statement 2 of Lemma 2.1).

Let us suppose that the O_i are numbered in increasing order. There exists i such that $x_a \in I_i$ and there exists j such that $x_b \in I_j$. If $i = j$ the proof is finished. Now, let us assume that $i < j$. Then, $x_a \in I_i \implies y_c \in O_i$ and $x_b \in I_i \implies y_d \in O_j$; moreover, $i < j \implies O_i \subset O_j$ and, therefore, $x_b \leq_G y_c$.

□

2.3 Conclusion

This study provides a formalization of all the communicating composite processes: the pin-graph [Len86]. The pin-graph allows one to detect if it is possible to compute a static sequentialization that works in any context. A pin-graph associated with the considered process models both the internal dependences and the external communications of the process, forgetting the effective semantics of the process; i. e., what the process actually does.

If the pin-graph is a semi-granule then at least one good sequentialization exists. A profile can be associated with any pin-graph; there is an equivalence between an immediate property of the profile of a pin-graph and the property of being a semi-granule. The profile computation is equivalent to the reflexo-transitive closure computation [Lep88].

Unfortunately, the set of semi-granules is strictly included in the set of the pin-graphs. The purpose of the following chapter is to introduce what we can do for a pin-graph that is not a semi-granule.

Chapter 3

Partition and composition

As it is well-known, it is unfortunately not always possible to sequentialize statically a set of tasks. In this case, we propose a decomposition of the considered process into several Communicating Sequential Processes. At this stage, however, we still consider that we have only one processor.

The idea is to cut up the pin-graph, which is assumed not to be a semi-granule, into several semi-granules. Then, every sub-pin-graph, which is assumed now to be a semi-granule, is sequentialized. All of the so obtained sequential schedules can then be connected together in order to build a schedule of the whole process into several Communicating Sequential (sub)Processes.

3.1 Composition

Let us suppose the considered pin-graph is already cut up. Consequently, we have several semi-granules. The problem is to reinforce each of them, in order to obtain a 1-schedule for each such that the composition of all gives a schedule of the whole pin-graph. But, if we do not build good local 1-schedules, then circuits may appear in the composition! Before looking at this more precisely, we define formally the pin-graphs composition.

3.1.1 Pin-graphs composition

In order to give an as elegant as possible formal definition of pin-graphs composition we use these notations: let $G = (X, \Gamma, V, I, O)$ be a pin-graph; $\forall x \in X$ $I_G(x) = \{a \in V \ / \ (a, x) \in I\}$ and $O_G(x) = \{a \in V \ / \ (x, a) \in O\}$.

Definition 3.1 (pin-graphs composition) *Let* $G_1 = (X_1, \Gamma_1, V_1, I_1, O_1)$ *and* $G_2 = (X_2, \Gamma_2, V_2, I_2, O_2)$ *be two pin-graphs such that* $X_1 \cap X_2 = \emptyset$. *The* **composed pin-graph** $G = G_1.G_2$ *is the pin-graph* (X, Γ, V, I, O)*:*

$$\begin{aligned}
X &= X_1 \cup X_2; \\
\Gamma &= \quad \Gamma_1 \\
&\quad \cup \ \Gamma_2 \\
&\quad \cup \ \{(x_1, x_2) \in X_1 \times X_2 \ / \ I_{G_2}(x_2) \cap O_{G_1}(x_1) \neq \emptyset\} \\
&\quad \cup \ \{(x_2, x_1) \in X_2 \times X_1 \ / \ I_{G_1}(x_1) \cap O_{G_2}(x_2) \neq \emptyset\}; \\
V &= V_1 \cup V_2; \\
I &= I_1 \cup I_2; \\
O &= O_1 \cup O_2.
\end{aligned}$$

This defines the composition of two distinct pin-graphs (i. e., the intersection of their vertices set is empty). We can define the composition of two pin-graphs that share vertices as the composition of isomorphic and distinct copies of these pin-graphs; in other words, the labels of vertices are considered as to be local.

The composed pin-graph is made up by adding a new arc from any ouput y of one of the pin-graphs to any input x of the other such that x and y carry complementary pins: the input x carries, for example, the input-pin (a, x) and the ouput y carries the output-pin (y, a), thus, $\{a\} \in I_{G_i}(x) \cap O_{G_j}(y) \neq \emptyset$. Moreover, the pins a, in this example, "survives" the composition, and in fact, all the pins "survives" the composition.

The composition masks neither input, nor ouput.

The following figure shows $G_\iota.G_\tau$, the composed pin-graph of G_ι and G_τ: the additional arcs (y, v), (u, r) and (p, z) (in boldface) are provided by the composition itself.

