Coordinating Multi-task Environments through the Methodology of Relations Graph

1
2
3

Abstract. One of the challenges related to the coordination of a collaborative system is the creation of structures to deal with possible conflicts caused by interdependency relations among the activities. This work presents a methodology to automate the generation of coordination mechanisms based on the temporal behaviors specified for the activities executed in the environment. It is possible to define alternative temporal behaviors and alternative activities, which may be selected in processing time, changing the temporal relations among the activities. The identification and modeling of the temporal restrictions that each activity must be subjected to result in the coordination mechanism, with a time complexity linear to the number of activities. The encapsulation and compacting capabilities of Colored Petri Nets are explored in the modeling of coordination mechanisms, although the use of the proposed methodology does not entail the knowledge of this formal system.

Keywords: Coordination, Petri nets, temporal behaviors modeling.

1 Introduction

A collaborative system may be viewed as a group of processes that, in their turn, are defined by a group of logically related activities. Activities represent well-defined fragments within the general functioning of a process and involve interdependency relations. For this reason, the execution of these activities may present conflicting requirements and interests that must be anticipated and solved in order to reach the process' goals. The coordination of these requirements may be viewed as the effort that must be added to the computational environment aiming to foresee and solve the conflict derived from the relations among activities. Therefore, in our context, coordination consists in managing the dependencies defined among activities [8] by establishing coordination mechanisms that guarantee the (temporal, spatial, causal, etc.) behaviors defined by these interdependencies.

Activity coordination processes found in the literature model the management of dependencies considering only activities that are directly related, such as those pre-

sented by [6], [12]. However, some kinds of interdependencies, such as temporal ones, may impose constraints on activities that are not directly related. The treatment of these cases by means of tools that model only directly related activities requires laborious analysis by the designer, who must identify all the cases and then verify if the model is also consistent from the global point-of-view, i.e., from the point-of-view of all non-direct relations.

Providing a solution for this issue is the motivation for a 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.

The methodology proposed herein, called GR (Relations Graph), deals with 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. This methodology does not restrict the quantity of relations in which an activity may be involved – it may be involved, directly or indirectly, with any number of other activities.

In order to generate the coordination mechanisms, an algorithm of complexity $O(n)$, n being the number of activities, is proposed. This algorithm automates the identification and modeling of temporal restrictions among activities. These restrictions are derived from the interdependencies defined among the activities. The result is 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. The mechanisms provide support for the coordination of temporal dependencies established among the application's activities.

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

2 Overview

The need to approach the issue of activity coordination in a collaborative system is due to the (temporal, resource, mutual exclusion, etc.) dependencies that exist among the activities. These dependencies impose constraints and conditions on the execution of activities, demanding an additional effort – coordination – to manage such dependencies in order to anticipate and solve possible conflicts in the environment. This issue has been the subject of varied scientific research – [4], [8], [10], [13] - and has at-

tracted the interest of experts in search for tools that help to coordinate computational environments.

Regarding coordination, *activities* represent the functional parts of the application, *dependencies* describe the relations among the activities, and *coordination processes* represent the interconnection protocol that manages the relationships and constraints resulting from the dependencies [8]. A collaborative system can be viewed as a system of interdependent activities managed through coordination processes. In this sense, the coordination of a computational environment requires two principal steps: first to identify the types of dependencies among the activities, and second to determine coordination processes to manage these dependencies, i.e. mechanisms to anticipate and solve possible conflicts generated by the dependencies.

The coordination of the collaborative environment when performing a task may demand resources beyond the capacities of an actor (a user, a computer system or a group), requiring the collaboration of other individuals for this purpose. This requirement creates dependency relations among the activities performed by these actors [4] and, as a consequence, possible conflicts among the activities. 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 several actors 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, used in the problem of coordinating the behaviors of agents, have been presented by [9] and [17].

The situations presented in this section illustrate the need for coordination models able to provide both local and global information about the system, allowing an adequate selection of the actions to be performed in the environment.

