Coordinating Multi-task Environments through the Methodology of Relations Graph

Adailton A. Cruz [1], Léo P. Magalhães [2], Alberto B. Raposo [3], Rafael S. Mendes [2], Dennis G. Pelluzi [2]

[1] Federal University of Dourados, Brazil
[2] Department of Computer Engineering and Industrial Automation, School of Electrical and Computer Engineering, State University of Campinas, Brazil
[3] Computer Science Department, Pontifical Catholic University of Rio de Janeiro, Brazil
adacruz@terra.com.br
{leopini, rafael, pelluzi}@dca.fee.unicamp.br
abraposo@tecgraf.puc-rio.br

Abstract. This paper presents Relations Graph – GR a methodology to automate the generation of coordination mechanisms in computational environments. GR explores encapsulation and compacting capabilities of Colored Petri Nets to generate temporal coordination mechanisms, although the use of the GR methodology does not depend on the knowledge of PN formalism. GR supports alternative temporal behaviors and alternative activities changing the temporal relations among activities in processing time. An algorithm to identify and model coordination mechanisms linear to the number of activities and its application to an illustrative collaborative authoring environment will be presented.

Keywords: Coordination, Petri nets, temporal behaviors modeling.

1 Introduction

Activities represent well-defined fragments within the general functioning of a process and involve in our context interdependency relations. A collaborative system may be viewed as a group of processes that are defined by a group of logically related activities. The execution of activities in collaborative environments may present conflicting requirements and interests that must be anticipated and solved in order to reach the desired goals. The effort aiming to foresee and solve the conflicts derived from dependencies among activities is defined as coordination [8] and it establishes the mechanisms that guarantee the (temporal, spatial, causal, etc.) behaviors defined by these interdependencies.

Our approach deals with dependencies considering both activities that are directly related and those indirectly related, facilitating the work of designers, which otherwise should identify and verify all the relationships – direct and indirect – for consistency.

Providing a solution for this issue is the motivation for this new methodology that enables the analytical and graphical representation of interdependency relations among activities in a computational environment. This methodology then generates coordination mechanisms that also model the global behavior, i.e., mechanisms that ensure the execution of all the interdependencies defined for the group of activities.

This way, the GR (Relation Graph) methodology considers the global and the local aspects of a coordination problem. The global approach is responsible for coordinating the necessary conditions to authorize the beginning of an activity. The local approach coordinates the execution of two authorized activities, complying with the kind of relation defined for them. Furthermore, this methodology does not restrict the quantity of relations in which an activity may be involved.

The coordination mechanisms are generated by an algorithm of complexity $O(n)$, n being the number of activities. This algorithm automates the identification and modeling of temporal restrictions among activities resulting in a coordination mechanism with the following characteristics:

1. Adherence to restrictions: the execution of an activity never violates any temporal restriction.
2. Behavior selection: it is possible to select different behaviors (activity interdependencies) for the same subset of activities.
3. Activity selection: it is possible to define whether a subset of activities will be executed or not.

The GR methodology does not gear a specific application. Its goal is to obtain coordination mechanisms considering the concepts of activity and interdependency.

Section 2 presents a discussion about coordination in computational environments. Section 3 introduces the GR methodology. Section 4 presents a case study. Conclusions are presented in Section 5.

2 Overview

The issue of activity coordination in collaborative environments has been the subject of varied scientific research [4], [6], [8], [10], [13] and has attracted the interest of experts in search for tools that help to coordinate computational environments.

This section briefly presents situations to illustrate the need for coordination models to couple with local and global aspects of computational environments.

A set of coordination mechanisms to manage temporal dependencies and resource dependencies among the participants of a collaborative environment was presented in [12]. For each participant, a Petri Net is designed, modeling the participant's activities and the dependencies among them. Then, possible dependencies among different participants are defined, concluding the first abstraction level, called by the author *workflow level*. Once the workflow level has been designed, the system's model is obtained automatically; in a second abstraction level, called *coordination level*, the activities, represented by transitions, are expanded according to a predefined model. The coordination mechanisms that correspond to the dependencies are added (they are also predefined), and the model of the collaborative system is obtained.

The separation between activities and dependencies, and the use of predefined coordination mechanisms provide the advantage of automatically generating the system's model based on the workflow level. The designer can focus only on specifying the system at the workflow level. The coordination mechanisms presented by [12] to temporal dependencies allow relating the activities two-by-two, thus ensuring the observance of temporal restrictions derived from this dependency. A global analysis and a verification of temporal inconsistencies (temporal behaviors that cannot be executed) must be made by the system designer.

In workflow applications, the cooperation among the different actors (human, computer, organization, etc.) in charge of executing the activities that define the processes must be promoted. A powerful coordination component is required by the dependency relation the activities usually present [1]. Dealing with possible conflicts among activities becomes more complex when the workflow involves multiple organizations [12]. An activity-based model using high-level Petri Nets (colored, temporized and hierarchical) to model both the workflow and the workflow's coordination system is proposed in [15]. Some of the difficulties found in the coordination and development of multi-organizational workflows are exposed in [16], which introduced a tool based on analysis techniques for Petri Nets to inspect multi-organizational workflows. The processes are specified in an XML-based language that is subsequently transformed into a specific Petri Net to verify if the workflow is correct.

In a multi-agent environment, the execution of tasks might affect or be affected by other tasks, which characterizes a dependency relationship among them. If the tasks refer to different agents, then the relationships among them represent a nature here called *non-local dependency*. This configuration restricts the agent's ability to select adequate actions, because the agent is not aware of restrictions derived from non-local dependencies [3], which motivates the use of coordination mechanisms for this purpose. Methodologies based on high-level Petri Nets, particularly colored ones, used in the problem of coordinating the behaviors of agents, have been presented by [9] and [17].

