# Comparing Semantic Frameworks for Coordination: On the Conformance Issue for Coordination Media

Mirko Viroli
DEIS, Università di Bologna
via Rasi e Spinelli 176
47023 Cesena (FC), Italy
mviroli@deis.unibo.it

## ABSTRACT

A fundamental issue in the engineering of coordination models is to design coordination abstractions that are correct with respect to the specification of the coordination model they implement. The traditional semantic framework for coordination is focused on describing the admissible evolutions over time of a coordinated system, and is particularly suitable for specifying the laws of a coordination model. On the other hand, formally describing run-time aspects of an implementation requires a different framework, capturing as fundamental idea the interactive behavior of a coordination medium.

In this paper, these two frameworks are compared by tackling a crucial issue of coordination models, that is, the conformance of an implementation with respect to a specification. In particular, a definition of conformance is introduced that is shown to be compatible with the standard notion of implementation by horizontal refinement promoted in the context of process algebras.

## Keywords

Semantics of Coordination Models, Linda

## 1. BACKGROUND AND MOTIVATION

Formal semantics are generally meant to capture, describe, and understand in a precise way some aspects of interest of a system at the desired level of abstraction. This issue becomes particularly crucial for those systems whose complexity cannot be simply tackled by the designer's early experience. In this case, the precise description provided by formal models generally makes it possible to devise a system design which can be easily guided to correct, reliable, and effective implementations.

Coordination languages and models are being exploited as a successful framework for structuring complex concurrent systems, providing the proper *coordination abstractions* to manage and rule the interactions of separated activities. Traditional examples of coordination abstractions are tuple spaces in the LINDA model [16] and channels in MANIFOLD [1]. The issue of coordination is particularly crucial for tackling complexity of today's distributed systems, and as a result, formal approaches have been extensively proposed to capture the true semantics of coordination models.

In particular, the original application of coordination models to closed, parallel environments promoted a viewpoint of *coordination as a concurrent language* for defining the interactive part of a system [27]. Correspondingly, the traditional semantic framework for coordination is the same used to formalize foundational calculi for concurrent languages such as CCS [19]. A coordinated system is understood as a concurrent program: the set of states of the coordinated system is modeled as a process algebra [2] and its evolution by means of Plotkin's structural operational semantics (SOS) [24]. By this approach, not only traditional coordination models have been given a semantic account – such as e.g. LINDA in [8] and MANIFOLD in [3] – but also advanced features of today's coordination infrastructures have been described – as in the formalization of JavaSpaces [15] proposed in [10]. We refer to this semantic framework as "*Coordination as a Language*" (CaL).

On the other hand, nowadays coordination is almost never deployed as a language supported by a compiler as originally conceived [17], but rather, as an infrastructure providing *coordination as a service* to component-oriented applications. Some examples of coordination infrastructures are JavaSpaces, Lime [23], and TuCSoN [21]. As recently claimed in [27], this new viewpoint on coordination calls for considering as first-class entity of investigation the *coordination medium* [12], which represents the run-time abstraction providing the coordination service, eventually deployed as part of a coordination infrastructure. The semantic framework promoted by this notion naturally describes coordination in terms of the interactive behavior of the coordination medium, which is amenable e.g. to a characterization as a labelled transition system [18]. Many works exist that proceeds along this direction, such as the study of Abstract Linda Systems in [17], the semantic framework for tuple-based coordination models proposed in [20], the observation framework for coordination [28, 26], the formal framework for event-based systems used in [14], the study of LINDA optimizations in [25], and the notion of expressiveness for coordination media developed in [29]. We refer to the general semantic framework underlying these approaches as "*Coordination as a Service*" (CaS).

The framework CaL has the main goal of representing in a sound way the possible evolutions of a system adopting a given coordination model, so it is quite useful to provide compact specifications of the laws underlying this model – namely, the *coordination laws* [12]. For instance, in [7] this framework is used to show that the ordering semantics for the LINDA *out* primitive may indeed change the expressiveness of the model, whereas existing informal spec-


*SAC* 2003 Melbourne, Florida USA

ifications lack this information, and existing systems implement different semantics. Moreover, the work in [11] showed that the framework CaL can also be exploited for discovering incorrectness of informal specifications, as in the case of the claimed serializability of transactions in JavaSpaces.

On the other hand, the framework CaS is more geared towards describing the implementation of a coordination model. In particular, it provides an operational description of the coordination media supporting the model, specifying their single-step capability of evolving their internal state and interacting with coordinated entities. As claimed also in [17], reasoning about a coordination model in terms of the interactive behavior of its media generally facilitates the understanding of the run-time evolution of the resulting system.

Given the different applications of the two frameworks, one for specifying the laws of a coordination model, the other for describing their implementations, a fundamental question naturally arises: in the context of coordination models, when can an implementation be assumed to conform to a specification? The main goal of this paper is to address this issue.

More technically, we provide a notion of conformance of a coordination medium with respect to some coordination laws, that is, a notion of conformance of a CaS model with respect to a CaL model. Some difficulties arise in solving this problem, stemming to the intrinsic semantic differences between the two frameworks. While a CaL model is essentially the definition of a concurrent programming language, possibly flavouring techniques such as encodings [9] and modular embeddings [5], a CaS model represents the coordination medium abstraction by a labelled transition system, enabling the adoption of techniques related to observational semantics, such as process equivalence and preorder [18].

## 2. OUTLINE

Section 3 describes the basic notation and syntactic conventions used in this paper, while Sections 4 and 5 briefly discuss the basic semantic frameworks CaL and CaS.

Then, our investigation of the conformance problem proceeds in two steps. In Section 6, we start by defining a notion of conformance of a coordination model $\mathbb{Y}$ with respect to another $\mathbb{X}$, both represented on the CaL framework. This is essentially based on the existence of an encoding of $\mathbb{Y}$ into $\mathbb{X}$ that is required to both *(i)* satisfy compatibility with respect to the operational semantics of $\mathbb{X}$ and $\mathbb{Y}$, and *(ii)* preserve a notion of successful termination. Since the CaS framework can be seen also as a more refined version of CaL – that is, a CaS model can be interpreted in terms of the framework CaL –, this result may be applied e.g. to check conformance between a CaL and a CaS model, stating whether a coordination medium satisfies given coordination laws. A notion of equivalence between two models is then defined as their mutual conformance, which can be used e.g. to check if a coordination medium exhibits all the behavior allowed by some coordination laws. As a relevant example of application, in Section 7 we define a coordination medium exactly implementing the standard laws of LINDA as reported in Section 4.

As a second step, in Section 8 we analyze the relationship between our notion of conformance and the existing notion of "process implementation" by *(horizontal) refinement* [18] promoted in the context of interactive systems and process algebras. We show an important result: if in a CaS model we substitute a coordination medium with a refinement of it, we obtain a new CaS model that conforms to the original one. This means that once we identify the coordination medium equivalent to a given CaL model, each refinement of it keeps satisfying the same rules, so it still conforms to the coordination model specification.

## 3. NOTATION

In the remainder of the paper, given any set $\mathcal{A}$, this is automatically ranged over by meta-variable $A$ (and by its decorations $A'$, $A''$, $A_0$, $A_1$, ... ) – and analogously, $\mathcal{B}$ is ranged over by meta-variable $B$ and its decorations, and so on. Then, in any algebraic structure $(\mathcal{A}, \|, 0)$, where "$\|$" is a binary composition operator, and $0 \in \mathcal{A}$ is the *zero* of the structure, we suppose that the following congruence rules hold:

$$A \,\|\, 0 \equiv A \quad A \,\|\, A' \equiv A' \,\|\, A \quad (A \,\|\, A') \,\|\, A'' \equiv A \,\|\, (A' \,\|\, A'')$$

Typically, one such structure is defined by a BNF rule of the kind $A ::= 0 \mid \ldots \mid A \parallel A$. In this case, operator $\prod$ is used for multiple application of composition, writing e.g. $\prod_{i\in\{1,2,3\}} A_i$ as a shorthand for $A_1 \parallel A_2 \parallel A_3$. Notation $a \in A$ means that $A$ is equivalent (modulo the above congruence rules) to a term of the kind $a \parallel A'$ (and intuitively $a \notin A$ is used when $a \in A$ does not hold).