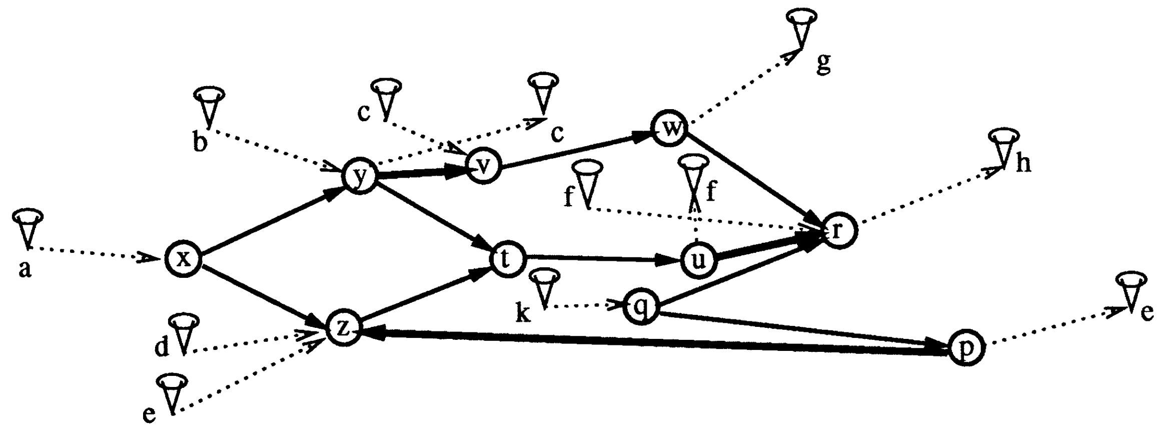

Figure 3.1: The pin-graph $G_\iota.G_\tau$

Corollary 3.1.1 *The pin-graphs composition is a commutative and associative internal operation. The empty pin-graph is the neutral element.*

3.1.2 Reinforcements composition

Let us consider the composition of pin-graphs. For each pin-graph, the remaining pin-graphs play the role of the context, where we consider context as in the previous section. Consequently, in the context of the composition of reinforcements, we require that local 1-schedules of pin-graphs be obtained from a circuit-consistent reinforcement. Thus, the following formal results come as a matter of course.

The first of them expresses it is possible to base the construction, by composition, of a schedule on the circuit-consistency property.

Proposition 3.1 *Let G be a circuit-free pin-graph and G_R be one of its reinforcements. The following properties are equivalent:*

1. *G_R is circuit-consistent;*
2. *for all circuit-free pin-graph G', if $G.G'$ is circuit-free, then $G_R.G'$ is a reinforcement of $G.G'$.*

Proof

(1) $\Longrightarrow$ (2) Let us consider a circuit-free pin-graph G' such that $G_R.G'$ includes a circuit, then it goes through G_R and G' alternatively (these pin-graphs are both circuit-free). Let us consider in this circuit a maximal path of G_R; let x be its first vertex and y be its last one, i. e., the vertex that precedes x is in G' and the one that follows y, too. So, $x \leq_{G_R} y$ and there exist (a,x), an input-pin of G, and (y,b), an output-pin of G. G_R is circuit-consistent; therefore, there exist (a,x'), an input-pin of G, and (y',b), an output-pin of G, such that $x' \leq_G y'$. In this way, we have substituted a path fully included in G for a portion of the circuit that belongs to G_R. By applying iteratively this method to every portion of the circuit that belongs to G_R, we construct a circuit of $G.G'$.

(2) $\Longrightarrow$ (1) Let (a,x) be an input-pin, and (y,b) be an output-pin of G, such that $x \leq_{G_R} y$. Let us consider the graph (see Fig. 3.2)

$$G_0 = (\{u\}, \{\}, \{a, b\}, \{(b, u)\}, \{(u, a)\})$$

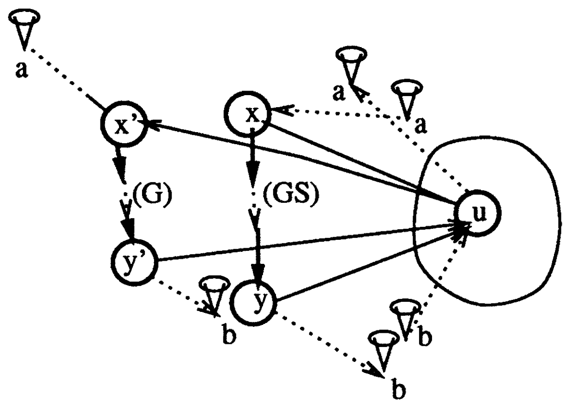

Figure 3.2: Circuits terminal behavior

The composition $G_R.G_0$ includes a circuit through x, y and u , in this order. Our proof assumption provides us with that property: there exists a circuit included in $G.G_0$, too. This circuit has to go through u. Therefore, there exist (a, x'), an input-pin of G, and (y', b), an output-pin of G, such that $x' \leq_G y'$. That is the definition of the circuit-consistency (see (2.1) of Definition 2.3).