The dependency class (causal, resource, simultaneity, prerequisite, producer-consumer, etc.) among activities varies according to the context of the problem to be modeled, but a common denominator among these classes seems to be the time factor. Thus, temporal models are essential to express dependencies among activities in order to support applications in different research fields [18], [19].

The following section introduces a new methodology to model coordination mechanisms. The concept of coordination will be further developed, because apart from (identification/association) specification, a linear-cost algorithm will be introduced for the automatic generation of coordination mechanisms based on the specification of temporal behaviors.

3 Methodology

The GR methodology consists in generating coordination mechanisms based on the temporal behaviors specified for the activities executed in the environment. These

behaviors correspond to temporal dependency relations among the activities.

Three abstraction levels are defined (Fig. 1a). The *specification level* establishes a temporal order among the activities through an expression. At the *coordination level*, the coordination mechanism is built after modeling all dependencies among the activities described in the previous level. At the *execution level*, a program called *coordinator* will implement the coordination mechanism obtained in the coordination level. This coordinator interacts with the set of activities, obeying the specification made at the specification level.

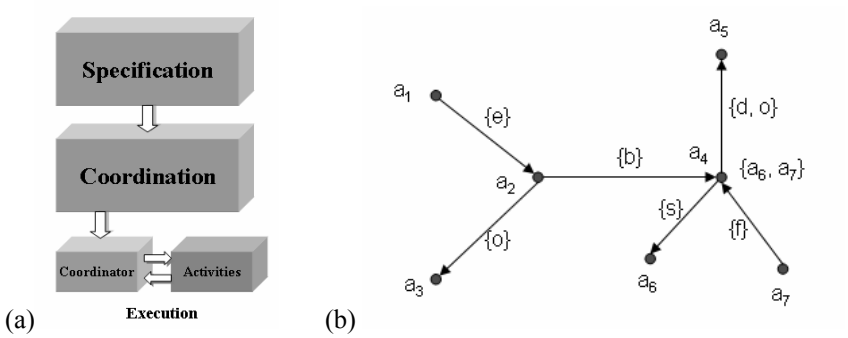


Fig. 1. (a) Abstraction levels. (b) Graph of temporal behaviors of an application.

At the more abstract level temporal behaviors are specified by an activity-based model; it has as essential element non-null intervals. These intervals contain the activities executed in the system, which on their turn may establish relationships among themselves through temporal primitives [7], [19]. The set of temporal primitives introduced by [2] was adopted as the possible temporal relationships among activities.

Table 1 presents seven temporal primitives relations between two activities **a** and **b**, occurring in time intervals x = [a_i, a_f) and y = [b_i, b_f), with a_i and a_f, b_i and b_f being the beginning and the end of **a** and **b**, respectively, and the relations according to the temporal primitives of the set **D** = {e, s, d, f, o, m, b},

Table 1. Temporal primitive relations between activities **a** and **b**.

e(a, b): equal	↔	$a_i = b_i$ and $a_f = b_f$ **- a** is executed in the same time interval as **b**.
s(a, b): start	↔	$a_i = b_i$ and $a_f < b_f$ **- a** and **b** begin together, but **a** ends before **b**.
d(a, b): during	↔	$a_i > b_i$ and $a_f < b_f$ **- a** begins after **b** and ends before **b.**
f(a, b): finish	↔	$a_i > b_i$ and $a_f = b_f$ **- a** begins after **b**, but **a** and **b** end together.
o(a, b): overlap	↔	$a_i < b_i$ and $b_i < a_f$ and $a_f < b_f$ **- a** begins before **b**, which begins before **a** is over. **a** ends before **b**.
m(a, b): meet	↔	$a_f = b_i$ **- b** is executed immediately after the end of **a**.
b(a, b): before	↔	$a_f < b_i$ **- a** must be executed before **b**.

In the GR model, temporal behaviors are represented by a labeled oriented graph whose vertices correspond to system activities and edges correspond to the dependency relations among them. Fig. 1b illustrates a hypothetical example with 7 activities. The direction of the edges indicates the order in which the activities are related,

and their labels state the type of dependency among them. For example, the label of edge $\mathbf{a_2,a_4}$ – {b} – specifies that activity $\mathbf{a_2}$ must be executed before activity $\mathbf{a_4}$. The label in edge $\mathbf{a_4,a_5}$ specifies alternative dependency relations, which means that any one of the alternatives may take place each time these activities are activated by the application. The selection of one relation instead of the other is made in execution time, according to environment requirements.

Still at specification level, alternative activities can be anticipated; for instance, the label of activity $\mathbf{a_4}$ indicates that one of the activities $\mathbf{a_6}$ or $\mathbf{a_7}$ must be selected to be related with $\mathbf{a_4}$. This selection occurs each time activity $\mathbf{a_4}$ is activated through events fired by the application, and any of the activities in its label can be selected.

The specification of a system S is stated by an expression E(A,R,F), an oriented and labeled[1] graph, with an acyclic subjacent[2] graph, where:

- A is a set of vertices representing the activities,
- $R \subset A \times A$ is a set of edges representing the relations,
- F is a function $F : R \rightarrow \wp(D)$ where $\wp(D)$ is the set of D parts, called edge (relation) labeling function – D={e, s, d, f, o, m, b}, and
- G is a function $G : A \rightarrow \wp(A)$ where $\wp(A)$ is the set of A parts, called vertex (activity) labeling function, which satisfies the following properties:

1. If $\mathbf{b} \in \mathbf{G(a)}$, then the edge defined by **a** and **b** belongs to **R**;
2. If $\mathbf{b} \in \mathbf{G(a)}$, then $\mathbf{a} \notin \mathbf{G(b)}$;
3. If $\mathbf{b} \in \mathbf{G(a)}$, then $\exists\ \mathbf{c} \in \mathbf{G(a)} \mid \mathbf{c} \neq \mathbf{b}$.