In this paper, a (labelled) transition system is a structure $\mathbb{X} = \langle \mathcal{X}, \longrightarrow_{\mathcal{X}}, Act_{\mathcal{X}}, \mathcal{X}_0 \rangle$. $\mathcal{X}$ is the set of states of the system of interest, $Act_{\mathcal{X}}$ is the set of its *actions* (or labels), $\longrightarrow_{\mathcal{X}} \subseteq \mathcal{X} \times Act_{\mathcal{X}} \times \mathcal{X}$ is the *relation transition*, orderly associating an old state, an action, and a new state, and $\mathcal{X}_0 \subseteq \mathcal{X}$ is the set of valid initial states. As usual, notation $X \xrightarrow{a}_{\mathcal{X}} X'$ is a shorthand for $\langle X, a, X' \rangle \in \longrightarrow_{\mathcal{X}}$. A transition system without labels is naturally defined as $\mathbb{X} = \langle \mathcal{X}, \longrightarrow_{\mathcal{X}}, \mathcal{X}_0 \rangle$ where the transition relation is of the kind $\longrightarrow_{\mathcal{X}} \subseteq \mathcal{X} \times \mathcal{X}$. For simplicity of treatment, in both kinds of transition systems we stick to the case where $\mathcal{X}$ coincides with the set of states reachable from $\mathcal{X}_0$ by a finite sequence of transitions $\longrightarrow_{\mathcal{X}}$.

## 4. COORDINATION AS A LANGUAGE

The traditional approach to the formalization of a coordination model is based on expressing a coordinated system in terms of a process algebra [2], providing a SOS semantics in the style of Plotkin's [24]. The distributed state of a coordinated system is seen as a parallel composition of agents and items of the interaction space – also called the *shared dataspace* [8, 5] – both represented as terms of the algebra. From a coordinated entity viewpoint, the execution of a coordination primitive is modeled similarly to synchronous communications as for instance in CCS [19]. From the dataspace viewpoint, instead, a coordination primitive can be executed only if some condition on the dataspace is satisfied (e.g. a datum occurs), and the execution causes a change on the dataspace (e.g. the datum is removed).

As a simple yet relevant example, consider the following formalization of LINDA model, adhering to the style of [8]. We stick to LINDA primitives $in$, $rd$, and $out$ for simplicity – avoiding the management of predicative queries $inp$ and $rdp$ as well as the primitive $eval$ for spawning processes. Primitive $out$ is given unordered interpretation as defined in [7], for generality. As typically done in the CaL framework, we also avoid to deal with tuple matching functionality. The set of LINDA operations is then simply defined as $\alpha ::= \beta \mid out(x)$, where $\beta ::= in(x) \mid rd(x)$ and $x$ is a generic datum of the dataspace – namely, a tuple. Without lack of generality, we consider coordinated entities $P \in \mathcal{P}$ as finite, sequential processes performing LINDA operations, with the CCS-like syntax $P ::= 0 \mid \alpha.P$.

The set $\mathcal{L}$ of admissible admissible configurations for a LINDA system – briefly, its set of states –, is defined as a (finite) composition of processes $P$, data $x$, and items $\langle x \rangle$ representing requests for inserting $x$ in the dataspace. This structure is easily represented by

grammar:

$$L ::= 0 \mid P \mid x \mid \langle x \rangle \mid L \,\|\, L$$

According to the CaL framework, then, a SOS semantics is given to this model by means of a transition system, describing the admissible evolutions of a system configuration. This is of the kind $\mathbb{L} = \langle \mathcal{L}, \longrightarrow_{\mathcal{L}}, \mathcal{L} \rangle$, where valid initial states are all the configurations, and where relation transition $\longrightarrow_{\mathcal{L}}$ is specified by the rules:

$$out(x).P \,\|\, L \longrightarrow_{\mathcal{L}} P \,\|\, L \,\|\, \langle x \rangle \qquad \text{[L-OUT]}$$

$$\langle x \rangle \,\|\, L \longrightarrow_{\mathcal{L}} L \,\|\, x \qquad \text{[L-INS]}$$

$$rd(x).P \,\|\, L \,\|\, x \longrightarrow_{\mathcal{L}} P \,\|\, L \,\|\, x \qquad \text{[L-RD]}$$

$$in(x).P \,\|\, L \,\|\, x \longrightarrow_{\mathcal{L}} P \,\|\, L \qquad \text{[L-IN]}$$

Execution of $out(x)$ primitive causes a pending request $\langle x \rangle$ to be inserted in the dataspace, waiting to be materialized in an actual datum $x$ by means of rule [L-INS]. Then, a $rd(x)$ primitive is executed only when $x$ actually occurs in the space, so that the process continuation $P$ can carry on. The primitive $in(x)$ has similar semantics, but its execution causes $x$ to be removed from the space. In general, these rules specify the safety conditions of LINDA, describing which behavior of an actual implementation of the model can be considered valid. So, these rules are easily seen as the coordination laws of LINDA.

Notice that the CaL framework does not promote the clear encapsulation of the coordination abstraction. On the one hand, it is generally unclear which terms of the algebra represent *(i)* parts of the processes subject to coordination, *(ii)* parts of the system providing coordination at run-time (the tuple space in the case of LINDA), or *(iii)* parts of others run-time abstractions living in the system. In the specific case analyzed in this section, for instance, pending requests $\langle x \rangle$ may either be considered as part of the tuple space or of the interaction space – respectively representing the case where unordered semantics of $out$ is caused by the implementation of the medium or by the communication infrastructure. Similar problems exist e.g. also with the formalization of transactions and notification in JavaSpaces [10]. Furthermore, the exact interaction acts occurring between coordination medium and coordinated entities are not here represented either. For instance, the invocation of primitive $in$ involves at run-time a request to the tuple space and then a subsequent reply: in the above model instead, only the reply phase is taken into account, since e.g. rule [L-IN] atomically represents the tuple removal and the process continuation carrying on.

Notice that there are some works on the expressiveness of coordination models such as [5] where a distinction between dataspace and coordinated entities is clearly remarked, but where run-time interaction acts are not fully taken into account. Therefore, we consider these works as still adhering to the CaL framework.

# 5. COORDINATION AS A SERVICE

In [27] the formal framework underlying the notion of "Coordination as a Service" is introduced as a means to provide a unique semantic setting for the many works developed to reason about coordination models at run-time. In that framework, a coordinated system is conceptually split in three distinct parts:

- A *coordinated space*, made of a finite number of coordinated entities, each tagged by a unique identifier.
- A *coordination space*, made of one or more coordination media implementing the coordination laws.
- An *interaction space*, where communication events can be posted and consumed, mediating between entities and media.

At a given time, a coordinated entity posts a *request event* on the interaction space, which will be eventually consumed by some coordination medium. Then, a coordination medium may post a *reply event* to the interaction space, which will be consumed by the coordinated entity it was directed to.

In this paper, as far as a comparison with LINDA traditional semantics is concerned, we stick to a slightly simpler formalization than the one presented in [27], assuming that *(i)* only one coordination medium exists in the coordination space and that *(ii)* the interaction space is not represented. In particular, the latter hypothesis makes us represent only closed coordinated systems with synchronous communications. However, all the results we obtain therefore can be easily extended to the case where the interaction space (as formalized in [27], Section 4) is simply supposed not to loose messages nor change their order of delivery – which is actually a very common assumption in distributed systems. Dealing with a coordination space with a multitude of coordination media interacting each other is left as future work.