□

That second result expresses it is possible to construct, by composition, circuit-consistent reinforcements, too.

Corollary 3.1.1 ***The composition of circuit-consistent reinforcements of two pin-graphs is a circuit-consistent reinforcement of the composition of these pin-graphs.***

Proof

Let G_1 and G_2 be two pin-graphs, and G_{R_1} and G_{R_2} be two respective circuit-consistent reinforcements. Two things are to be proved: the composition $G_{R_1}.G_{R_2}$ is a reinforcement, and it is circuit-consistent.

The proposition 3.1 allows one to conclude with no more development that $G_{R_1}.G_{R_2}$ is a reinforcement of $G_1.G_2$.

About the circuit-consistency: let G' be a circuit-free pin-graph; $(G_{R_1}.G_{R_2}).G'$ includes a circuit then (by associativity) $G_{R_1}.(G_{R_2}.G')$, too; G_{R_1} is circuit-consistent; therefore, $G_1.(G_{R_2}.G')$ includes a circuit. By using the same argument and the commutativity it is easy to conclude that $G_{R_2}.(G_1.G')$, $G_2.(G_1.G')$ and $(G_1.G_2).G'$ include circuits.

□

The last of them expresses that we can freely choose any reinforcement of a pin-graph, provided that it was circuit-consistent.

Corollary 3.1.2 *Let G_R be a circuit-consistent reinforcement of G, a circuit-free pin-graph, and G'_R be a reinforcement of G', a circuit-free pin-graph. The following properties are equivalent:*

1. *$G_R.G'_R$ is not a reinforcement of $G.G'$;*
2. *the composition of G'_R and a reinforcement of G cannot be a reinforcement of $G.G'$.*

Proof

(2) $\Longrightarrow$ (1) is immediate. So, let us prove (1) $\Longrightarrow$ (2). $G_R.G'_R$ is not a reinforcement; therefore, it includes a circuit. G_R is circuit-consistent; therefore, $G.G'_R$ includes also a circuit. Consequently, the composition of any reinforcement of G and G'_R includes a circuit. □

3.2 Scheduling by (re)composition

The situation we are considering is the one in which a pin-graph is not a semi-granule. This pin-graph must therefore be cut up into several semi-granules. The aim is to reinforce every semi-granule in order to obtain, by (re)composition, a global and sequential by pieces schedule of the whole pin-graph.

The previous section shows the circuit-consistency exactly befits the construction by (re)composition of schedules. This section is devoted to more technical and algorithmic aspects of the (re)composition. In first, let us look at what is a sequential by pieces schedules.

3.2.1 Scheduling into n-segments

Definition 3.2 (n-schedule) *Let G be a pin-graph, a* **n-schedule** *of G is a pair (G_R, Π) where:*
G_R is a reinforcement of G;
Π is a partition of G_R into n elementary paths, called **segments.**

Figures 2.5 and 2.6 show the 1-schedules $(G_{R_1}, \{X_\iota\})$ and $(G_{R_2}, \{X_\tau\})$ of G_ι and G_τ respectively. In G_{R_1}, the arc (y, z) has been added, thus the missing arcs (x, z) and (y, t) of G_ι are represented in G_{R_1} by the paths (x, y, z) and (y, z, t) respectively; in G_{R_2}, the arc (p, v) has been added, thus the missing arc (q, r) of G_τ is represented by the path (q, p, v, w, r) in G_{R_2}.

Let us suppose that the pin-graphs G_ι and G_τ have been obtained by cutting up the pin-graph $G_\iota.G_\tau$, shown by the following figure. Is $G_{R_1}.G_{R_2}$ a n-schedule of $G_\iota.G_\tau$?