Applying the above definitions to the example in Fig. 1b, the following qualified expression (E, G)[3] is obtained:

$\mathbf{A}=\{a_1, a_2, a_3, a_4, a_5, a_6, a_7\}$;
$\mathbf{R}=\{(a_1,a_2), (a_2,a_3), (a_2,a_4), (a_4,a_6), (a_7,a_4), (a_4,a_5)\}$;

The edge labeling function **F** is given by $\mathbf{F}(a_1,a_2)=\{e\}$, $\mathbf{F}(a_2,a_3)=\{o\}$, $\mathbf{F}(a_2,a_4)=\{b\}$, $\mathbf{F}(a_4,a_6)=\{s\}$, $\mathbf{F}(a_7,a_4)=\{f\}$, and $\mathbf{F}(a_4,a_5)=\{d, o\}$, and the vertex labeling function **G** is given by:

$\mathbf{G}(a_4)=\{a_6,a_7\}$ and $\mathbf{G}(a_1)= \mathbf{G}(a_2)= \mathbf{G}(a_3)= \mathbf{G}(a_5)= \mathbf{G}(a_6)= \mathbf{G}(a_7)=\varnothing$

Once the temporal ordering among the activities is known, construction of the co-ordination mechanism begins. This process corresponds to modeling the temporal constraints derived from the dependency relations among the activities.

---

1 A graph is called node-(or edge-) labeled when to each vertex (or edge) there is an associated set, called label [14]. In this text, we consider node- and edge-labeled graphs, which we call simply labeled graphs.

2 The subjacent graph is the one obtained from the oriented graph by removing the directions of the edges [14].

3 A qualified expression is a pair (E,G) where E is an expression E(A,R,F) and G is the function defined above.

3.1 Abstract Levels and Coordination Mechanisms

Fig. 2a shows the diagram – a colored Petri Net – of an activity **a** at coordination level.

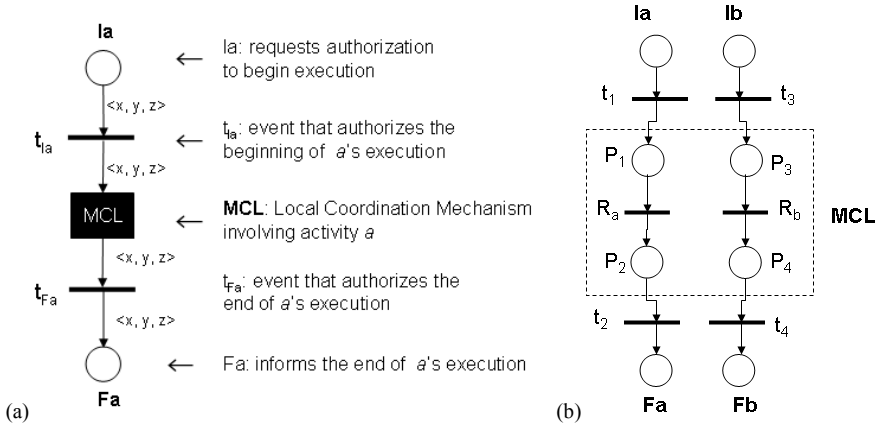


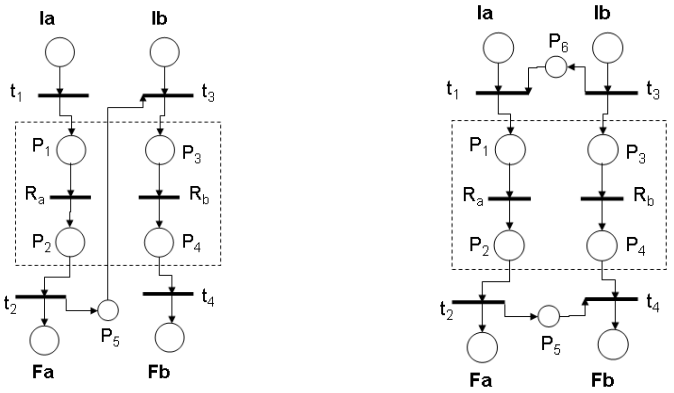
Fig. 2. (a) Diagram of an activity at coordination level. (b) Colored Petri Net of r(a,b)

The beginning of activity **a** is authorized by the event generated by firing transition $\mathbf{t_{Ia}}$; firing transition $\mathbf{t_{Fa}}$ generates the end of execution of activity **a**. The MCL - Local Coordination Mechanism - commands the beginning of **a** and waits for the end of **a**'s execution. The ordered triple **<x , y , z>** represents the colored token so that **(y,z) ∈ R** and **x ∈ F(y,z)**, where **R** is the set of edges of an expression and **F** is the edge labeling function.

The diagram in Fig. 2b details the MCL for the relation between activities **a** and **b;** activity **a** is related to activity **b** through relation **r ∈ D – r(a,b)**. Firing transition $\mathbf{t_1}$ ($\mathbf{t_3}$) corresponds to the event that authorizes the beginning of execution of activity **a** (**b**), while firing transition $\mathbf{t_2}$ ($\mathbf{t_4}$) corresponds to the event that authorizes the end of execution of activity **a** (**b**). The time spent in the execution of an activity is represented using the concept of transition with token reservation [11]. In this type of transition, the firing takes place in two moments. Firstly the tokens are removed from the input place when the transition is active; secondly the tokens are sent to the output place after a given time span.

Fig. 3 illustrates the relations before(a,b), or b(a,b), and during(a,b), or d(a,b), derived from Fig. 2b.

The GR methodology also allows the specification of alternative temporal behaviors and the anticipation of alternative activities.



(a) MC of relation **before: b(a, b)** **(b) MC** of relation **during: d(a, b)**

Fig. 3. Examples of basic MCs.