The key idea of this formal framework is to represent a coordinated system $\mathbb{S}$ as the explicit composition of two transition systems: one describing the behavior of the coordinated space $\mathbb{C} = \langle \mathcal{C}, \rightarrow_{\mathcal{C}}, Act_{\mathcal{C}}, \mathcal{C}_0 \rangle$, and one describing the behavior of the coordination medium $\mathbb{M} = \langle \mathcal{M}, \rightarrow_{\mathcal{M}}, Act_{\mathcal{M}}, \mathcal{M}_0 \rangle$.

Each states $C \in \mathcal{C}$ of the coordinated space is made of a set of coordinated entities $Q \in \mathcal{Q}$, each tagged by a unique identifier $id \in Id$, and denoted by the syntax:

$$C ::= 0 \mid \langle id, Q \rangle \mid (C \parallel C)$$

The behavior of each coordinated entity can be understood as an interactive component producing requests $req \in Req$, consuming replies $rep \in Rep$, or performing a silent action $\tau$, that is, in terms of a transition system $\langle \mathcal{Q}, \longrightarrow_{\mathcal{Q}}, Act_{\mathcal{Q}}, \mathcal{Q}_0 \rangle$ where the set of actions $Act_{\mathcal{Q}}$ is of the kind $Act_{\mathcal{Q}} ::= \tau \mid \uparrow req \mid \downarrow rep$. The set of actions for the coordinated space is defined as $Act_{\mathcal{C}} ::= \tau \mid id \uparrow req \mid id \downarrow rep$, respectively representing *(i)* silent action of any coordinated entity, *(ii)* a request event of entity $id$ providing request $req$, or *(iii)* a reply event sent towards entity $id$ providing reply $rep$. The set of request events is denoted by $E^{\uparrow}$ and is ranged over by variable $e^{\uparrow}$, the set of reply events by $E^{\downarrow}$ ranged over by $e^{\downarrow}$. Semantics is simply assigned to $\longrightarrow_{\mathcal{C}}$ by rules:

$$\frac{Q \xrightarrow{\uparrow req}_{\mathcal{Q}} Q'}{\langle id, Q \rangle \,\|\, C \xrightarrow{id \uparrow req}_{\mathcal{C}} \langle id, Q' \rangle \,\|\, C}$$

$$\frac{Q \xrightarrow{\downarrow rep}_{\mathcal{Q}} Q'}{\langle id, Q \rangle \,\|\, C \xrightarrow{id \downarrow rep}_{\mathcal{C}} \langle id, Q' \rangle \,\|\, C}$$

$$\frac{Q \xrightarrow{\tau}_{\mathcal{Q}} Q'}{\langle id, Q \rangle \,\|\, C \xrightarrow{\tau}_{\mathcal{C}} \langle id, Q' \rangle \,\|\, C}$$

The set of initial states $\mathcal{C}_0$ of the coordinated space is simply defined by the composition of coordinated entities in an initial state.

In order to compose the coordinated space $\mathbb{C}$ to a coordination medium, we suppose that $Act_{\mathcal{M}} = Act_{\mathcal{C}}$, that is, the coordination medium may accept request events and produce reply events compatible to $\mathbb{C}$, and may perform the internal silent action $\tau$. [1] The system $\mathbb{S}$ obtained by composing $\mathbb{C}$ and $\mathbb{M}$ is denoted by the syn-

[1]For simplicity, this formalization of the coordination medium is again slightly different from the one reported in [27], where reply events are sent only correspondingly to the reception of incoming requests.

tax $\mathbb{M} \otimes \mathbb{C}$, which forms the specification of the whole coordinated system in the CaS framework.

Each model in the CaS framework can be actually given a representation in the CaL framework, by associating to $\mathbb{S}$ a transition system $\langle \mathcal{S}, \longrightarrow_{\mathcal{S}}, \mathcal{S}_0 \rangle$, describing the evolutions of the whole coordinated space. To this end, elements $S \in \mathcal{S}$ are defined by the syntax $S ::= M \otimes C$, simply composing states of $\mathbb{M}$ and $\mathbb{C}$, $\mathcal{S}_0$ is made by composing initial elements of $\mathbb{M}$ and $\mathbb{C}$, namely, $S_0 ::= M_0 \otimes C_0$, and $\longrightarrow_{\mathcal{S}}$ is defined by rules:

$$\frac{M \xrightarrow{e\uparrow}_{\mathcal{M}} M' \qquad C \xrightarrow{e\uparrow}_{\mathcal{C}} C'}{M \otimes C \longrightarrow_{\mathcal{S}} M' \otimes C'} \qquad \frac{M \xrightarrow{\tau}_{\mathcal{M}} M'}{M \otimes C \longrightarrow_{\mathcal{S}} M' \otimes C}$$

$$\frac{M \xrightarrow{e\downarrow}_{\mathcal{M}} M' \qquad C \xrightarrow{e\downarrow}_{\mathcal{C}} C'}{M \otimes C \longrightarrow_{\mathcal{S}} M' \otimes C'} \qquad \frac{C \xrightarrow{\tau}_{\mathcal{C}} C'}{M \otimes C \longrightarrow_{\mathcal{S}} M \otimes C'}$$

As a result, each model in the CaS framework can be considered also as a model in the CaL framework, in that it is still used to describe the admissible evolutions of the coordinated system. However, the CaS framework also provides a clear separation of the coordinated space and the coordination medium. Since the formalization of coordinated entities simply concerns local properties of the coordination models [17], such as the kind of communication protocol associated to each primitive, the core semantic part of the specification is then the one within the coordination medium. Hence, the CaS framework can be seen as the refinement of the CaL framework that explicitly promotes the description of a coordination model in terms of the interactive behavior of its coordination medium.

# 6. CONFORMANCE AND EQUIVALENCE

We start our investigation by defining notions of conformance and equivalence between two coordination models expressed in terms of the framework CaL – that is, expressed as transition systems $\langle \mathcal{X}, \longrightarrow_{\mathcal{X}}, \mathcal{X}_0 \rangle$. Informally speaking, our intention is to consider a model $\mathbb{A}$ as conforming to $\mathbb{B}$, if all the evolutions allowed by $\mathbb{A}$ are also allowed by $\mathbb{B}$, namely, if the laws defining $\mathbb{A}$ are more strict than the laws described by $\mathbb{B}$. Correspondingly, a natural notion of equivalence can be defined as mutual conformance of two models. To the end of introducing these notions in a formal way, we first briefly survey existing techniques for comparing coordination models.

## 6.1 On embeddings and encodings

There are some formal techniques used to compare concurrent languages that have been so far exploited also for semantics of coordination models.

In [5] a technique called *modular embedding* [13] is used to compare the expressiveness of coordination models including various features related to transactions, constraint programming, and term-rewriting models. First of all, given two languages (or models) $\mathcal{L}$ and $\mathcal{L}'$, their comparison is defined in terms of the *observation criteria* $\mathcal{O} \in L \mapsto Obs$ and $\mathcal{O}' \in L' \mapsto Obs'$, associating to each element of the language a representation of it according to a given abstraction level. According to these observation criteria, $\mathcal{L}$ is said to *embed* $\mathcal{L}'$ if there exists an encoding of languages $\mathcal{C} \in \mathcal{L}' \mapsto \mathcal{L}$ and a decoding of observations $\mathcal{D} \in Obs \mapsto Obs'$ so that for any $L'$ we have $\mathcal{D}(\mathcal{O}[\mathcal{C}(L')]) = \mathcal{O}'[L']$. In the context of concurrent languages, this embedding is required to be *modular*, that is, to satisfy the following three properties: *(i)* since typically observations are powersets, decoding $\mathcal{D}$ should be defined elementwise, *(ii)* encoding $\mathcal{C}$ should be compositional with respect to nondeterministic operators (such as choice and parallel composition), and *(iii)* a notion of termination invariance on observations should be preserved by the decoding.