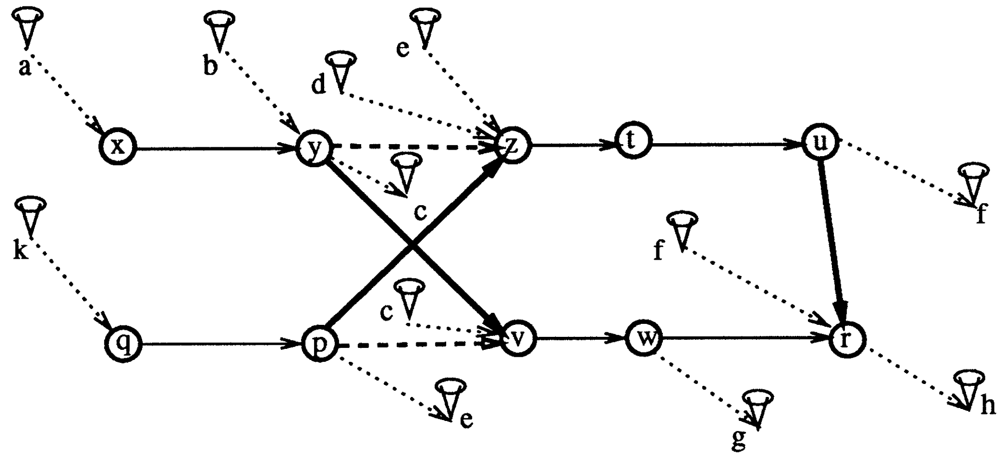

Figure 3.3: $G_{R_1}.G_{R_2}$ is a 2-schedule of $G_\iota.G_\tau$

This figure allows one to answer positively to the question; indeed, no circuit can be observed in $G_{R_1}.G_{R_2}$. G_{R_1} is a circuit-consistent reinforcement of G_ι: (z, y), the only G_ι's pair of communication, is therein shielded. That does not imply we could have chosen whichever other 1-schedule of G_τ in order to construct a 2-schedule of $G_\iota.G_\tau$. The fact that G_{R_1} is a circuit-consistent reinforcement provides us with only two things:

1. if there exists a circuit-consistent reinforcement of G_τ, the composition of this reinforcement and G_{R_1} constructs a circuit-consistent reinforcement of $G_\iota.G_\tau$ (corollary 3.1.1);
2. if there exists G_{R_3}, a reinforcement of G_τ, such that $G_{R_1}.G_{R_3}$ includes a circuit, the responsible is G_{R_3} and not, in any case, G_{R_1} (corollary 3.1.2).

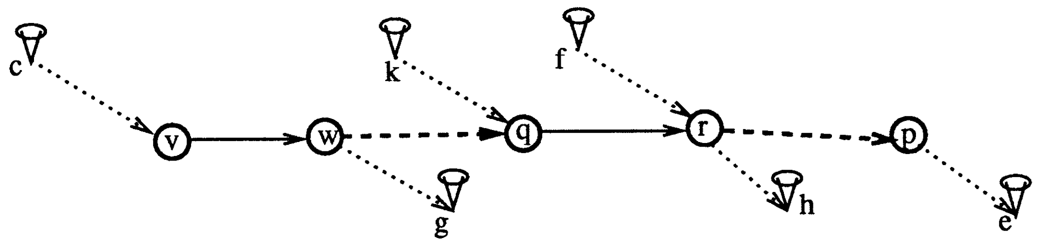

Figure 3.4: G_{R_3}: a 1-schedule of G_τ

Indeed, a reinforcement G_{R_3} exists (see Fig. 3.4). The composition of G_{R_3} and G_{R_1} includes effectively a circuit: (z, t, u, r, p, z).

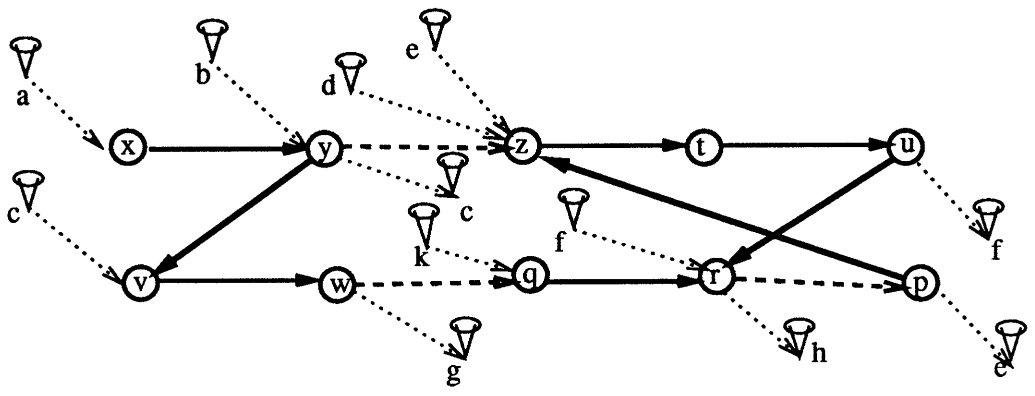

Figure 3.5: $G_{R_1}.G_{R_3}$ is not a schedule of $G_\iota.G_\tau$

3.2.2 A few technical aspects

Globally, the method we propose can be summarized by the following steps, in order:

1. compute a partition into semi-granules of the being treated pin-graph;
2. compute a circuit-consistent 1-schedule of each so obtained semi-granule;
3. compute the composition of the so-obtained 1-schedules, what provides with a n-schedule of the whole pin-graph (n being the cardinal of the partition).

Thus, the semi-granules are defined by a partition and have to be extracted from the initial pin-graph. We then need a notion of sub-pin-graph.

Definition 3.3 (sub-pin-graph) *Let $G = (X, \Gamma, V, I, O)$ be a pin-graph. $G' = (X', \Gamma', V', I', O')$ is called* **sub-pin-graph** *of G iff (X', Γ') is a directed sub-graph of (X, Γ) and*

$$\begin{aligned} V' &= \textstyle\bigcup_{x\in X'} O'_{G'}(x) \cup I'_{G'}(x), \\ I' &= \textstyle\bigcup_{x\in X'}(\{(a,x) \ / \ a \in I_G(x)\} \cup \{((y,x),x) \ / \ (y,x) \in \Gamma \ et \ y \in X \setminus X'\}), \\ O' &= \textstyle\bigcup_{x\in X'}(\{(x,a) \ / \ a \in O_G(x)\} \cup \{(x,(x,y)) \ / \ (x,y) \in \Gamma \ et \ y \in X \setminus X'\}). \end{aligned}$$

The following figure shows a sub-pin-graph of G_ι. A pin obtained from an arc break is, as it is described in the definition, labelled by the pair made up of the broken arc extremity labels.

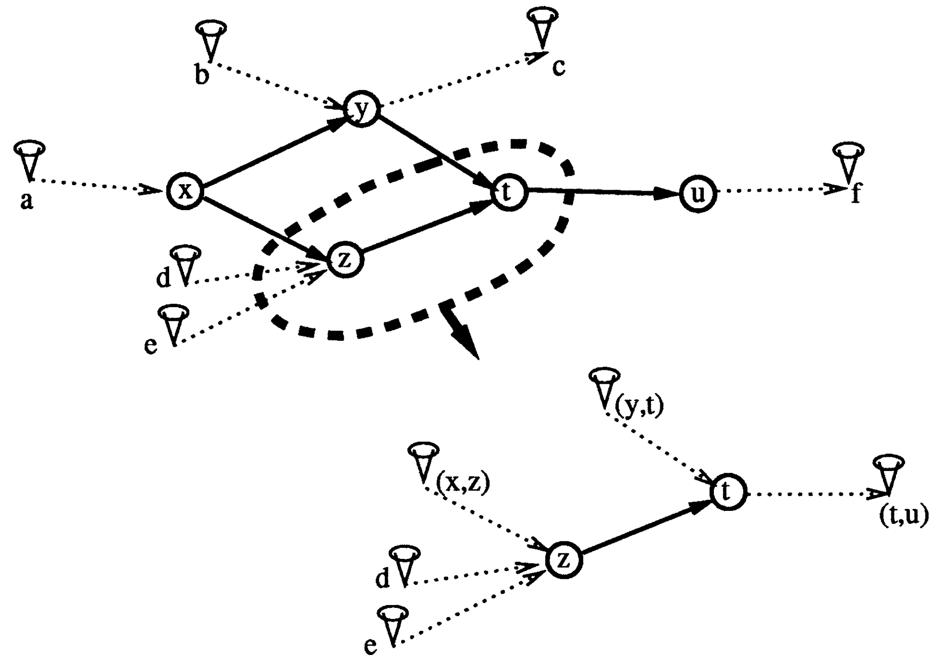

Figure 3.6: A sub-pin-graph of G_ι

An important property that the (de)composition-(re)composition process must have is to preserve the pin-graph, if nothing is performed but the (de)composition-(re)composition itself, of course. Unfortunately, as the following figures show so, some modifications may appear if no restriction is added to the definition of the pin-graphs.