Fig. 4 shows the case in which alternative temporal relations must be anticipated, at specification level, between the activity pairs that define such behaviors. The selection of one behavior instead of another is made in execution time, according to environment requirements. This mechanism, called basic MC – Coordination Mechanism – models all relations anticipated for activities **a** and **b** and allows only one of them to be selected at a time.

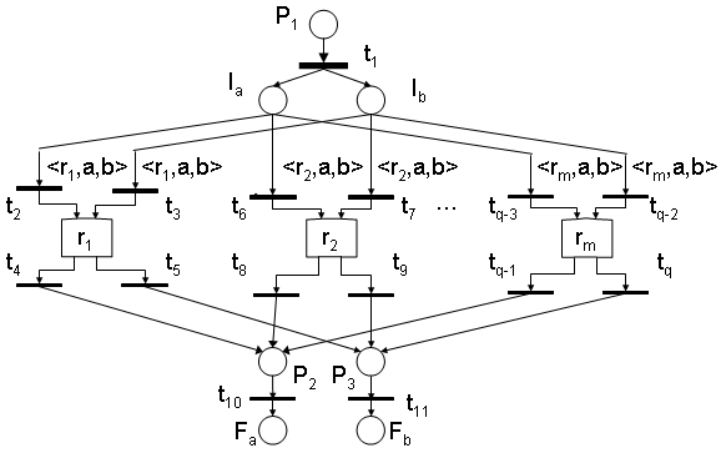


Fig. 4: Basic MC for alternative behaviors – relations.

Fig. 5 presents the basic mechanism that allows the anticipation of alternative activities in the specification of a temporal behavior. The alternative activities that take part in a given behavior are selected at execution time. This basic MC allows the selection of any one of the activities to be related with activity **a**. Such selection is made according to the token color at places **Ia** and $\mathbf{Ib_i}$ (i={1, 2, ... m}).

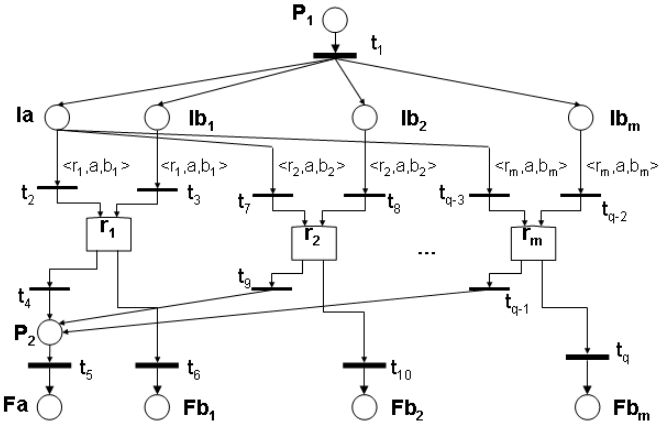


Fig. 5: Basic MC for alternative behaviors – activities.

Following, the construction of coordination mechanisms of an expression with more than one dependency relation will be derived from the basics MCs. The construction procedure is based on the connection of pairs of coordination mechanisms (**MC$_1$, MC$_2$**) with a common activity, i.e. the same activity modeled both in **MC$_1$** and in **MC$_2$**. Figs. 6 (a) and (b) illustrate the mechanisms corresponding to *before* and *during* relations, respectively, **b** being the common activity.

Transitions **t$_2$** and **t$_5$** (**t$_4$** and **t$_7$**) in Figs. 6 (a) and 6 (b), respectively, are associated to the beginning (end) of **b**'s execution. Input arcs to these transitions correspond to the temporal restrictions involving **b**; firing **t$_2$** (**t$_4$**) in **MC$_1$** and **t$_5$** (**t$_7$**) in **MC$_2$** indicates that all temporal restrictions involving **b** were satisfied, resulting in the events that authorize the beginning (end) of their execution in the respective mechanisms. To keep the synchronization condition regarding the beginning of **b**'s execution, in the mechanism resulting from the connection between **MC$_1$** and **MC$_2$** this event must be generated by a single transition. This synchronization condition can be satisfied by merging transitions **t$_2$** and **t$_5$** (**t$_4$** and **t$_7$**), generating a single transition that receives all temporal restrictions involving the beginning (end) of **b**. Fig. 6 (c) illustrates the **MC** resulting from the connection operation between **MC$_1$** and **MC$_2$**.

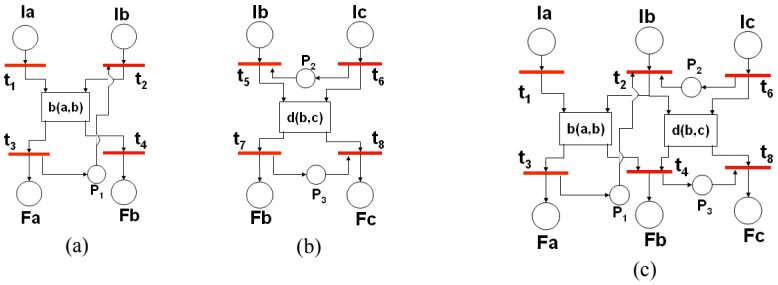


Fig. 6 (a) MC$_1$: relation *before* **b(a,b);** (b) MC$_2$: relation *during* **d(b,c);** (c) MC resulting from the connection operation between MC$_1$ and MC$_2$.

The **merging of** two transitions t' and t" is executed by transferring to one of them all the input and output arcs of the other.