In the context of this paper, we cannot fully rely on this technique for comparing coordination models for a number of reasons. First, we cannot assume the existence of nondeterministic operators in our languages, so that property *(ii)* cannot be satisfied. In fact, a CaS model is intrinsically non compositional, since it considers the coordination system as an aggregation of one coordination medium and more coordinated entities [2]. Then, the notion of termination invariance alone – without property *(ii)* – now is too weak, since it would equate programs that simply lead to the same termination result, but perhaps to different final states and through different evolutions. Notice that in the embedding approach, the operational semantics of a language is generally used only so as to determine observables, e.g. to check if the evolution of a program eventually terminates in a successful way. On the other hand, for our purpose operational semantics should play a more crucial role to compare the evolutions induced by two coordination models.

Another approach, which seems more interesting to our end, is based on the notion of *encoding*, which has been used e.g. by Busi *et al.* in their study of expressiveness for coordination models [9]. Its basic idea is that a model $\mathbb{X} = \langle \mathcal{X}, \longrightarrow_{\mathcal{X}}, \mathcal{X}_0 \rangle$ is considered more expressive than another $\mathbb{Y} = \langle \mathcal{Y}, \longrightarrow_{\mathcal{Y}}, \mathcal{Y}_0 \rangle$ if $\mathbb{X}$ is able to represent all the system transformations represented by $\mathbb{Y}$. Variations of this kind of encoding have been used to study relative notions of expressiveness of different coordination primitives, and to provide separation results between two models by devising behavior properties – such as termination and divergence – decidable for one language but not for the other.

In spite of the many variations of encodings exploited in these papers, a general way to formally represent an encoding is as a function $\left|.\right| \in \mathcal{Y} \mapsto \mathcal{X}$ so that:

$$Y \rightarrow_{\mathcal{Y}} Y' \quad \Rightarrow \quad \left|Y\right| \longrightarrow^{+}_{\mathcal{X}} \left|Y'\right|$$

that is, a single transition in $\mathbb{Y}$ must be simulated by one or more transitions in $\mathbb{X}$ (denoted by $\longrightarrow^{+}$). We take this encoding technique as a basic framework for defining our notion of conformance for coordination models.

Notice that amongst the many papers exploiting embeddings and encoding techniques to compare coordination models, the closest in spirit to our own is [4]. There, an embedding technique called *architectural embedding* is used to map a coordination model on top of an architecture, and is used to analyze the implementability of different coordination models on three kinds of architectures called *undelayed*, *globally delayed*, and *locally delayed*. The main difference with our paper is that the notion of implementability we tackle is more related to process refinement, and mostly concerns aspects of the coordination medium abstraction, while the approach in [4] is more concerned with properties of the interaction space. Indeed, relaxing our assumptions on the ordering of the interaction space, and integrating the notion of implementability promoted in [4] with our own is interesting and deserves further studying.

## 6.2 Conformance and equivalence by encoding

In this section, the general notion of encoding is exploited to provide a suitable definition of conformance and equivalence between

[2]We think that this problem is quite general: as a model becomes more and more oriented to run-time aspects concerning implementation, it may be natural to witness the vanishing of compositional properties. Indeed, this reflects a typical property of certain systems, e.g., coordination infrastructures are in general not compositional.

two models $\mathbb{X} = \langle \mathcal{X}, \longrightarrow_{\mathcal{X}}, \mathcal{X}_0 \rangle$ and $\mathbb{Y} = \langle \mathcal{Y}, \longrightarrow_{\mathcal{Y}}, \mathcal{Y}_0 \rangle$.

Since we want to apply the notion of conformance on both the CaL and CaS frameworks, we allow some flexibility on how system evolutions in a model are simulated by the other. Not only a single transition in $\mathbb{Y}$ is to be simulated by one or more transitions of $\mathbb{X}$, but it may be the case that a transition in $\mathbb{Y}$ is associated to no transitions in $\mathbb{X}$. For instance, a sequence of transitions of $\mathbb{Y}$ – say with actions $a_1, a_2, \ldots, a_n$ – may be logically associated to just one of $\mathbb{X}$ (as shown in next section). In this case, a reasonable encoding may associate only one transition in that sequence – say e.g. the latter $a_n$ – to the transition of $\mathbb{X}$, while the others to none. To cope with this problem, we use an encoding of the kind:

$$Y \longrightarrow_{\mathcal{Y}} Y' \quad \Rightarrow \quad \left|Y\right| \longrightarrow^*_{\mathcal{X}} \left|Y'\right|$$

mapping a transition of $\mathbb{Y}$ to zero or more transitions of $\mathbb{X}$ (denoted by $\longrightarrow^*$). We say that an encoding satisfying this property is *compatible with respect to the operational semantics*.

However, now we also need to prevent those cases where the encoding makes all the elements of $\mathcal{Y}$ collapse into a unique element in $\mathcal{X}$, that is, generally speaking, we need our encoding to preserve a certain degree of injectivity. Since we want to do this in a quite general way, we borrow the observables technique from embeddings, so that preserving a suitable notion of observables may ensure – along with the above compatibility with respect to operational semantics – that encodings properly deal with our conformance notion.

The notion of observable we use here is that of successful termination, that is, the encoding should preserve information on whether all the coordinated entities have successfully terminated, and are not waiting for some blocking primitive to be executed. On the one hand, this notion is independent of the actual internal definition of a coordination medium, hence it also allows us to compare models providing different implementations of a coordination medium. On the other hand, it guarantees that the final result of a coordinated system's evolution is correctly preserved, preventing the encoding from erasing relevant information about the system dynamics allowed. It is worth noting that this notion of observables is quite similar to that used in the modular embeddings of [5], which however also distinguishes the case of unsuccessful termination.

So, we also consider both models $\mathbb{X}$ and $\mathbb{Y}$ equipped with a notion of successful termination, denoting by $\mathcal{T}_{\mathcal{X}} \subseteq \mathcal{X}$ and $\mathcal{T}_{\mathcal{Y}} \subseteq \mathcal{Y}$ the set of successful termination states of $\mathbb{X}$ and $\mathbb{Y}$.

DEFINITION 1. *$\mathbb{Y}$ is said to conform to $\mathbb{X}$, written $\mathbb{Y} \prec_\lambda \mathbb{X}$ if there exists an encoding $\left|.\right|_{\mathcal{X}} \in \mathcal{Y} \mapsto \mathcal{X}$, so that the two following properties hold:*

- $Y \longrightarrow_{\mathcal{Y}} Y' \quad \Rightarrow \quad \left|Y\right|_{\mathcal{X}} \longrightarrow^*_{\mathcal{X}} \left|Y'\right|_{\mathcal{X}}$
  *(compatibility w.r.t. operational semantics)*
- $Y \in \mathcal{T}_{\mathcal{Y}} \Leftrightarrow \left|Y\right|_{\mathcal{X}} \in \mathcal{T}_{\mathcal{X}}$
  *(preservation of termination success)*

It is straightforward to see that the relation $\prec_\lambda$ is a preorder on structures $\langle \langle \mathcal{X}, \longrightarrow_{\mathcal{X}}, \mathcal{X}_0 \rangle, \mathcal{T}_{\mathcal{X}} \rangle$, that is, it is transitive and reflexive. We naturally extend this notion by defining equivalence between two models.