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. In this abstraction level, called *specification level* (Figure 1a), a temporal ordering is established among the activities, which is represented through an expression. In a second abstraction level, called *coordination level*, the coordination mechanism is built. This structure is the result of modeling all dependencies among the activities described in the previous level and is responsible for fulfilling those specifications. Finally, in 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. Details on the implementation of the activity execution environment and on the program that implements the coordination mechanism will not be dealt with in the present work.

3.1 Specification Level

To specify temporal behaviors, an activity-based temporal model is adopted; 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, presented as follows.

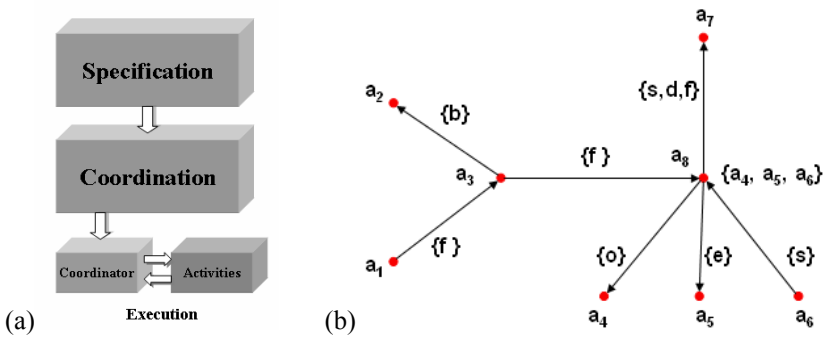


Figure 1. (a) Abstraction levels. (b) Graph of temporal behaviors of an application.

Different 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, can relate between themselves according to the temporal primitives of the set **D** = {e, s, d, f, o, m, b}, where

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 activities and edges correspond to the dependency relations among them. Figure 1b illustrates a hypothetical example. The direction of the edge indicates the order in which the activities are related, and its label states the type of dependency among them. For example, the label of edge $\mathbf{a_3}$,$\mathbf{a_2}$ – {b} – specifies that activity $\mathbf{a_3}$ must be executed before activity $\mathbf{a_2}$. The label in edge $\mathbf{a_8}$,$\mathbf{a_7}$ specifies alternative dependency relations, which means that any one of these three alternatives may take place each time these activities are activated by the application. The selection of one relation instead of others is made in execution time, according to environment requirements.

Still at specification level, alternative activities can be anticipated; for instance, the label of activity $\mathbf{a_8}$ indicates that one of the activities $\mathbf{a_4}$, $\mathbf{a_5}$ or $\mathbf{a_6}$ must be selected to be related with $\mathbf{a_8}$. This selection occurs each time activity $\mathbf{a_8}$ is activated through events fired by the application, and any of the three activities in its label can be selected. The temporal behavior specifications discussed above will be determined in more detail as follows, through definitions 1 and 2.

Definition 1: An expression **E(A,R,F)** is an oriented and labeled[1] graph, with an acyclic subjacent[2] graph, where **A** is a set of vertices representing the activities, $\mathbf{R} \subset \mathbf{A} \times \mathbf{A}$ is a set of edges and **F** is a function $\mathbf{F} : \mathbf{R} \rightarrow \wp(\mathbf{D})$ where $\wp(\mathbf{D})$ is the set of **D** parts, called edge (relation) labeling function.

Definition 2: A qualified expression is a pair **(E, G)** where **E** is an expression **E(A,R,F)** and **G** is a function $\mathbf{G} : \mathbf{A} \rightarrow \wp(\mathbf{A})$, where $\wp(\mathbf{A})$ 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 Definitions 1 and 2 to the example in Figure 1b, the following qualified expression **(E, G)** is obtained:

$\mathbf{A}=\{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8\}$;
$\mathbf{R}=\{(a_1,a_3), (a_3,a_2), (a_3,a_8), (a_8,a_4), (a_6,a_8), (a_8,a_5), (a_8,a_7),\}$;