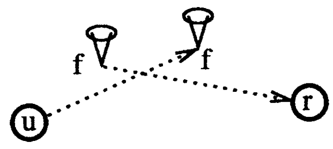

Figure 3.7: First singulary case of the (re)composition

Let us consider a partition of the pin-graph, shown in the previous figure, which has two parts such that u is in one and v is in the other. By (re)composing, we would create the new arc (u, r). The following condition must be added to the definition of the pin-gaphs in order to avoid this.

$$(u, f) \in O \ et \ (f, r) \in I \implies (u, r) \in \Gamma$$

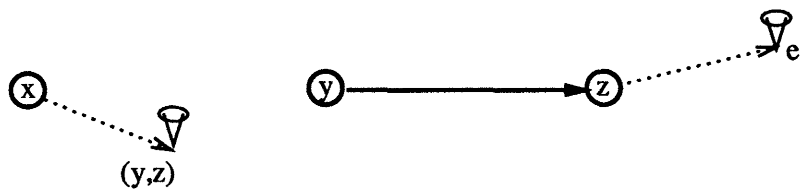

Figure 3.8: Second singulary case of the (re)composition

Let us now consider the case of this second figure. Let a partition be such that x and y are in a part and z in an other. The (de)composition would break the arc (y, z), consequently, the previous case would be created. So, the following condition must be added to the definition of pin-gaphs in order to avoid this.

$$((x, y), z) \in I \implies y = z \ \textit{and} \ (x, (y, z)) \in O \implies y = x$$

3.3 Algorithms

This section is an overview of the algorithms we are developing. Basically, two aspects are to be considered:

- how to compute the partition into semi-granules;
- how to compute the circuit-consistent schedules of the semi-granules.

An algorithm that splits the initial pin-graph, then splits the so obtained sub-pin-graphs, continuing iteratively, has been studied. The challenge is to compute a minimal partition. The principle is to compute sequences of pin-graphs that decreasingly converge to semi-granules. This algorithm is based on the second characterization of the semi-granules (see the 2 of Theorem 1). It uses the transitive closure; the complexity of the computation of the first semi-granules, i. e., without minimizing the partition, is polynomial, degree 4 [Lep88].

Let us now look at two others algorithms.

3.3.1 Barbedwires

The most sensitive aspect of the computation of circuit-consistent schedule is the shielding. Indeed, in order to schedule a pin-graph that has communication pairs care must be taken about the ordering of the elements of these pairs. In other words, the set of the circuit-consistent reinforcements of such a pin-graph is only strictely included in the set of reinforcements. In contrast, the set of circuit-consistent reinforcements of a pin-graph that does not include any communication pair is equal to the set of its reinforcements. This property makes possible a huge simplification of the computation of schedules of such pin-graphs, which are called barbedwires.

Definition 3.4 (barbedwire) *Let G be a circuit-free pin-graph, G is called a* **barbedwire** *iff it is equal to its own shielding graph.*

The following figure shows a barbedwire.

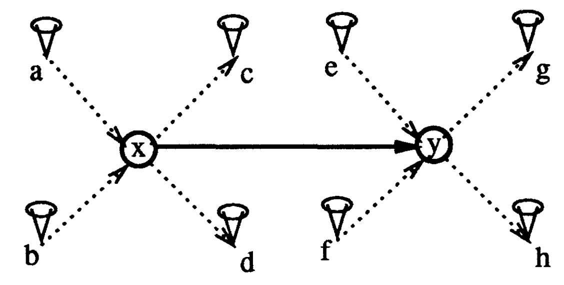

Figure 3.9: $G_{\aleph}$, an example of barbedwire

Corollary 3.4.1 *A barbedwire is a semi-granule.*

Having to add new arcs for scheduling statically states an important question. The reinforcement of constraints can generate execution delays that could be avoided by a dynamic scheduler. Let us consider the system modelled by the pin-graph G_{ι} (see Fig. 2.3). The sole circuit-consistent 1-schedule of G_{ι} is (x, y, z, t, u), which is shown by Figure 2.5. We know this schedule always works. But with a context that can provide d and e with their values before providing b with its, the static schedule above proposed will generate a delay, for waiting the value of b. On the other hand, a dynamic scheduler might have ordered the execution of the elementary task z; the time of the execution of z is maybe the time the context needs to provide b with its value.

Thus, the ideal partition we can consider seems to be a partition into barbedwires that are well-ordered: no reinforcement is done, the well-ordering is provided by the initial constraints of the pin-graph. In this case, if dead-locks and delays appear they are due to the incompatibility between the modelled application itself and the context, and nothing else. Of course, the following property is true.