The extension of basic MCs for alternative behaviors consists in inserting a feature, here called connection feature. Fig. 7 presents the basic MC for alternative dependency relations with the connection feature for activity **a**. This extension, identified by traced arcs, splits the transitions (see definition below) $\mathbf{t_2}$, $\mathbf{t_6}$, ··· , $\mathbf{t_{q-3}}$ from Fig. 5 into the transitions $\mathbf{t_2'}$ and $\mathbf{t_2''}$, $\mathbf{t_6'}$ and $\mathbf{t_6''}$, ··· , $\mathbf{t_{q-3}'}$ and $\mathbf{t_{q-3}''}$ in Fig. 7, respectively, and adds places $\mathbf{P_4}$ and $\mathbf{P_5}$ and transition $\mathbf{t_N}$. The temporal restrictions derived from relations $\mathbf{r_1}$, $\mathbf{r_2}$, ··· , $\mathbf{r_m}$ that must be imposed on activity **a** correspond to the input arcs of transitions $\mathbf{t_2'}$, $\mathbf{t_6'}$, ··· , $\mathbf{t_{q-3}'}$ (Fig. 7). These restrictions are not represented here because relations $\mathbf{r_1}$, $\mathbf{r_2}$ ··· , $\mathbf{r_m}$ are generic.

Splitting transition t into transitions t' and t" consists in attributing all input arcs of t to t' and all output arcs of t to t".

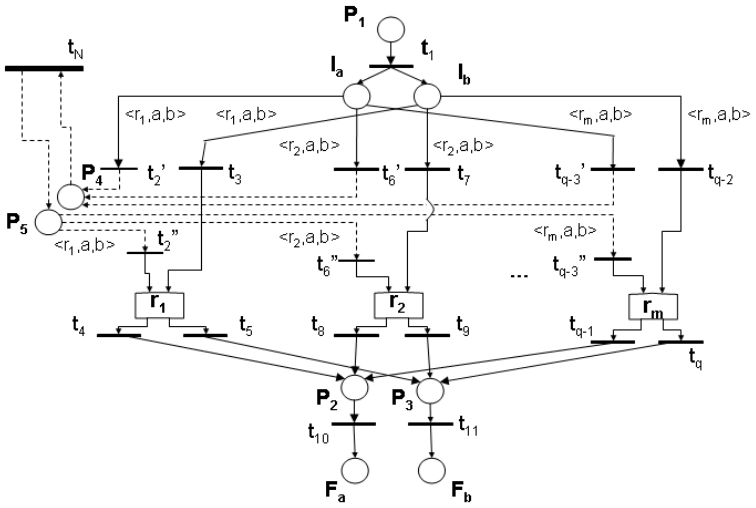


Fig. 7: Basic MC with connection features for alternative relations.

The extension of the coordination mechanism from one dependency to two or more dependencies can be reached performing connections between MCs as follows.

Considering Fig. 7, to perform the connection operation between the basic MC for alternative dependency relations and another MC, a **merge** must be made between transition $\mathbf{t_N}$ and a transition from the other MC that generates the authorization event to begin activity **a**, as well as a merge between transition $\mathbf{t_{10}}$ and a transition from the other MC that generates the authorization event to end activity **a**, considering **a** as the common activity to these mechanisms.

The extension of basic MCs for alternative activities is completely analogous to that of basic MCs for alternative relations. Fig. 8 presents this basic MC with the connection feature to activity **a**. The temporal restrictions derived from relations $\mathbf{r_1}$, $\mathbf{r_2}$, ··· , $\mathbf{r_m}$ that must be imposed on activity **a** correspond to the input arcs of transitions $\mathbf{t_2'}$, $\mathbf{t_7'}$ ··· , $\mathbf{t_{q-3}'}$. Firing one of the transitions indicates that the temporal restrictions derived

from one of the alternative relations was fulfilled, which allows transition $\mathbf{t_2}$**"** whose firing generates the authorization event to begin execution of activity **a**.

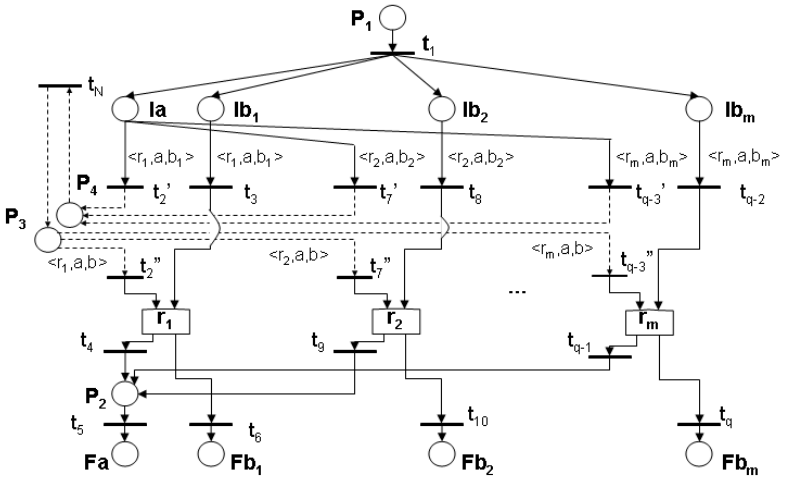


Fig. 8: Basic MC with connection features for alternative behaviors – activities.

3. Algorithm for the Coordination Mechanism

Now that the basic mechanisms have been introduced, we can construct the coordination mechanism of an expression through connection operations among them. A possible approach is to build the mechanisms corresponding to each edge and each vertex of the relation graph according to their labels and then make the connections according to the graph's adjacencies. The steps of the algorithm are listed below.

```
Given an expression ε[4]
Obtain Partition of A: P(A) = {I0, I1, …, Iu}
For k ← 1 step 1 until (number of elements of P(A)) do
  Step 1. Select set Ik of activities ai;
  Step 2. Identify and model the list of direct re-
strictions for each ai-star, ai ∈ Ik;
  Step 3. Connect the MC of ai-stars, ai ∈ Ik with the
proper MC of aj-stars, aj ∈ Ik-1;
End for
If center Iu from ε has two activities ap and aq
  then connect MCap with MCaq;
End algorithm
```

In step 1 of the algorithm, the selection of set $\mathbf{I_k}$ of the current iteration is made based on the definition of sub expressions of **ε** and the partition of set **A**. Although the identification process can be initiated by any set of activities, it is interesting to define an activity selection order. This provides advantages to the identification process of global conditions, because the fact that an activity $\mathbf{a_i}$ is a leaf (an activity with only

4 ε denotes the expressions **E(A, R, F)** and **(E, G)**.

one dependency relation $\partial(a_i)^5 = 1$) or is internal (an activity with more than one dependency relation $\partial(a_i) > 1$) can be known beforehand and used to simplify the modeling process.

