Architectural Interoperability Checking in Visual Coordination Networks [1]

David Šafránek [2]

Department of Computer Science, Faculty of Informatics
Masaryk University Brno, Czech Republic

Abstract

In this paper, the approach [1] of architectural interoperability checking is revisited and utilized for interoperability checking of architectures with connectors and components treated as behaviorally and expressively different elements of architecture description. By that way, the framework of architectural interoperability checking for a diagrammatic language Visual Coordination Diagrams is established.

1 Introduction

The hierarchical component-based system design can be underlied with precise operational semantics which allows to join the traditional software and hardware design methods with the formal methods known from the theory of process algebras (e.g., refinement, equivalence or model checking). Additionally, syntactical and semantical separation of modeling the coordination aspects from modeling the behavioral aspects in architectural descriptions makes it possible to define a static communication infrastructure of a system independently of modeling the behavioral parts. In such a setting, it can be ensured by architectural description that the system under specification is correct by design concerning the interoperability correctness of component cooperation.

In this paper we present a formal architectural interoperability checking methodology for the diagrammatic language Visual Coordination Networks (VCN). This language is based on our previous work [14].

VCN employ an exogenous coordination model [5]. In such a model coordination aspects are semantically separated from computational aspects. VCN

[1] This work has been supported by the Grant Agency of Czech Republic grant No. 201/06/1338, the research center Institute for Theoretical Computer Science (ITI), project No. 1M0021620808, and the Academy of Sciences of CR grant No. 1ET408050503.

[2] Email: xsafran1@fi.muni.cz

can be viewed as static architecture diagrams specifying connections among components. To capture the coordination aspects, connectors — in VCN called buses, are introduced at the same level as computational components. By the notion of buses the dynamism of coordination model is modeled.

1.1 Related Work

Graphical calculus of communicating systems (GCCS) [6], its synchronous extension SGCCS [12], and Architectural Interaction Diagrams (AID) [10] are the previous work on this topic. To our best knowledge, for none of those languages interoperability aspects were studied.

In the community of coordination languages, there is a large group of languages which have properties of architectural languages. The most significant languages from this domain are Visifold and REO [4,9]. These languages support control-driven exogenous coordination and are execution-oriented in contrast to design-oriented VCN.

In the process of hardware and software design, it has appeared useful to treat the behavioral aspects separately from the architectural aspects of a system under design. Ideas of this aspect separation have been pioneered in the work on interface-based design [11] and they have been been further refined in theory of interface automata [7]. These principles are currently included in the recent version of the UML language.

1.1.1 Our Contribution

There is an exhaustive piece of work by Bernardo et al. on the topic of interoperability checking of architectural descriptions formalized in traditional process algebraic framework. In [3] it has been proved that for checking of an acyclic component topology it suffices to check interaction compatibility of all pairs of mutually connected components. The notion of such compatibility is based on a weak bisimilarity of the two components in a pair. Abstraction of both components is considered comprising only actions of mutual interaction, while all other actions are hidden. In [1] this methodology of compatibility checking has been extended to arbitrary network topologies which can include cyclic relationships among components.

We show that the similar approach can be applied to the behavioral model of VCN. In contrast to [1,3], dependency graph of an architecture in VCN is bipartite, as VCN distinguishes connectors and components as two semantically different members of the architecture. In other words, we have to define the notion of architectural compatibility between its two arbitrary adjacent nodes, which are always a connector and a component. Moreover, VCN introduces set-labeled transition operational semantics to capture behavioral model of connectors. This extension lifts the expressiveness of the coordination model. In this paper, the notion of architectural interoperability is revisited and extended to capture the needs of such a setting. Especially, there is no traditional notion of a parallel composition operator in VCN and therefore the congruence results applied in [1,3] cannot be directly employed here. However, the main result of this paper shows that the framework of [1] applies to some extent also to the VCN setting.