DEFINITION 2. *$\mathbb{X}$ is said to be equivalent to $\mathbb{Y}$, written $\mathbb{X} \approx_\lambda \mathbb{Y}$, if $\mathbb{X}$ and $\mathbb{Y}$ mutually conform to each other, that is, $\mathbb{X} \prec_\lambda \mathbb{Y}$ and $\mathbb{Y} \prec_\lambda \mathbb{X}$.*

Again, it is simple to show that $\approx_\lambda$ is an equivalence relation, that is, it is transitive, reflexive, and symmetric.

# 7. THE LINDA COORDINATION MEDIUM

In this section we present a formalization of LINDA model in the CaS framework that is proved to be equivalent to the CaL model of LINDA presented in Section 4. Indeed, this not only shows an application of previous definitions, but it is also meant to provide some hints on how a formalization in the CaL framework can be generally turned into an equivalent coordination medium by taking into account implementation issues.

## 7.1 Definition

As a first step, we analyze the primitives allowed by the LINDA coordination model, and the kind of interaction schema they rely on. Requests are invocations of LINDA primitives, so the set $Req$ coincides with the set of primitives $\alpha$; on the other hand, since here we do not deal with predicative queries $inp$ and $rdp$, only positive reply to $in$ and $rd$ is considered, which is denote by symbol $ok$, so that $Rep = \{ok\}$. The set of states of the coordinated entities is modeled by the syntax $Q^F ::= 0 \mid \alpha.Q^F \mid ok.Q^F$, with operational semantics defined by rules:

$$\alpha.Q^F \xrightarrow{\uparrow\alpha}_{\mathcal{Q}} Q^F \qquad ok.Q^F \xrightarrow{\downarrow ok}_{\mathcal{Q}} Q^F$$

In order to properly describe the synchronicity/blocking aspects of Linda primitives, then, we suppose the set of initial states $\mathcal{Q}_0$ of a coordinated entity to have elements defined as:

$$Q_0 ::= 0 \mid in(x).ok.Q_0 \mid rd(x).ok.Q_0 \mid out(x).Q_0$$

In the transition system $\mathbb{Q} = \langle \mathcal{Q}, Act_{\mathcal{Q}}, \rightarrow_{\mathcal{Q}}, \mathcal{Q}_0 \rangle$ defining the behavior of a coordinated entity, then, we define $\mathcal{Q}$ as the set of states $Q^F$ that are reachable by elements in $\mathcal{Q}_0$ through a finite sequence of transitions $\longrightarrow_{\mathcal{Q}}$. In this way, we obtain that coordinated entities are always well formed with respect to the expected interaction protocol of primitives – e.g., situations like $ok.ok.0$ or $in(x).rd(x).0$ never occurs.

As second step, we provide a LINDA coordination medium $\mathbb{M}^L = \langle \mathcal{M}^L, \longrightarrow_{\mathcal{ML}}, Act_M, \mathcal{M}^L_0 \rangle$ that is meant to represent all and only the system evolutions described by the CaL formalization reported in Section 4. This is described by the algebra:

$$M^L ::= 0 \mid e^{\uparrow} \mid e^{\downarrow} \mid x \mid M^L \parallel M^L$$

so that at any time the state of a medium is a finite composition of pending requests $e^{\uparrow}$, pending replies $e^{\downarrow}$, and tuples $x$. Transition relation $\longrightarrow_{\mathcal{ML}}$ is defined by rules:

$$M^L \xrightarrow{e^{\uparrow}}_{\mathcal{ML}} M^L \parallel e^{\uparrow} \qquad \text{[S-REQ]}$$

$$M^L \parallel e^{\downarrow} \xrightarrow{e^{\downarrow}}_{\mathcal{ML}} M^L \qquad \text{[S-REP]}$$

$$id{\uparrow}out(x) \parallel M^L \xrightarrow{\tau}_{\mathcal{ML}} M^L \parallel x \qquad \text{[S-OUT]}$$

$$id{\uparrow}rd(x) \parallel M^L \parallel x \xrightarrow{\tau}_{\mathcal{ML}} M^L \parallel x \parallel id{\downarrow}ok \qquad \text{[S-RD]}$$

$$id{\uparrow}in(x) \parallel M^L \parallel x \xrightarrow{\tau}_{\mathcal{ML}} M^L \parallel id{\downarrow}ok \qquad \text{[S-IN]}$$

Rules [S-REQ] and [S-REP] can be seen as standard: they mean that requests and replies are handled asynchronously. For the particular case of our LINDA system, furthermore, rule [S-REQ] implicitly defines unordering of operation $out$, in that an $out$ request remains pending until eventually evaluated. Rules [S-OUT], [S-RD], and [S-IN] have a one-to-one, syntactical mapping with rules [L-INS], [L-RD], and [L-IN] of the LINDA CaL formalization: simply, instead of allowing the process continuation to carry on, they just reify the reply. As initial state for this medium we suppose $\mathcal{M}^L_0 = \{0\}$.

Finally the CaS specification of the LINDA model is simply given by $\mathbb{S} = \mathbb{M}^L \otimes \mathbb{C}^L$, where $\mathbb{C}^L$ is the coordinated space obtained

$$\left|\prod_{i\in I} id_i \uparrow \beta_i \,\|\, \prod_{j\in J} id_j \uparrow out(x_j) \,\|\, \prod_{u\in U} e_u^{\downarrow} \,\|\, \prod_{h\in H} x_h \otimes \prod_{v\in I\uplus J\uplus K} \langle id_v, Q_v\rangle\right|_{\mathcal{L}} = \prod_{i\in I} \left|\beta_i.Q_i\right|_{\mathcal{L}} \,\|\, \prod_{j\in J}\langle x_j\rangle \,\|\, \prod_{h\in H} x_h \,\|\, \prod_{k\in K} \left|Q_k\right|_{\mathcal{L}}$$

$$\left|0\right|_{\mathcal{L}} = 0 \qquad \left|\alpha.Q\right|_{\mathcal{L}} = \alpha.\left|Q\right|_{\mathcal{L}} \qquad \left|ok.Q\right|_{\mathcal{L}} = \left|Q\right|_{\mathcal{L}}$$

---

$$\left|\prod_{i\in I} P_i \,\|\, \prod_{j\in J}\langle x_j\rangle \,\|\, \prod_{k\in K} x_k\right|_{\mathcal{S}} = \prod_{j\in J}\langle id_0, x_j\rangle \,\|\, \prod_{k\in K} x_k \otimes \prod_{i\in I}\langle id_i, \left|P_i\right|_{\mathcal{S}}\rangle$$

$$\left|0\right|_{\mathcal{S}} = 0 \qquad \left|out(x).P\right|_{\mathcal{S}} = out(x).\left|P\right|_{\mathcal{S}} \qquad \left|rd(x).P\right|_{\mathcal{S}} = rd(x).ok.\left|P\right|_{\mathcal{S}} \qquad \left|in(x).P\right|_{\mathcal{S}} = in(x).ok.\left|P\right|_{\mathcal{S}}$$

**Figure 1: Mutual encodings of Linda-CaS and Linda-CaL**

by the above defined coordinated entities $\mathbb{Q}$. Notice that our hypothesis on the initial state of coordinated entities and coordination medium allow us to suppose some trivial sanity condition, such as that there cannot occur two pending requests coming from the same coordinated entity, or replies with no corresponding entity waiting for their consumption, and so on.

## 7.2 Encoding

In this section, this formalization of a Linda coordinated system is shown to be equivalent to the one presented in Section 4. We first define a suitable notion of successful termination states for $\mathbb{L}$ and $\mathbb{S}$, simply as:

$$\mathcal{T}_{\mathcal{L}} = \{\prod_i x_i\} \qquad \mathcal{T}_{\mathcal{S}} = \{\prod_i x_i \otimes \prod_j \langle id_j, 0\rangle\}$$

Both cases characterize states where coordinated entities are terminated and no pending requests are to be treated, that is, the state of the coordinated system is only made of tuples.