The edge labeling function **F** is given by $\mathbf{F}(a_1,a_3)=\{f\}$, $\mathbf{F}(a_3,a_2)=\{b\}$, $\mathbf{F}(a_3,a_8)=\{f\}$, $\mathbf{F}(a_8,a_4)=\{o\}$, $\mathbf{F}(a_6,a_8)=\{s\}$, $\mathbf{F}(a_8,a_5)=\{e\}$ and $\mathbf{F}(a_8,a_7)=\{s, d, f\}$, and the vertex labeling function **G** is given by:

$\mathbf{G}(a_8)=\{a_4,a_5,a_6\}$ and $\mathbf{G}(a_1)= \ldots = \mathbf{G}(a_7)=\varnothing$

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

3.2 Coordination Level

In this section, the proposed coordination mechanisms are described.

3.2.1 Construction of Coordination Mechanisms – One Dependency

According to the definition of activity, a given activity **a**:

1. has a beginning and an end;
2. is related to **b** through a relation $\mathbf{r} \in \mathbf{D}$;
3. both the beginning and the end of **a** may be subject to temporal constraints.

Translating these items into a colored Petri Net, the diagram illustrated in Fig. 2a is obtained.

---

[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 a node- and edge-labeled graph, which we call simply labeled graph.

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

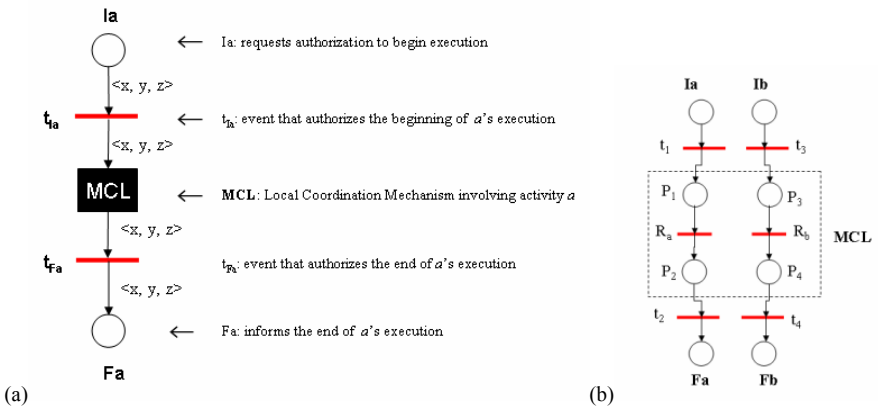


Figure 2. (a) Diagram of an activity at coordination level. (b) **CPNet** of the relation **r(a, b)**.

For an activity to be executed, it must receive an execution request (from the user, the system or the initial or final event of an activity). This request corresponds to a token in place **Ia**. Firing transition $\mathbf{t_{Ia}}$ corresponds to the event that authorizes the beginning of the execution of the activity, which means that all global conditions imposed on **a** have been fulfilled.

The Local Coordination Mechanism (MCL) relative to **a** receives the authorization event and: 1) commands the beginning of **a** ; 2) waits for the end of **a**'s execution.

Firing transition $\mathbf{t_{Fa}}$ generates the event that ends the execution of **a**. A token in place $\mathbf{F_a}$ means that **a** has been executed.

Input and output arcs of a transition are labeled. These labels determine the quantity and type of (color) tokens that will be removed from a place or added to another. The color value that can be attributed to a token is an ordered triple **<x , y , z>** 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.

For notation simplicity, the absence of a label indicates the removal of one token from the input place or the addition of one token to the output place with the same type as the token that has been removed.

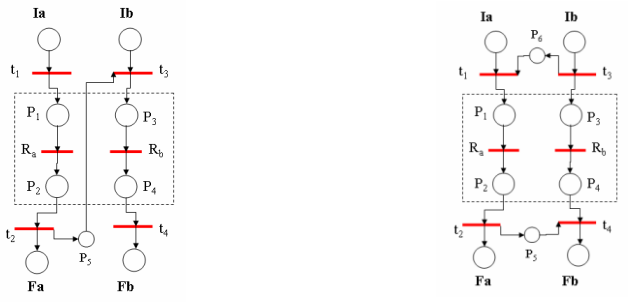
If the expression that defines the system's temporal behaviors has only one dependency relation, then the corresponding coordination mechanism, called basic MC, is derived from the diagram illustrated in Figure 2b, adding the temporal restrictions that characterizes such dependency relation.