2 Visual Coordination Networks

2.1 Overview

At the most abstract level of view, the hierarchy of VCN introduces two separate layers — *computation layer* and *coordination layer*. The computation layer focuses on computational aspects of a system under design, while the coordination layer deals with interaction aspects. The principle of such a layered structure respects the nature of component-based system design, and is inspired by the work concerning software architecture description [2].

In our setting, the computation layer is treated as a low level layer of system specification, upon which the coordination layer rests. Thus from the designers point of view, both the top-down and the bottom-up design methodologies can be applied during system design using the VCN. On one hand, VCN language allows the computation layer to be taken as a supplementary layer, which can be added to the system hierarchy later during the design. On the other hand, the computation of components can be specified at first, while the coordination layer can be added later.

2.1.1 Coordination layer

The main idea of the VCN coordination layer, which revises the concept introduced in Wright [2], is to describe interaction aspects of a static component-based architecture. The notion of such a static structure concerns topology of concurrently running *components* permanently coordinated by specific *connectors*. This topology is determined by the point-to-point *links* which connect component interface ports to connectors. The particular VCN construct that represents such topologies of components and connectors is called *network*. An example of a network is depicted in Figure 1.

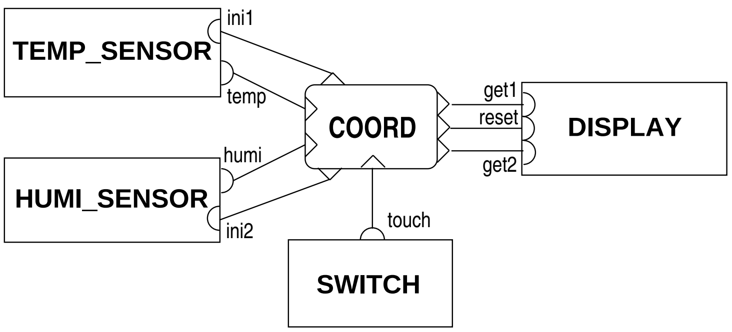

Fig. 1. A network of components coordinated by the *COORD* bus

Similarly to Wright, connectors are treated as first-class citizens (at the same level as components). In the VCN setting, connectors are called *buses*. Buses represent coordination mechanisms which control component interaction. The coordination model represented by buses is characterized by atomic actions called cooperations, which have the meaning of atomic synchronization of a set of some components in a particular network.

2.1.2 Computation layer

Component computation is described using VCN *leaves*. VCN leaves are cornerstones of the computation layer. In our setting, leaves are assumed to be abstract computation models of system components. More particularly, a leaf is an atomic element in the VCN structure and can be specified in an arbitrary formalism for formal description of reactive computation. It is assumed that the kind of formalism which can be used for this purpose is compatible with the used semantic model of VCN. In general, the potential set of such compatible formalisms includes any reactive computation description language which can be encoded into a labelled transition system.

2.1.3 Network hierarchy

The computation layer makes the bottom most level of the VCN hierarchy. It is determined by the set of all the leaves which are used in the particular system design. As it has been mentioned above, leaves directly represent computation of components and are interconnected by buses to form network topologies.

In a natural sense, we would like to view such a network topology of leaves and buses as a black box with complex behavior hidden inside (defined by computation of leaves coordinated by buses). More specifically, not only leaves represent computation, but also entire network topologies do that. This idea leads us to consider the notion of a component to be more abstract than is the notion of a leaf. In particular, either a leaf or a network can be sensed as a component in the VCN style of thinking. The possibility of taking a network as a component allows the coordination layer to have more levels of hierarchy.

3 VCN Structure and its Behavioral Model

We present here the simplified version of VCN language. In this simplification we focus on VCN networks in order to develop the framework for architectural interoperability checking.

3.1 Structural Terms

Similarly as in Statecharts-like formalisms [8], the semantics of VCN is defined by textual terms. In VCN we distinguish between structural terms, formalizing the system architecture, and behavioral terms, formalizing the behavioral model. The key elements of VCN are ports which form component interfaces.

Each port is identified by its label. A port can be sensed as a place on which events observed during the particular component computation occur. We distinguish two kinds of ports – input ports, on which the relevant events of the environment are captured, and output ports, on which the relevant events are emitted to the environment.