To determine this order, given an expression ε, a partition of activity set **A** is obtained, denoted by **P(A)**. The subsets that determine **P(A)** are established through a sequence of expressions $\{\varepsilon_i\}$, $i = 0,1,2,...,u$, obtained from the original expression ε, according to the formation rule in Table 2.

Table 2. Formation Rule

ε_i	Formation Rule
ε_0	is the original expression ε;
ε_1	Expression derived from ε_0, eliminating the activities with degree 1 from ε_0;
ε_2	Expression derived from ε_1, eliminating the activities with degree 1 from ε_1;
⋮	⋮
ε_u	Expression derived from ε_{u-1}, eliminating the activities with degree 1 from ε_{u-1}.

Sequence $\{\varepsilon_i\}$ has maximum size $u+1$, smaller than or equal to half the number of activities **n** involved in an expression ε, i.e. $u \leq n/2 - 1$. The largest value of **u** occurs in linear expressions.

A partition **P** of activity set **A** of an expression ε is given by $P(A) = \{I_0, I_1,...,I_u\}$, where $u \leq n/2 - 1$, **n** being the number of activities and

$$I_0 = \{a_i \in \varepsilon_0 \mid \partial(a_i) = 1\}$$
$$I_1 = \{a_i \in \varepsilon_1 \mid \partial(a_i) = 1\}$$
$$I_2 = \{a_i \in \varepsilon_2 \mid \partial(a_i) = 1\}$$
$$\vdots$$
$$I_u = \{a_i \in \varepsilon_u \mid \partial(a_i) = 1\}$$

The purpose of sequence $\{\varepsilon_i\}$ is to formalize the partition criterion for **A**. In fact, it is neither necessary nor convenient to effectively create $\{\varepsilon_i\}$. Computing a list containing the degree of all activities is sufficient. Thus, I_0 is determined by selecting the a_i activities in the list such that $\partial(a_i) = 1$. To determine I_1, the list must be updated. For each activity a_i from I_0 : i) remove a_i from the list; ii) subtract one unit from the degree of activity a_j related to a_i. Then I_1 is determined by all a_j activities, $j \neq i$ such that $\partial(a_j) = 1$. This procedure ends with a list that has only one or two activities, i.e. the center of the expression, which corresponds to set I_u.

Once the partition criterion has been established, without losing generality an outside-in selection order is determined, i.e. from external activities (set I_1) towards internal ones (I_2 , I_3 ,..., I_u). An advantage of this selection order is to begin by I_1 rather

5 $\partial(a_i)$ denotes activity degree, i.e., the number of activities a_j ($j \neq i$) related to a_i.

than by $\mathbf{I_0}$. $\mathbf{I_0}$'s temporal restrictions can be determined when $\mathbf{I_1}$'s activities are processed.

Step 2 of the algorithm determines for each $\mathbf{a_i}$ the list of direct restrictions. This requires knowing the activity set $\mathbf{a_j}$ related to it.

An $\mathbf{a_i}$**-star** is a sub-expression determined by a_i, by the activity set a_j related to a_i and by its respective relations.

Analyzing an $\mathbf{a_i}$**-star** means determining the set of direct restrictions activity $\mathbf{a_i}$ must satisfy, called $\mathbf{a_i}$'s direct restriction list. To do this, consider **A** and **B** as two sets of temporal restrictions and the following conventions:

1. **A .and. B** is the set formed by all temporal restrictions of **A** and **B**, read as **A** conjunction **B**;
2. **A .or. B** is the set formed by all temporal restrictions of exclusively **A** or **B**, read as **A** disjunction **B**.

The direct restrictions of the $\mathbf{a_i}$**-star** are the temporal restrictions derived from all relations involving $\mathbf{a_i}$ and formed by the conjunction of $\mathbf{C_1}$, $\mathbf{C_2}$ and $\mathbf{C_3}$, where:

$\mathbf{C_1}$ is the conjunction of the restriction sets derived from the relations between $\mathbf{a_i}$ and the $\mathbf{a_j}$ activities that satisfy the following properties:

1. $\mathbf{a_j} \notin G(ai)$;
2. the label of the edge defined by a_i and a_j is unitary (has one primitive).

$\mathbf{C_2}$ is the disjunction of the restriction sets derived from the relations between $\mathbf{a_i}$ and the $\mathbf{a_j}$ activities such that:

1. $a_j \in G(a_i)$.

$\mathbf{C_3}$ is the conjunction of $\mathbf{C_{ij}}$'s where each $\mathbf{C_{ij}}$ is the disjunction of the restriction sets derived from the relations of the edge's label, defined by $\mathbf{a_i}$ and $\mathbf{a_j}$ provided that $\mathbf{a_j}$ satisfies the following properties:

1. $a_j \notin G(a_i)$;
2. the label of the edge defined by a_i and a_j is not unitary.

Illustrating the algorithm using the example in Fig. 1b, we have:

1. Considering the expression presented in Fig. 1b: $\mathbf{a_1}$, $\mathbf{a_3}$, $\mathbf{a_5}$, $\mathbf{a_6}$, $\mathbf{a_7}$ are degree 1 activities.
2. Following the outside-in order, first the basic MCs must be built for the labels of edges ($\mathbf{a_1}$, $\mathbf{a_2}$) and ($\mathbf{a_2}$, $\mathbf{a_3}$), and the connection operation between these two basic MCs must be executed.
3. Then the basic MC for edge ($\mathbf{a_4}$, $\mathbf{a_5}$) and the basic MC for the label of activity a_4 are built, and the connection operation between these two basic MCs is executed.