The diagram in Figure 2b represents the networks of two activities placed side by side. 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**). In the MCL, transition $\mathbf{R_a}$ **(**$\mathbf{R_b}$**)** represents the time span for the execution of **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.

As shown in Section 3.1, a relation **r(a,b)** is defined by equations or inequations involving the initial and final moments of activities **a** and **b**. From these equations or inequations and the CPNet (Colored Petri Net) diagram shown in Figure 2b, the MCs for each of the basic relationships can be built, according to the following rules:

Rule 1: Inequations $x < y$ or $x > y$, where x and y represent the initial or final moments in the execution of the activities involved, are translated into two CPNets by adding an arc from the transition that corresponds to the least-value variable to place P_Z, $z \in N$, and an arc from P_Z to the transition that represents the largest value. For example, in relation *before*, b(a, b), we have $a_f < b_i$, which means including an arc from t_2 (which represents a_f) to place P_5 and an arc from P_5 to t_3, which represents b_i (Figure 3a).

Rule 2: Adding an equation to the CPNet, shown in Figure 2b, consists in performing a merge operation (see Definition 3, Section 3.2.2) of the transitions associated to the equation variables. Figure 3 presents the MCs for relations before and during, respectively. The complete list of basic MCs can be found in [5].



(a) MC of relation before: b(a, b)

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

Figure 3. Examples of basic MCs.

The GR methodology also allows the specification of alternative temporal behaviors. In this case, 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. As follows (Figure 4), the coordination mechanism for the case in which alternative behaviors are defined by only one activity pair is presented. This mechanism, called basic MC, models all relations anticipated for activities **a** and **b** and allows only one of them to be selected at a time.

Selection of one of the relations estimated for activities **a** and **b** is made according to the color of tokens at places **Ia** and **Ib**. In the ordered triple <**r,a,b**> that defines the color of tokens in the net, **r** is the relation that must be selected between **a** and **b**.

The GR methodology also 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. The coordination mechanism presented in

Figure 5 is a basic mechanism for alternative activities defined by only one relation. 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 **Ib$_i$**.

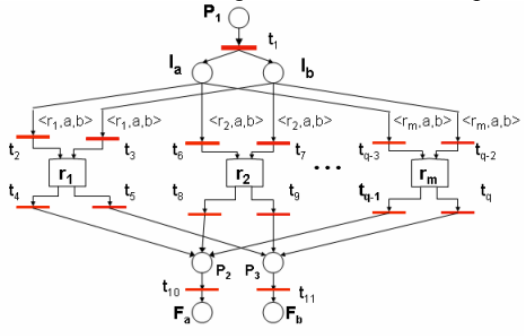


Figure 4: Basic MC for alternative behaviors – relations.

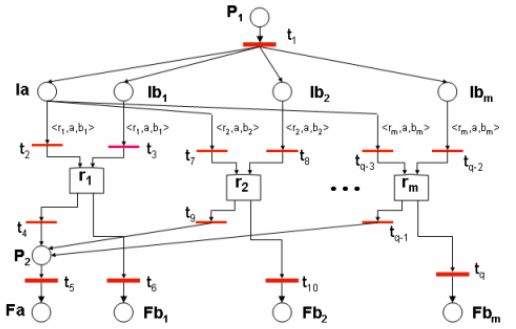


Figure 5: Basic MC for alternative behaviors – activities.