Then, we define the encodings $\left|.\right|_{\mathcal{L}} \in \mathcal{S} \mapsto \mathcal{L}$ and $\left|.\right|_{\mathcal{S}} \in \mathcal{L} \mapsto \mathcal{S}$ as reported in Figure 1. In these definitions, the allowed sets of indexes $I, J, U, H, K$ are those for which the encodings make sense, that is, for which the encoded state is actually in $\mathcal{S}$ and $\mathcal{L}$, respectively.

The upside part of the figure contains the encoding from the CaS framework to CaL. Coordinated entities $Q$ are simply encoded into LINDA processes $P$ by erasing their identifier and forgetting about reception of replies. By doing that, however, any pending request $id \uparrow \beta$ must rebuild the $in$ or $rd$ operation in the requesting entity $id$, since that operation is not to be considered already executed, but just requested. On the other hand, any pending $out(x)$ is encoded into a pending datum $\langle x\rangle$, pending reply events are simply dropped, and tuples are simply encoded into tuples. Symbol $\uplus$ in the definition of the encoding is used for disjoint union.

The downside part of the figure describes the opposite encoding from CaL to CaS. Processes $P$ are translated into entities $Q$ by adding a fresh identifier (all $id_i$ are supposed to be different from each other), and by inserting reply operation $ok$ after each $rd$ and $in$. Pending data are encoded into pending requests for $out$ sent by a dummy coordinated entity $id_0$, and tuples are encoded into tuples. As an example, the CaS model:

$$x \,\|\, id' \uparrow out(x') \,\|\, id'' \uparrow rd(x'') \otimes \langle id, in(x).ok.0\rangle \,\|\, \langle id', 0\rangle \,\|\, \langle id'', 0\rangle$$

is translated by $\left|.\right|_{\mathcal{L}}$ into the CaL model:

$$x \,\|\, \langle x'\rangle \,\|\, in(x).0 \,\|\, rd(x'').0$$

that is translated by $\left|.\right|_{\mathcal{S}}$ back to:

$$x \,\|\, id_0 \uparrow out(x') \;\otimes\; \langle id_1, in(x).ok.0\rangle \,\|\, \langle id_2, rd(x'').ok.0\rangle$$

Notice that the latter system can be essentially obtained from the starting one after applying a [S-REQ] transition, which has no actual counterpart in the CaL model. The validity of this encoding is stated by the following theorem:

THEOREM 1. *The* LINDA *CaL model* $\mathbb{L}$ *is* $\approx_\lambda$*-equivalent to the* LINDA *CaS model* $\mathbb{S}$.

*Proof Sketch.* It is easy to show that both encodings preserve the notion of successful termination. In fact, $\left|.\right|_{\mathcal{S}}$ transforms terminated processes $P$ into terminated processes $Q$, pending data into pending requests, and tuples to tuples, and similarly for $\left|.\right|_{\mathcal{L}}$. In particular, a pending request is reconstructed into an operation only to coordinated entities that wait to execute operation $ok$, i.e., not in the case of successful termination. Compatibility of $\left|.\right|_{\mathcal{S}}$ with respect to operational semantics is easily proved by showing that transitions [L-IN] and [L-RD] correspond to transition sequences [S-REQ] [S-IN] [S-REP] and [S-REQ] [S-RD] [S-REP], transition [L-OUT] to [S-REQ], and transition [L-INS] to [S-OUT], and vice versa for $\left|.\right|_{\mathcal{L}}$.

# 8. CONFORMANCE AND MEDIUM IMPLEMENTATION

The notion of conformance we promote here amounts to considering a model $\mathbb{Y}$ as conforming to a model $\mathbb{X}$ if all the system evolutions described by $\mathbb{Y}$ are allowed by $\mathbb{X}$. Roughly speaking, $\mathbb{Y}$ should allow for fewer evolutions than $\mathbb{X}$, so that $\mathbb{Y}$ can be seen as a more deterministic, more directly executable version of $\mathbb{X}$, that is, an implementation of the specification provided by $\mathbb{X}$.

Indeed, this notion resembles the idea of implementation as horizontal refinement [18]. The core part of a CaS specification is the description of the behavior of the coordination medium in terms of a transition system. Then, transition systems are already equipped with notions of equivalence and preorder, namely, according to the concept of *observable behavior* [18]. Each process described by the transition system is associated with the set of *observations* it allows: each observation provides some information about an allowed dynamics of the process. Correspondingly, a process is considered equivalent to another if it allows for the same set of observations. Moreover, a preorder on processes can also be introduced

to reflect a notion of refinement: a process is considered an implementation of another if it allows for fewer observations.

In particular, we are interested in studying how our notion of conformance fits this idea of process implementation. We take as the notion of observation semantics, *weak trace* (WT) semantics [6, 18], where a process observation is any sequence of actions it can execute (i.e., trace semantics), without considering silent actions (i.e., weak semantics). So, two processes are considered WT-equivalent if they allow for the same sequences of actions modulo the occurrence of silent $\tau$ actions. To this end, given any sequence of actions $a^*$, also called *trace*, we denote by $a^*|_\tau$ the sequence obtained by dropping any $\tau$ action from it. Given a transition system $\langle \mathcal{X}, \longrightarrow_{\mathcal{X}}, Act_X, \mathcal{X}_0 \rangle$, process $X$ is said to be a WT-refinement of $X'$ if we have:

$$X \xrightarrow{a^*}_{\mathcal{X}} X_F \;\Rightarrow\; \exists b^*, X'_F :\; X' \xrightarrow{b^*}_{\mathcal{X}} X'_F \text{ and } a^*|_\tau = b^*|_\tau$$

that is, for any trace $a^*$ moving $X$ to $X_F$ there is at least a trace $b^*$ of $X'$ (moving $X'$ to some $X'_F$) so that $a^*$ and $b^*$ are equal modulo occurrences of $\tau$[3].

Our choice of weakness is motivated by the fact that implementations typically need to execute more $\tau$ actions in order to realize a given specification (see e.g. the approach based on weak equivalence in [22]). Trace semantics has been chosen instead of others such as bisimulation because of its simplicity: it is in fact one of the larger (i.e., weaker) notions of equivalence. Notice that the discussion we provide here applies to any stronger observation semantics, such as e.g. bisimulation.

We state that the notion of (weak trace) preorder for coordination media implies our notion of conformance, namely, in two systems $\mathbb{X} = \mathbb{M}^{\mathbb{X}} \otimes \mathbb{C}$ and $\mathbb{Y} = \mathbb{M}^{\mathbb{Y}} \otimes \mathbb{C}$, if $\mathbb{M}^{Y}$ is a refinement of $\mathbb{M}^{X}$, then $\mathbb{Y}$ conforms to $\mathbb{X}$. Roughly speaking, by refining a medium implementation $\mathbb{M}$ we remain conforming to the original coordination model of $\mathbb{M}$.