4. To determine the next basic MCs to be built, degree 1 activities are removed from the graph; the resulting graph will display new degree 1 activities.
5. Basic MCs must be built for these edges involving degree 1 activities, and connection operations must be executed according to the graph's adjacencies.

In the next section the GR methodology will be explored in a case study.

4 Case Study

In this section, we present a case approaching the use of the methodology presented in the previous section. The case study illustrates a situation involving a collaborative authoring environment.

The authoring tool allows an author to write and/or to review a paper's section. In this example, there are three authors (A, B and C). Each author writes one section. The section 2 is reviewed by author A and the section 3 is reviewed by either author B or author A. The temporal dependencies between the activities are showed in Fig. 9.

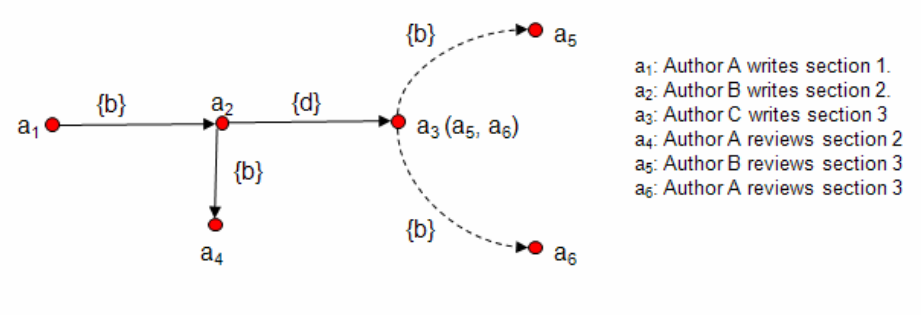


Fig. 9: Relation graph of the case study.

The relation graph showed in Fig. 9 is formally specified in the expression **E**(**A**, **R**, **F**) and in functions **G**, provided as follows.

A = $\{a_1, a_2, a_3, a_4, a_5, a_6\}$
R = $\{(a_1, a_2), (a_2, a_4), (a_2, a_3), (a_3, a_5), (a_3, a_6)\}$
F$(a_1, a_2) = \{b\}$, **F**$(a_2, a_4) = \{b\}$, **F**$(a_2, a_3) = \{d\}$, **F**$(a_3, a_5) = \{b\}$, **F**$(a_3, a_6) = \{b\}$
G$(a_3) = \{a_5, a_6\}$, others **G**$(a_i) = \varnothing$.

Now we construct the coordination mechanism using the algorithm described in section 3. The partition **P** of activity set **A** of the expression **E** is given by **P**(**A**) = $\{\mathbf{I_0}, \mathbf{I_1}\}$, where $\mathbf{I_0} = \{a_1, a_4, a_5, a_6\}$ and $\mathbf{I_1} = \{a_2, a_3\}$. According to the algorithm, we begin the construction of the coordination mechanism selecting the activities of set $\mathbf{I_1}$.

Fig. 10 illustrates the connection of MC of relations (a_1, a_2) and (a_2, a_4), which share a common activity (a_2), through merging the transitions t_{Ia2} and t_{Fa2}.

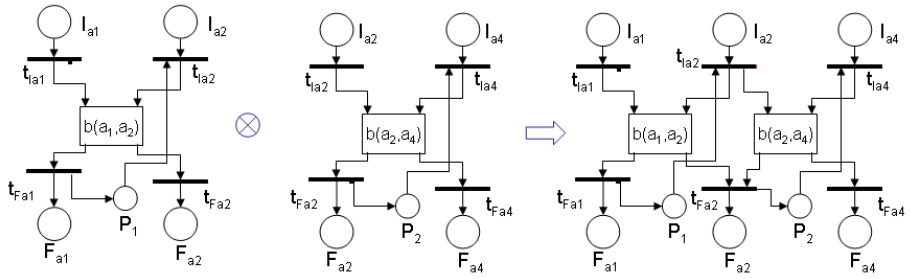


Fig. 10: Connecting MC of relations b(a_1, a_2) and b(a_2, a_4).

The same operation is done with the a_3 activity's relations. Finally, we connect the MC of relation (a_2, a_3) with the MC of activities a_2 and a_3. Fig. 11 presents the final coordination mechanism.

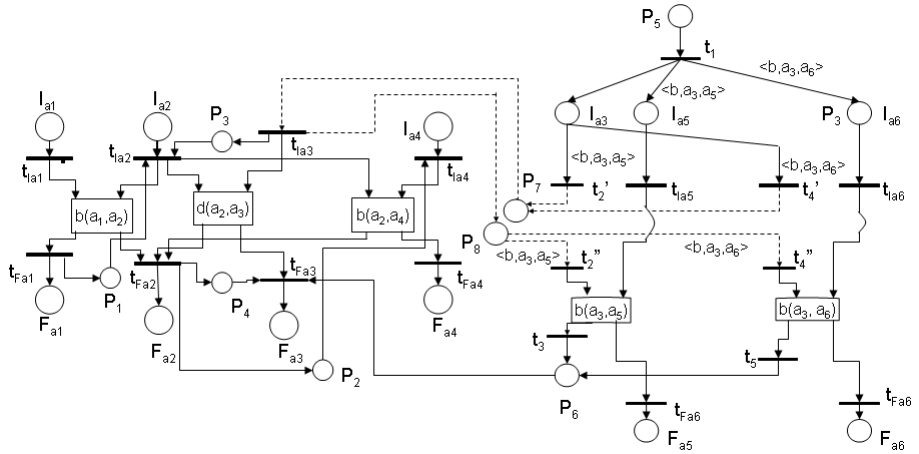


Fig. 11: Final Coordination Mechanism.