3.2.2. Construction of Coordination Mechanisms – Two or More Dependencies

MC connection through transition merging

Coordination mechanisms of an expression with more than one dependency relation can be derived from basic MCs. The construction procedure is based on the connection of pairs of coordination mechanisms (**MC$_1$, MC$_2$**) with a common activity. Figures 6a and 6b 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 Figures 6a and 6b, 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 $\mathbf{t_2}$ and $\mathbf{t_5}$ ($\mathbf{t_4}$ and $\mathbf{t_7}$), generating a single transition that receives all temporal restrictions involving the beginning (end) of **b**, according to Definition 3. Figure 6c illustrates the **MC** resulting from the connection operation between $\mathbf{MC_1}$ and $\mathbf{MC_2}$.

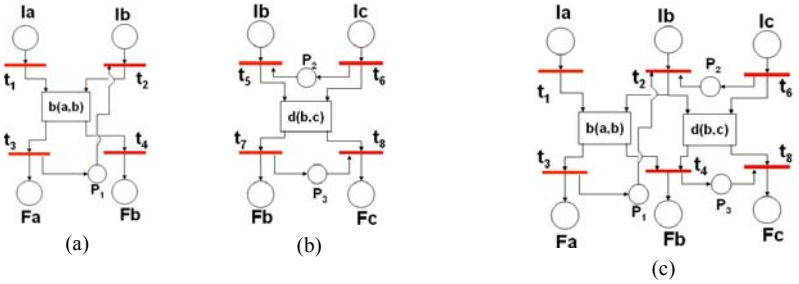


Figure 6 (a) MC_1: relation *before* **b(a,b);** (b) MC_2: relation *during* **d(b,c);** (c) MC resulting from the connection operation between MC_1 and MC_2.

Definition 3: Merging two transitions t' and t" is executed by transferring to one of them all the input and output arcs of the other.

MC connection through transition splitting

Connections involving basic MCs for alternative behaviors require an extension of these mechanisms, because temporal restrictions derived from alternative behaviors are only known after the dependency relation or the activity is selected. Moreover, the fact that two mechanisms share a common activity implies the existence of temporal restrictions relative to this activity, modeled in both mechanisms. The MC resulting from this connection must manage these temporal restrictions in order to ensure that the event that authorizes the beginning of this activity is only generated in the instant when all temporal restrictions have been satisfied.

The extension of basic MCs for alternative behaviors consists in inserting a feature, here called connection feature. Figure 7 presents the basic MC for alternative relations with the connection feature for activity **a**. This extension, identified by traced arcs, splits the transitions (Definition 4) $\mathbf{t_2}$, $\mathbf{t_6}$, ··· , $\mathbf{t_{q-3}}$ from Figure 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 Figure 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 $\mathbf{t_2'}$, $\mathbf{t_6'}$, ··· , $\mathbf{t_{q-3}'}$ (Figure 7). These restrictions are not represented here because relations $\mathbf{r_1}$, $\mathbf{r_2}$ ··· , $\mathbf{r_m}$ are generic.

Definition 4: 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".

Alternative behaviors: dependency relations and alternative activities

Let us consider that $\mathbf{r_1}$ (Figure 7) represents relation **b(b, a)** – activity **b** must be executed before **a**. In this case a temporal restriction must exist starting at transition $\mathbf{t_5}$, whose firing indicates that **b** was already executed, and reaching transition $\mathbf{t_2'}$,

similarly to the place $\mathbf{P_1}$ inserted between transitions $\mathbf{t_3}$ and $\mathbf{t_2}$ in Figure 6a. Back to Figure 7, firing one of the transitions $\mathbf{t_2'}$, $\mathbf{t_6'}$, ⋯ , $\mathbf{t_{q-3}'}$ indicates that the temporal restrictions derived from one of the alternative relations was fulfilled, which allows transition $\mathbf{t_N}$ whose firing generates the authorization event to begin execution of **a**.

Finally, to perform the connection 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 $\mathbf{t_{10}}$ and a transition from the other MC that generates the authorization to end **a**, considering **a** as the common activity to these mechanisms.