Definition 3.1 Fix $\mathcal{L}$ a countable set of *labels* and assume $\tau \notin \mathcal{L}$. Define *ports* as members of the set $\mathcal{P} \stackrel{\text{df}}{=} \mathcal{L} \times \{in, out\}$. Further define projections of $\mathcal{P}$, the set of *output ports* $\mathcal{W} \stackrel{\text{df}}{=} \{p \in \mathcal{P} \parallel p = \langle l, out\rangle, l \in \mathcal{L}\}$, and the set of *input ports* $\mathcal{R} \stackrel{\text{df}}{=} \{p \in \mathcal{P} \parallel p = \langle l, in\rangle, l \in \mathcal{L}\}$.

To avoid ambiguity of ports in the scope of network, we annotate each port with an index denoting the component to which it belongs. We assume that each component in a network is identified by a unique natural number. By the way of such annotation, the required port unambiguity in the scope of entire network topology is guaranteed. Moreover, unambiguity of ports connected to a specific bus is also guaranteed.

Definition 3.2 Define set of *annotated ports* $\mathcal{P}^\sharp \stackrel{\text{df}}{=} \{\langle p, i\rangle \parallel p \in \mathcal{P}, i \in \mathcal{N}\}$. Further define the respective projections as sets of *annotated output* and *annotated input ports*, $\mathcal{W}^\sharp \stackrel{\text{df}}{=} \{\langle w, i\rangle \parallel w \in \mathcal{W}, i \in \mathcal{N}\}$, and $\mathcal{R}^\sharp \stackrel{\text{df}}{=} \{\langle r, i\rangle \parallel r \in \mathcal{R}, i \in \mathcal{N}\}$.

Remark 3.3 For some $i \in \mathcal{N}$ we denote the set of ports annotated by i as $\mathcal{P}^{\sharp_i} \stackrel{\text{df}}{=} \{\langle p, i\rangle \parallel p \in \mathcal{P}\}$.

Members of the set $\mathcal{P}$ are usually represented by symbols $p, p_1, p_2, \ldots$, members of $\mathcal{W}$ by $w, w_1, w_2, \ldots$, and, finally, members of $\mathcal{R}$ by $r, r_1, r_2, \ldots$ Note the important fact that $\mathcal{W} \cap \mathcal{R} = \emptyset$. By the notation $p^{\sharp_i}$ for some $i \in \mathcal{N}$ we mean $\langle p, i\rangle \in \mathcal{P}^\sharp$. Thus an input port $r \in \mathcal{R}$ annotated by $i \in \mathcal{N}$, is denoted $r^{\sharp_i}$. The annotated output ports are denoted in the same way. Wherever the annotation number i is not important to be specified in the particular context for an annotated port $p^{\sharp_i}$, we omit the upper index '$\sharp i$' and write simply p.

For a specific set of (unannotated) ports $P \subseteq \mathcal{P}$, the set containing these ports all annotated by $i \in \mathcal{N}$ is denoted $P^{\sharp_i}$. For the set of ports of all annotation indices the notation $P^\sharp$, $P^\sharp \stackrel{\text{df}}{=} \bigcup_{i \in \mathcal{N}} P^{\sharp_i}$, is used.

Now we introduce a fundamental notion of the concept of buses – the notion of cooperations. A cooperation can be sensed as a group of ports on which the respective computational events are synchronized when the cooperation occurs.

Definition 3.4 Let $W^\sharp \subset \mathcal{W}^\sharp$ and $R^\sharp \subset \mathcal{R}^\sharp$ finite sets of ports. Define *cooperation* as the pair $\langle W^\sharp, R^\sharp\rangle$, denoted $\langle W^\sharp / R^\sharp\rangle$, satisfying:

$$(1) \quad \forall i, j \in \mathcal{N}, w_1, w_2 \in \mathcal{W}, r_1, r_2 \in \mathcal{R}.\, ({w_1}^{\sharp_i}, {w_2}^{\sharp_j} \in W^\sharp \Rightarrow i \neq j) \\ \wedge ({r_1}^{\sharp_i}, {r_2}^{\sharp_j} \in R^\sharp \Rightarrow\ i \neq j)$$

We say that a *port* p *is included in a cooperation* $c := \langle W^\sharp / R^\sharp\rangle$ and write $p \in c$ if either $p \in W^\sharp$ or $p \in R^\sharp$. The set of all cooperations is denoted Coops.