First of all, we need to adapt the definition of preorder to our framework, considering the general case of two media $\mathbb{M}^{\mathbb{X}} = \langle \mathcal{M}^{\mathbb{X}}, \longrightarrow_{\mathcal{MX}}, Act_{\mathcal{M}}, \mathcal{M}_0^{\mathbb{X}} \rangle$ and $\mathbb{M}^{\mathbb{Y}} = \langle \mathcal{M}^{\mathbb{Y}}, \longrightarrow_{\mathcal{MY}}, Act_{\mathcal{M}}, \mathcal{M}_0^{\mathbb{Y}} \rangle$ with different sets of states, but clearly with the same set of actions (since they should be composed to the same coordinated space $\mathbb{C}$. We are interested in a global notion of preorder between $\mathbb{M}^{\mathbb{Y}}$ and $\mathbb{M}^{\mathbb{X}}$, and not between any two states of them. So, we suppose that each coordination media has only one initial state, namely $0^{\mathbb{X}}_{\mathbb{M}} \in \mathcal{M}^{\mathbb{X}}$ and $0^{\mathbb{Y}}_{\mathbb{M}} \in \mathcal{M}^{\mathbb{Y}}$, representing an initial situation where no coordination patterns have been set up by coordinated entities. We believe that this notion is widely applicable to coordination models, e.g. in the case of LINDA (and tuple spaces in general) this naturally amounts to a medium without tuples nor pending requests.

DEFINITION 3. *Given the two coordination media $\mathbb{M}^{\mathbb{X}}$ and $\mathbb{M}^{\mathbb{Y}}$, $\mathbb{M}^{\mathbb{Y}}$ is said to be a refinement of $\mathbb{M}^{\mathbb{X}}$, written $\mathbb{M}^{\mathbb{Y}} \prec_\sigma \mathbb{M}^{\mathbb{X}}$, if:*

$$0^{\mathbb{Y}}_{\mathbb{M}} \xrightarrow{a^*}_{\mathcal{MY}} M^{\mathbb{Y}} \Rightarrow \exists b^*, M^{\mathbb{X}} :\; 0^{\mathbb{X}}_{\mathbb{M}} \xrightarrow{b^*}_{\mathcal{MX}} M^{\mathbb{X}} \text{ and } a^*|_\tau = b^*|_\tau$$

that is, for any trace $a^*$ of $0^{\mathbb{Y}}_{\mathbb{M}}$ in $\mathbb{M}^{\mathbb{Y}}$ there is a trace $b^*$ of $0^{\mathbb{X}}_{\mathbb{M}}$ in $\mathbb{M}^{\mathbb{X}}$ so that $a^*$ and $b^*$ are equal modulo occurrences of $\tau$. Then, the main result we obtain is as follows:

THEOREM 2. *Given the coordination media $\mathbb{M}^{\mathbb{X}}$ and $\mathbb{M}^{\mathbb{Y}}$ and a compatible coordinated space $\mathbb{C}$, we have:*

$$\mathbb{M}^{\mathbb{Y}} \prec_\sigma \mathbb{M}^{\mathbb{X}} \quad \Rightarrow \quad \mathbb{M}^{\mathbb{Y}} \otimes \mathbb{C} \prec_\lambda \mathbb{M}^{\mathbb{X}} \otimes \mathbb{C}$$

[3]In this formal definition, the syntax $X \xrightarrow{a^*} X'$ with $a^* = a_0, a_1, \ldots, a_n$ naturally means $X \xrightarrow{a_0} \xrightarrow{a_1} \ldots \xrightarrow{a_n} X'$.

which states that preorder for coordination media implies conformance of the corresponding coordination models.

*Proof.* Let $\mathbb{M}^{\mathbb{X}} = \langle \mathcal{M}^{\mathbb{X}}, \longrightarrow_{\mathcal{MX}}, Act_{\mathcal{M}}, \{0^{\mathbb{X}}_{\mathbb{M}}\} \rangle$ and $\mathbb{M}^{\mathbb{Y}} = \langle \mathcal{M}^{\mathbb{Y}}, \longrightarrow_{\mathcal{MY}}, Act_{\mathcal{M}}, \{0^{\mathbb{Y}}_{\mathbb{M}}\} \rangle$. For hypothesis, for any $M^{\mathbb{Y}} \in \mathcal{M}^{\mathbb{Y}}$ there exists at least one $M^{\mathbb{X}} \in \mathcal{M}^{\mathbb{X}}$ so that:

$$0^{\mathbb{Y}}_{\mathbb{M}} \xrightarrow{a^*}_{\mathcal{MY}} M^{\mathbb{Y}} \quad \text{and} \quad 0^{\mathbb{X}}_{\mathbb{M}} \xrightarrow{b^*}_{\mathcal{MX}} M^{\mathbb{X}} \quad \text{and} \quad a^*|_\tau = b^*|_\tau$$

We define an encoding $\big|.\big|^{\mu}_{\mathcal{X}} \in \mathcal{M}^{\mathbb{Y}} \mapsto \mathcal{M}^{\mathbb{X}}$ so that an element $M^{\mathbb{Y}}$ is assigned to any of these $M^{\mathbb{X}}$. We prove that the encoding $\big|.\big|^{\lambda}_{\mathcal{X}}$ from $\mathbb{M}^{\mathbb{Y}} \otimes \mathbb{C}$ to $\mathbb{M}^{\mathbb{X}} \otimes \mathbb{C}$ defined as:

$$\big|M^{\mathbb{Y}} \otimes C\big|^{\lambda}_{\mathcal{X}} = \big|M^{\mathbb{Y}}\big|^{\mu}_{\mathcal{X}} \otimes C$$

satisfies the two properties of Definition 1, from which the thesis follows.

Preservation of termination states holds since:

$$\big|M^{\mathbb{Y}}\big|^{\mu}_{\mathcal{X}} = 0^{\mathbb{X}}_{\mathbb{M}} \Leftrightarrow M^{\mathbb{Y}} = 0^{\mathbb{Y}}_{\mathbb{M}}$$

so that successful termination states $0^{\mathbb{Y}}_{\mathbb{M}} \otimes C$ are encoded into successful termination states $0^{\mathbb{X}}_{\mathbb{M}} \otimes C$ (and unsuccessful ones to unsuccessful ones).

To prove compatibility with respect to operational semantics, suppose:

$$M^{\mathbb{Y}} \otimes C \longrightarrow^*_{\mathcal{Y}} M^{\mathbb{Y}}_F \otimes C'$$

is a generic evolution of $\mathbb{Y}$. From the definition of $\longrightarrow_{\mathcal{Y}}$ we have:

$$M^{\mathbb{Y}} \xrightarrow{a^*}_{\mathcal{MY}} M^{\mathbb{Y}}_F, \qquad C \xrightarrow{c^*}_{\mathcal{C}} C'$$

for some $a^*, c^*$ so that $a^*|_\tau = c^*|_\tau$, because each non-silent action of $\mathbb{M}^{\mathbb{Y}}$ is associated with the same action in $\mathbb{C}$. On the other hand, from $\mathbb{M}^{\mathbb{Y}} \prec_\sigma \mathbb{M}^{\mathbb{X}}$ and for the definition of $\big|.\big|^{\mu}_{\mathcal{X}}$, we have:

$$\begin{array}{llll} 0^{\mathbb{Y}}_{\mathbb{M}} & \xrightarrow{a'^*}_{\mathcal{MY}} & M^{\mathbb{Y}} & \xrightarrow{a^*}_{\mathcal{MY}} \; M^{\mathbb{Y}}_F \\ 0^{\mathbb{X}}_{\mathbb{M}} & \xrightarrow{b'^*}_{\mathcal{MX}} & \big|M^{\mathbb{Y}}\big|^{\mu}_{\mathcal{X}} & \xrightarrow{b^*}_{\mathcal{MX}} \; \big|M^{\mathbb{Y}}_F\big|^{\mu}_{\mathcal{X}} \end{array}$$

for some $a'^*, b'^*, b^*$ so that, in particular, $a^*|_\tau = b^*|_\tau$. As a result, by composing $\mathbb{M}^{\mathbb{X}}$ (obtained by encoding $\mathbb{M}^{\mathbb{Y}}$) and $\mathbb{C}$ we have:

$$\big|M^{\mathbb{Y}}\big|^{\mu}_{\mathcal{X}} \otimes C \longrightarrow_{\mathcal{X}} \big|M^{\mathbb{Y}}_F\big|^{\mu}_{\mathcal{X}} \otimes C'$$

from which

$$\big|M^{\mathbb{Y}} \otimes C\big|^{\lambda}_{\mathcal{X}} \longrightarrow_{\mathcal{X}} \big|M^{\mathbb{Y}}_F \otimes C'\big|^{\lambda}_{\mathcal{X}}$$

follows for the definition of $\big|.\big|^{\lambda}_{\mathcal{X}}$. The compatibility of $\big|.\big|^{\lambda}_{\mathcal{X}}$ with respect to the operational semantics is then proved.