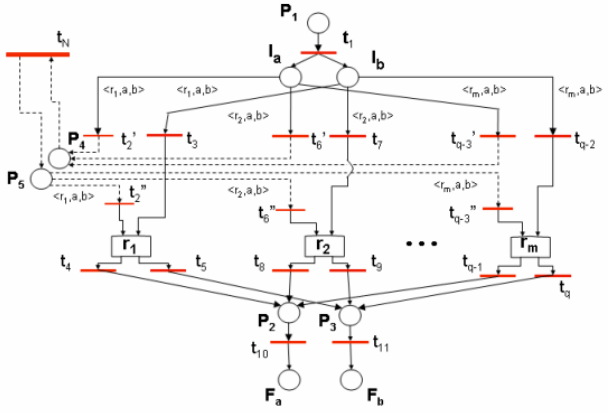


Figure 7: Basic MC with connection features for alternative relations.

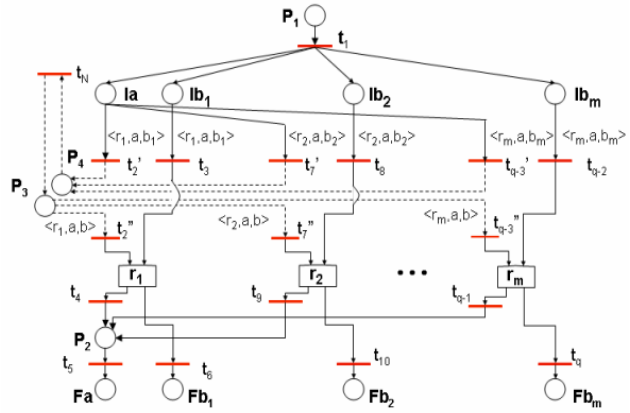


Figure 8: Basic MC for alternative behaviors to alternative activities**.**

The extension of basic MCs for alternative activities is completely analogous to that of basic MCs for alternative relations. Figure 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**

3.2.3 Algorithm for Global Conditions (CGs)

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[3]
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 CGs, because the fact that an activity $\mathbf{a_i}$ is a leaf (an activity with only one dependency relation $\boldsymbol{\partial(a_i)}$[4] $\mathbf{= 1}$) or is internal (an activity with more than one dependency relation $\boldsymbol{\partial(a_i) > 1}$) 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 $\{\boldsymbol{\varepsilon_i}\}$, $\mathbf{i = 0,1,2,...,u}$, obtained from the original expression **ε**, according to the following formation rule:

Table 2. Formation Rule

$\boldsymbol{\varepsilon_i}$	Formation Rule
$\boldsymbol{\varepsilon_0}$	is the original expression **ε**;
$\boldsymbol{\varepsilon_1}$	Expression derived from $\boldsymbol{\varepsilon_0}$, eliminating the activities with degree 1 from $\boldsymbol{\varepsilon_0}$;
$\boldsymbol{\varepsilon_2}$	Expression derived from $\boldsymbol{\varepsilon_1}$, eliminating the activities with degree 1 from $\boldsymbol{\varepsilon_1}$;
⋮	⋮
$\boldsymbol{\varepsilon_u}$	Expression derived from $\boldsymbol{\varepsilon_{u-1}}$, eliminating the activities with degree 1 from $\boldsymbol{\varepsilon_{u-1}}$.

[3] ε denotes the expressions **E(A, R, F)** and **(E, G)**.

[4] ∂(ai) denotes activity degree, i.e., the number of activities aj (j ≠ i) related to ai.

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

A partition $\mathbf{P}$ of activity set $\mathbf{A}$ of an expression $\boldsymbol{\varepsilon}$ is given by $\mathbf{P(A) = \{I_0, I_1,...,I_u\}}$, where $\mathbf{u \leq n/2 - 1}$, $\mathbf{n}$ 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 $\mathbf{A}$. In fact, it is not necessary to effectively create $\{\varepsilon_i\}$. Computing a list containing the degree of all activities is sufficient. Thus, $\mathbf{I_0}$ is determined by selecting the $\mathbf{a_i}$ activities in the list such that $\boldsymbol{\partial(a_i) = 1}$. To determine $\mathbf{I_1}$, the list must be updated. For each activity $\mathbf{a_i}$ from $\mathbf{I_0}$: i) remove $\mathbf{a_i}$ from the list; ii) subtract one unit from the degree of activity $\mathbf{a_j}$ related to $\mathbf{a_i}$. Then $\mathbf{I_1}$ is determined by all $\mathbf{a_j}$ activities, $\mathbf{j \neq i}$ such that $\boldsymbol{\partial(a_j) = 1}$. This procedure ends with a list that has only one or two activities, i.e. the center of the expression, which corresponds to set $\mathbf{I_u}$.