The condition (1) ensures the natural requirement that at most one input port or one output port of each component can be involved in a cooperation.

Definition 3.5 Define *bus* B as a finite set of cooperations $B \subset \text{Coops}$ such that $B \neq \emptyset$.

Countable set of all buses is denoted Buses. Members of Buses are typically denoted $B, B_1, B_2, \ldots$

Input of the bus B is denoted $In(B)$ and defined as the set $In(B) \stackrel{\text{df}}{=} \bigcup\{W^\sharp \parallel \langle W^\sharp / R^\sharp \rangle \in B\}$. Similarly, *output of the bus* B is denoted $Out(B)$ and defined as the set $Out(B) \stackrel{\text{df}}{=} \bigcup\{R^\sharp \parallel \langle W^\sharp / R^\sharp \rangle \in B\}$.

Definition 3.6 Define the *link relation* L as the relation $L \subset \mathcal{P}^\sharp \times \text{Buses}$ satisfying for each $B \in \text{Buses}$ all the following conditions:

(2) $\forall i, j \in \mathcal{N}, {p_1}^{\sharp_i}, {p_2}^{\sharp_j} \in \mathcal{P}^\sharp, \langle {p_1}^{\sharp_i}, B \rangle \in L.\ \langle {p_2}^{\sharp_j}, B \rangle \in L \Rightarrow p_1 \neq p_2 \vee i \neq j$

$\forall p \in \mathcal{P}^\sharp.\ \langle p, B \rangle \in L \Rightarrow \exists c \in B.\ p \in c$

(3) $\forall w \in \mathcal{W}^\sharp.\ w \in In(B) \Rightarrow \langle w, B \rangle \in L$

$\forall r \in \mathcal{R}^\sharp.\ r \in Out(B) \Rightarrow \langle r, B \rangle \in L$

Denote the set of all links of the bus B as $links(B, L)$ and define it as $links(B, L) \stackrel{\text{df}}{=} \{l \parallel \exists p \in \mathcal{P}^\sharp.\ l = \langle p, B \rangle \in L\}$.

The condition (2) in the definition above ensures that at most one link can be defined for a particular component interface port. The set of conditions (3) guarantees consistency of embedding a particular bus into a particular link relation.

Remark 3.7 For a particular link $l \in L$ of some link relation L we denote its port as $port(l) \stackrel{\text{df}}{=} p$, where $l \equiv \langle p, B \rangle$. Moreover, the set of all ports included in links of L is denoted $ports(L)$, $ports(L) \stackrel{\text{df}}{=} \{port(l) \parallel l \in L\}$.

Definition 3.8 Define the set of *structural terms*, denoted $\mathbf{T}_{st}$, as the least set satisfying:

(i) $A \in \mathbf{T}_{st}$ is a *leaf*

(ii) $N \in \mathbf{T}_{st}$, if N is a *network* defined as a tuple $N \stackrel{\text{df}}{=} \langle \bar{C}, \bar{B}, L \rangle$ where
- $\bar{C} = \langle C_1, \ldots, C_n \rangle$ for some $n > 0$ is a tuple of component terms, for each $i \in \{1, ..., n\}$, $C_i = \langle S_i, I_i, G_i \rangle$ is a component satisfying:
  (a) $I_i \subset \mathcal{P}$ finite set of ports is an *interface*, also denoted as $I(C_i)$
  (b) $S_i \in \mathbf{T}_{st}$
  (c) G_i is a *gate* which maps the component body S_i to the interface I_i
- $\bar{B} = \langle B_1, \ldots, B_m \rangle$ for some $m \geq 0$ is a tuple of buses
- L a link relation satisfying

$$L \subseteq \bigcup_{i \in \{1,...,n\}} I_i^{\sharp_i} \times \{B_i \parallel i \in \{1, ..., m\}\}$$

Additionally, we say that some port p is a *free port* in the network N if it has no link attached.