The selection of relation (a_3, a_5) or (a_3, a_6) is done putting the appropriate token in the place P_5. If the token is <b, a_3, a_5> then the relation (a_3, a_5) will be selected. On the other hand, if the token is <b, a_3, a_6> then the relation (a_3, a_6) will be selected.

5 Conclusion

This work has presented a methodology to describe and coordinate interdependencies among a set of activities performed in a computer environment. An algorithm was also presented to automate the generation of a Coordination Mechanism (MC) free from temporal inconsistencies, with a time linear to the number of activities.

The GR methodology allows automating the generation process of coordination mechanisms by means of modeling tools and of the algorithm to identify and model global conditions. Apart from reducing designers' work, automation eliminates errors

in the determination of such conditions and standardizes the process of modeling global conditions.

The coordination policy adopted in GR explores the advantages of both global and local coordination. At the local level, the MCL explores the concept of modularization, which allows modifying local mechanisms without interfering with the global MC. This coordination policy enhances the use of the methodology presented herein in collaborative environments, because generating coordination mechanisms that operate at the global level and at the local level simultaneously is one of the difficulties found in the coordination of collaborative activities [5].

Inserting one coordination level allowed a distinction between task coordination and how this task is performed. The GR methodology does not determine how the task is carried out, but what must be done and when. Another advantage of the abstraction-level approach is to make the use of the GR methodology independent from previous knowledge of the formalism used to model the MCs (in this case, Petri Nets).

References

1. Agostini A., De Michelis, G. : A Light Workflow Management System Using Simple Process Models. Computer Supported Cooperative Work 9, (2000) 335-363.
2. Allen, J.F.: Towards a General Theory of Action and Time. Artificial Intelligence 23, (1984) 123-154.
3. Chen, W., Decker, K.S.: Coordination Mechanisms for Dependency Relationships among Multiple Agents. In: Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS´02) Part 1, ACM Press, (2002) 172-173.
4. Crowston, K.: A Taxonomy of Organizational Dependencies and Coordination Mechanisms. In: Malone, T.W., Crowston, K., Herman, G. (Eds.), Organizing Business Knowledge, MIT Press, (2003) 85-108.
5. Cruz, A.J.A., Raposo, A.B., Magalhães, L.P.: Coordination in Collaborative Environments - A Global Approach. In: Proceedings of the 7th International Conference on Computer Supported Cooperative Work in Design – CSCWD, Rio de Janeiro, Brazil, (2002) 25-30.
6. Dellarocas, C.N.: A Coordination Perspective on Software Architecture: Towards a Design Handbook for Integrating Software Components. PhD Thesis, Dept. of Electrical Engineering and Computer Science, MIT (1996).
7. Lo Presti, S., Bert, D., Duda., A.: TAO: Temporal Algebraic Operators for Modeling Multimedia Presentations. Journal of Network and Computer Applications 25, Academic Press, London, UK, (2002) 319-342.
8. Malone, T.W., K. Crowston, K.: The Interdisciplinary Study of Coordination. ACM Computing Surveys 26 (1), (1994) 87-119.
9. Moldt, D., Wienberg, F.: Multi-Agent-Systems base on Colored Petri Nets. In: Proceedings of the 18th International Conference on Application and Theory of Petri Nets, Toulouse, June 23-27, France (1997) 82-101.
10. Prasad, S.K., Balasooriya, J.: Fundamental Capabilities of Web Coordination Bonds: Modeling Petri Nets and Expressing Workflow and Communication Patterns over Web Services. In: Proceedings of the 38th Hawaii International Conference on System Sciences, Big Island, Hawaii, Jan. 5-8, IEEE CS Press (2005) 12-19.
11. Ramamoorthy, C.V., Ho, G.S.: Performance evaluation of asynchronous concurrent systems using Petri Nets. IEEE Transactions in Software Engineering 6 (5), (1980) 440-449.

12. Raposo, A.B., Magalhães, L.P., Ricarte, I.L.M.: Petri Nets Based Coordination Mechanisms for Multi-Workflow Environments. International Journal of Computer Systems Science & Engineering, Special Issue on Flexible Workflow Technology Driving the Networked Economy 15 (5), CRL Publishing (2000) 315-326.
13. Schmidt, K., Simone, C.: Coordination Mechanisms: Towards a conceptual foundation of CSCW systems design. Computer Supported Cooperative Work: The Journal of Collaborative Computing 5 (2-3), (1996) 155-200.
14. Szwarcfiter, J.L.: Grafos e algoritmos computacionais (Graphs and Computational Algorithms). 2nd Edition, Editora Campus, Rio de Janeiro, Brazil (1986).
15. van der Aalst, W.M.P., van Hee, K.M., Houben, G.J.: Modeling and analyzing workflow using a Petri-net based approach. In: Proceedings of the 2nd Workshop on Computer-Supported Cooperative Work, Petri nets and related formalisms, (1994) 31-50.
16. Verbeek, H.M.W., W.M.P. van der Aalst, W.M.P., Kumar, A.: XRL/Woflan: Verification and Extensibility of an XML/Petri-Net-Based Language for Inter-Organizational Workflows. Information Technology and Management 5 (1-2), (2004) 65–110.
17. Weyns, D., Holvoet, T.: A Colored Petri Net for Multi-Agent Application. In: Proceedings of Modeling Objects, Components and Agents (MOCA'02), Aarhus, Denmark, (2002) 121-140.
18. Yoon, K., Berra, P.B.: Interactive Temporal Model for Interactive Multimedia Documents. In: Proceedings of the International Workshop on Multimedia Database Management Systems (IW- MMDBMS), (1998) 136-144.
19. Zaidi, A.K.: On temporal Logic Programming Using Petri Nets. IEEE Transactions on Systems, Man, and Cybernetics – Parte A: Systems and Humans 29 (3), (1999) 245-254.