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

Definition 5: An a_i-star is a subexpression 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 $\mathbf{A}$ and $\mathbf{B}$ 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 ai and aj 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 Figure 1b, we have:

1. Considering the expression presented in Figure 1b, $\mathbf{a_1}$, $\mathbf{a_2}$, $\mathbf{a_4}$, $\mathbf{a_5}$, $\mathbf{a_6}$, $\mathbf{a_7}$ are degree 1 activities.
2. Following the outside-in order, first the basic MCs must be built for the labels of edges ($\mathbf{a_1}$, $\mathbf{a_3}$) and ($\mathbf{a_3}$, $\mathbf{a_2}$), and the connection operation between these two basic MCs must be executed.
3. Then the basic MC for edge ($\mathbf{a_8}$, $\mathbf{a_7}$) and the basic MC for the label of activity a_8 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.

For the example in Figure 1b, the new degree 1 activities are $\mathbf{a_3}$ and $\mathbf{a_8}$, which determine the edge ($\mathbf{a_3}$, $\mathbf{a_8}$). Therefore, the basic MC for the edge label must be built and the connection operations must be executed with both mechanisms constructed in the previous iteration, resulting in the MC of the expression of the example illustrated in Figure 1b. The construction process always ends, because in a given moment the graph will only have one or two activities corresponding to the center[5] of the original graph. This occurs because, without the direction of the edges, the original expression is an acyclic and connected graph, and the center of a graph with these characteristics must have one or two vertices [14].

One can demonstrate that the algorithm to identify and model the CGs has complexity **O(n)**, where **n** is the number of activities. This demonstration is made based on the following sequence of facts:

1. To select the activity set $\mathbf{I_k}$, $\mathbf{k = 1,2,...,u < n/2 - 1}$, in the k^{th} iteration all degree 1 activities must be identified, which requires **O(n)** operations for each iteration. Therefore, the algorithm would become $\mathbf{O(n^2)}$. To maintain linearity, the degree of all activities is computed only once, and at each iteration the relation among the activities' degrees is updated by removing leaves from $\boldsymbol{\varepsilon_k}$. This avoids having to compute the activities' degrees at each new iteration. Thus, only the cost of the activity removal operation remains at each iteration, which is **O(n)**, because there are **n** activities that are removed only once.
2. The cost of the global MC construction process for a star is given according to its number of relationships, being **O(1)** for each relation. The maximum number of re-

[5] "The center of a graph is the subset of vertices with minimum eccentricity". The eccentricity of a vertex is the maximum distance between this vertex and any other vertex in the graph [14]. The same definition is adopted for the center of an expression in GR.

lations is 7(n-1), considering the case where each one of the n – 1 edges has a label with 7 elements. Thus, the construction of the global MC for all stars in the expression is **O(n)**.

3. The cost to connect the stars' coordination mechanisms is also **O(n)**, because the number of stars is always smaller than the number of activities in the expression.

4 Conclusion

This work has presented a methodology to describe and coordinate interdependencies among a set of activities performed in a collaborative environment. An algorithm was presented to automate the generation of a Coordination Mechanism (MC) free from temporal inconsistencies. The use of concepts from graph theory in the description process and in temporal behavior modeling allowed the principal contribution of this work: the development of an algorithm that is linear with the number of activities involved in a collaborative environment.

The GR methodology allows automating the generation 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. <omitted for blind review>
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: Proc. 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.