Corollary 3.4.2 *An elementary path is a barbedwire.*

So, Berge's algorithm [Ber70] based on alternated strings, which is usually applied to compute the maximal matching of a bipartite graph, allows one to compute a partition into elementary paths. This algorithm is linear with respect to the number of arcs; Kuhn's theorem [Kuh55] states this partition is minimal with respect to the number of communications among the so obtained elementary paths.

3.3.2 Granules

An other class of the semi-granules is really interesting: the granules [LL88].

Definition 3.5 (granule) *Let G be a circuit-free pin-graph, G is called a* **granule** *iff* $COM(G) = \emptyset$.

Corollary 3.5.1 *A granule is a barbedwire*

Corollary 3.5.2 *Let G be a pin-graph and* $(\{I_i \ / \ 1 \leq i \leq n\}, \{O_i \ / \ 1 \leq i \leq n\})$ *be its profile, then the following properties are equivalent:*

1. *G is a granule;*
2. $n = 1$.

This definition says there is a path from any input to every output of a granule. This means if we consider a sub-pin-graph that is a granule then we can forget what is inside; knowing the inputs and the outputs is sufficient: the sub-pin-graph-granule can be considered like a regular vertex; a granule is a black-box. The granule is a useful tool for mapping onto multiprocessors [FGLL88].

The granule allows one to reduce a pin-graph without losing its connectivity.

A method of granulation, i. e., partition into granules, by condensation has been developed [Le 89]. It is a greedy algorithm which constructs a granule by absorbing the neighborhoud of the granule which is being constructed. The absorption is stopped, and a new granule is begun as soon as it is impossible to absorbe new neighbors without losing the property of being a granule. This algorithm is linear with respect to the number of arcs in the pin-graph.

3.4 Conclusion

Firstly, the semi-granules are presented, and it is shown that they constitute the class of the pin-graphs that admit a circuit-consistent 1-schedule; in other words, the semi-granules are the only ones that can be scheduled sequentially off-line without adding any constraint that could induce a dead-lock. Secondly, we describe a strategy of scheduling off-line, as much as possible, pin-graphs that are not semi-granules.

The formal definition of the composition of pin-graphs provides a way of stating precisely how semi-granules can be used for scheduling. It is proved that the circuit-consistency property is preserved by composition, and that a circuit-consistent scheduling is definitively safe, i. e., if a problem exists, then it comes from elsewhere. These properties make possible the computation of schedules by (re)composing local 1-schedules, which were previously computed by local investigations; these investigations are performed on the semi-granules associated with a partition of the whole pin-graph.

The point is then to compute these partitions. In a short overview, several algorithms were presented. Basically, the common thread is the research of sub-classes of the semi-granules such that a low complexity algorithm of partitionning into sub-pin-graphs of pin-graphs of the considered sub-class exists.

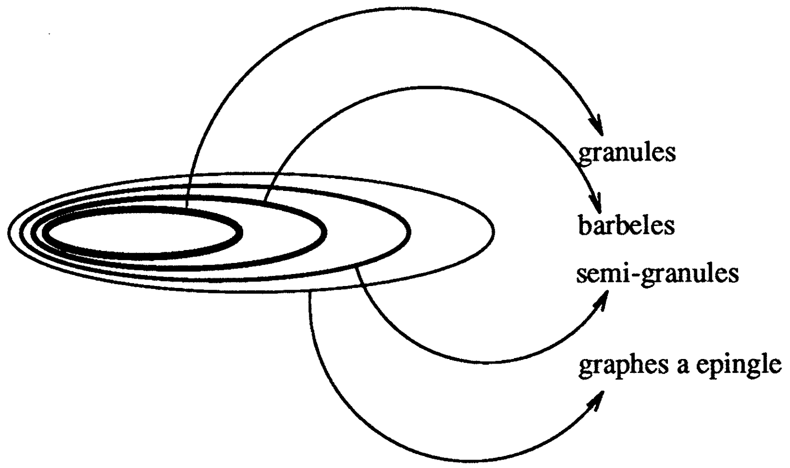

Figure 3.10: The semi-granules classification

Two sub-classes are really interesting (see Fig. 3.10): the barbedwire and the granule. Partitions into barbedwires answer to the question of delays during the execution due to the rigidity of static schedules. And the granule, which is truly a black-box, is the natural object for mapping onto multiprocessors.

Chapter 4

General conclusion

In the general framework of the implementation of real-time applications and dynamic systems, we propose methods for restructuring their specification [Le 89]. The purpose of these rewritings is to exhibit the information that is useful for implementation: the organization of the timing, of external communications and of internal dependences.