# 9. CONCLUSIONS AND FUTURE WORKS

Coordination models and languages promote a strong methodology for building today's distributed systems. They call for the encapsulation of the intrinsic complexity of system interactions into well defined coordination abstractions, amenable to an explicit design and supporting the implementation of coordination infrastructures at each step of the development process. This goal is so crucial that requires a profound understanding of the semantics of a coordination model and of the properties of its proposed implementations.

We believe that the framework built up in this paper can be a suitable tool for reasoning about this issue. Our analysis is based on a notion of conformance for coordination models, stating adequacy of a coordination medium with respect to some coordination laws. In particular, by exploiting the LINDA use case, we show how

to define a coordination medium equivalent to given coordination laws.

Several future works can be developed to deepen the investigation of our framework. It would be interesting to deepen the comparison of our notion of conformance with other forms of embeddings and encodings exploited for comparing the expressiveness of coordination models [5, 4]. Then, studying the conformance issue of other, more complex coordination models such as JavaSpaces would provide interesting case studies to further evaluate our approach. Finally, one of our main future works is to evaluate the applicability of the formal framework developed in this paper as a means to devise an engineering methodology for coordination models in distributed systems.

## 10. REFERENCES


[1] F. Arbab, I. Herman, and P. Spilling. An overview of MANIFOLD and its implementation. *Concurrency: Practice and Experience*, 5(1):23–70, February 1993.

[2] J. A. Bergstra, A. Ponse, and S. A. Smolka, editors. *Handbook of Process Algebra*. North-Holland, 2001.

[3] M. M. Bonsangue, F. Arbab, J. W. de Bakker, J. J. M. M. Rutten, A. Scutella, and G. Zavattaro. A transition system semantics for the control-driven coordination language MANIFOLD. *Theoretical Computer Science*, 240(1):3–47, June 2000.

[4] M. M. Bonsangue, J. N. Kok, and G. Zavattaro. Comparing coordination models and architectures using embeddings. Science of Computer Programming, to appear, 2002.

[5] A. Brogi and J. Jacquet. On the expressiveness of coordination models. In P. Ciancarini and A. L. Wolf, editors, *Coordination Languages and Models*, volume 1594 of *LNCS*, pages 134–149. Springer-Verlag, 1999.

[6] M. Broy and E.-R. Olderog. Trace-oriented models of concurrency. In Bergstra et al. [2], chapter 2, pages 101–195.

[7] N. Busi, R. Gorrieri, and G. Zavattaro. Three semantics of the output operation for asynchronous communication. In *Coordination Languages and Models*, volume 1282 of *LNCS*, pages 205–219. Springer-Verlag, 1997.

[8] N. Busi, R. Gorrieri, and G. Zavattaro. A process algebraic view of Linda coordination primitives. *Theoretical Computer Science*, 192(2):167–199, 1998.

[9] N. Busi, R. Gorrieri, and G. Zavattaro. On the expressiveness of Linda coordination primitives. *Information and Computation*, 156(1-2):90–121, Jan. 2000.

[10] N. Busi, R. Gorrieri, and G. Zavattaro. Process calculi for Coordination: From Linda to JavaSpaces. In T. Rus, editor, *Algebraic Methodology and Software Technology. 8th International Conference, AMAST 2000*, volume 1816 of *LNCS*, pages 198–212. Springer-Verlag, 2000.

[11] N. Busi and G. Zavattaro. On the serializability of transactions in JavaSpaces. In U. Montanari and V. Sassone, editors, *ConCoord: International Workshop on Concurrency and Coordination*, volume 54 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science, July 2001.

[12] P. Ciancarini. Coordination models and languages as software integrators. *ACM Computing Surveys*, 28(2):300–302, June 1996.

[13] F. S. de Boer and C. Palamidessi. Embedding as a tool for language comparison. *Information and Computation*, 108(1):128–157, 1994.

[14] L. Fiege, G. Mühl, and F. C. Gärtner. A modular approach to build structured event-based systems. In *2002 ACM Symposium on Applied Computing (SAC) 2002*, pages 385–392, Madrid, Spain, 2002. ACM.

[15] E. Freeman, S. Hupfer, and K. Arnold. *JavaSpaces: Principles, Patterns, and Practice*. Addison-Wesley, 1999.

[16] D. Gelernter. Generative communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7(1):80–112, January 1985.

[17] D. Gelernter and L. Zuck. On what Linda is: Formal description of Linda as a reactive system. In *Coordination Languages and Models*, volume 1282 of *LNCS*, pages 187–219, Berlin (Germany), September 1997. Springer-Verlag.

[18] R. v. Glabbeek. The linear time – branching time spectrum I. The semantics of concrete, sequential processes. In Bergstra et al. [2], chapter 1, pages 3–100.

[19] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[20] A. Omicini. On the semantics of tuple-based coordination models. In *1999 ACM Symposium on Applied Computing (SAC'99)*, San Antonio (TX), 28 Feb. – 2 Mar. 1999.

[21] A. Omicini and E. Denti. From tuple spaces to tuple centres. *Science of Computer Programming*, 41(3), 2001.

[22] S. Orzan and J. van de Pol. Distribution of a simple shared dataspace architecture. In *1st International Workshop on Foundations of Coordination Languages and Software Architectures*, volume 68(3) of *Electronic Notes in Theoretical Computer Science*. Elsevier Science B. V., 2002.

[23] G. P. Picco, A. L. Murphy, and G.-C. Roman. LIME: Linda meets mobility. In *Proceedings of the 1999 International Conference on Software Engineering (ICSE'99)*, pages 368–377. ACM, 1999. May 16-22, Los Angeles (CA), USA.

[24] G. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Department of Computer Science, AArhus University, Denmark, 1991.

[25] A. Rowstron. Optimising the Linda in primitive: Understanding tuple space run-times. In *Proceedings of the 2000 ACM Symposium on Applied Computing (SAC 2000)*, pages 227–232, Como (I), 19–21 March 2000. ACM.

[26] M. Viroli, G. Moro, and A. Omicini. On observation as a coordination paradigm: An ontology and a formal framework. In *16th ACM Symposium on Applied Computing (SAC 2001)*, pages 166–175, Las Vegas (NV), 11–14 Mar. 2001. ACM. Track on Coordination Models, Languages and Applications.

[27] M. Viroli and A. Omicini. Coordination as a service: Ontological and formal foundation. In *1st International Workshop on Foundations of Coordination Languages and Software Architectures*, volume 68(3) of *Electronic Notes in Theoretical Computer Science*. Elsevier Science B. V., 2002.

[28] M. Viroli and A. Omicini. Tuple-based models in the observation framework. In *Coordination Languages and Models*, volume 2315 of *LNCS*, pages 364–379. Springer-Verlag, 2002.

[29] M. Viroli, A. Omicini, and A. Ricci. On the expressiveness of event-based coordination media. In H. R. Arabnia, editor, *International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'02)*, pages 1414–1420, Las Vegas, NV, USA, 24–27 July 2002. CSREA Press.