This could be done by emphazising the interesting information and/or by encapsulating the uninteresting information. The granule, which is briefly presented in this paper, is a good example of encapsulation; the interested reader is referred to Le Goff 89 [Le 89], Figueira 88 & al. [FGLL88] and Le Goff 88 & Le Guernic [LL88]. It is a natural base for a mapping method. The semi-granule, and its profile, which summerizes the dependences among communications, are, such as it is developed in this paper, a good example of synthesis of information. In this case, this information is used in order to compute schedules that are compatible with whichever context of execution.

The directed graph is a good model for a set of tasks that are related by dependences. Scheduling is expressed naturally in a directed graph. The external communications are represented by special marks which were introduced by Lengauer [Len86]. The so completed directed graph provides a way of defining formally the problem of dead-locks due to schedules. The semi-granules are the class of graphs that admit a sequential schedule compatible with whichever context of execution. The formalization of properties of the semi-granule represents an opportunity for interesting developments in graph theory.

The motivation of scheduling off-line is based on two things: the reduction of overhead, and the computation of deterministic run-schemes from parallel specifications [GIB89]. Semi-granulation, i. e., the computation of partitions into semi-granules, addresses the second aspect. If the considered application is already a semi-granule then a deterministic run-scheme exists. Otherwise, a semi-granulation reveals the fragile spots of the run-scheme. On the other hand, although a static schedule can reduce the overhead, because it precludes the use of a dynamic scheduler, its rigidity may generate numerous delays due to communication during the execution. These delays occur because we add artificial constraints to well-order the tasks. However, let us suppose that the static schedule does not include anything but the initial constraints of the application. In this case, any delay that occurs would be due entirely to the application itself. The partitions into barbedwires, which are formally defined, provide a way of getting such static schedules.

Acknowledgments

The authors would like thank Professor Jean-Claude Bermond and Doctor Ken Chou.

Bibliography

[Ber70] Claude Berge. **Graphes et hypergraphes**, volume 37 of **Monographie universitaire de mathématique**. DUNOD, 92, rue Bonaparte, Paris 6, France, 1970.

[BIM88] B. Bloom, S. Istrail, and A. R. Meyer. Bisimulation can't be traced: Preliminary report. In **POPL'88, Symposium on Principles of Programming Languages**, pages 229–239, San Diego, California, January 1988. ACM.

[BLL88] Albert Benveniste, Bernard Le Goff, and Paul Le Guernic. Hybrid dynamical systems theory and the language SIGNAL. Research Report 838, INRIA, Rocquencourt, April 1988.

[FGLL88] Carlos Figueira, Thierry Gautier, Bernard Le Goff, and Paul Le Guernic. Towards multiprocessor implementation of real-time dataflow programs. In W. W. Wadge, editor, **ISLIP'88: International Symposium on LUCID and Intensional Programming**, pages 127–138, Sidney, Canada, April 1988. University of Victoria.

[GIB89] E. W. Giering III and T. P. Baker. Toward the deterministic scheduling of ADA tasks. In **Real-Time Systems Symposium, IEEE**, pages 31–40. IEEE Computer Society Press, December 1989.

[GLB87] Thierry Gautier, Paul Le Guernic, and Loïc Besnard. SIGNAL: a declarative language for synchronous programming of real-time systems. In Gilles Kahn, editor, **Functional programming languages and computer architecture**, pages 257–277. Lecture Notes in Computer Science, 274, Springer-Verlag, 1987.

[Kuh55] H. W. Kuhn. The hungarian method for the assignement problem. Technical Report 2, Naval Research Logistics Quartely, 1955. 83.

[Le 89] Bernard Le Goff. **Inférence de contrôle hiérarchique : application au temps-réel.** PhD thesis, Université de Rennes 1, 1989.

[LL88] Bernard Le Goff and Paul Le Guernic. The granules, glutton: An idea, an algorithm to implement on multiprocessor. In R. Cori M. Wirsing, editor, **STACS 88, Lectures Notes in Computer Science**, Bordeaux France, February 1988. AFCET, Springer-Verlag. Volume 294.

[Len86] T. Lengauer. Efficient algorithms for finding minimum spanning forests of hierarchically defined graph. In **STACS 86, Lectures Notes in Computer Science**, pages 153–170, USA, 1986. Springer-Verlag. Volume 210.

[Lep88] Hugue Lepoittevin. Réduction d'un graphe d'ordonnancement par partitionnement en granules, pour une implantation répartie. Rapport de stage DEA informatique encadré par B. Le Goff, IRISA, Rennes, 1988.

Index