Remark 3.9 We denote $Llinks(B, C)$ the set of all links connecting ports of a component C to a bus B, $Llinks(B, C) \stackrel{\text{df}}{=} \{l \in L \parallel l \equiv \langle p, B\rangle,\ p \in I(C)\}$.

We do not define the notion of gate, as we focus only on internetwork aspects of the architecture. However, gate is a crucial element of VCN structure concerning the hierarchical aspects [14]. For purpose of this paper, we treat a gate as an injective mapping $gate_G$ which maps a port of the component body to some port of the encompassing interface. Moreover, $gate_G$ is assumed to satisfy $gate_G(\tau) = \tau$.

3.2 Behavioral Model

3.2.1 Leaves

Each leaf is characterized by the set of events which can occur during its computation. The behavioral model of a leaf is determined by the traditional transition system $S = \langle \mathtt{S}, T, \mathtt{s}_0\rangle, \mathtt{s}_0 \in \mathtt{S}$ where the transition relation has the form $T \stackrel{\text{df}}{=} \mathtt{S} \times (\mathcal{P} \cup \{\tau\}) \times \mathtt{S}$. Behavioral model of the leaf S is determined by the initial state $\mathtt{s}_0$ and denoted $\Phi(S) = \mathtt{s}_0$.

3.2.2 Buses

The behavioral model of a bus represents some coordination mechanism. An example of bus behavior of the *COORD* bus from Figure 1 is depicted in Figure 2. The model is represented as a VCN network with four components

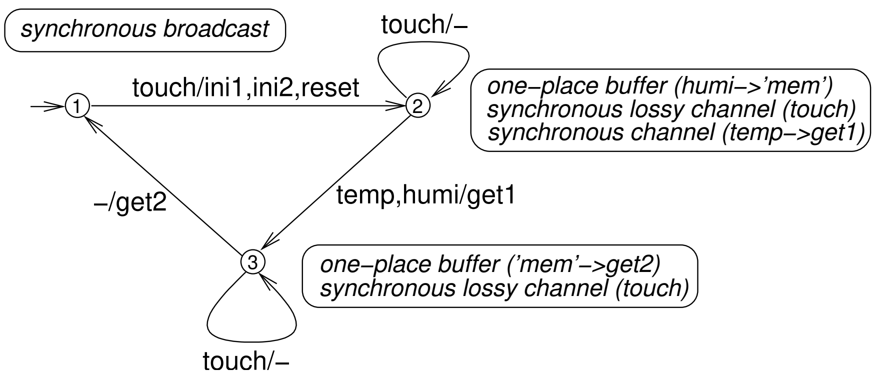

Fig. 2. Behavioral model of a bus (cooperation machine)

Temp_sensor, *Humi_sensor*, which represent weather conditions sensors, the *Display* component representing the LCD panel reporting the current theater information, and the *Switch* component which initiates entire system computation. The coordination model of these components is determined by the bus *COORD*. The meaning of this bus is coordination behavior which combines atomic broadcast, synchronous channel, inhibitor, and a one-place buffer. The overall principle of the coordination model determined by the bus *COORD* in this particular example can be summarized in the following phases:

(i) In the initial phase, pushing of the *touch* trigger causes broadcasting of initiating signal through the $ini1, ini2$, and *reset* ports to the corresponding components. This broadcasting is performed atomically in an indivisible instant of time to ensure immediate and uninterruptible reaction of the system to the initiating signal.

(ii) After initiation, the coordination model is waiting for the information to be signaled on the producers ($Temp\_sensor$ and $Humi\_sensor$) output ports. In this phase, it is also capable of receiving the *touch* signal, to preserve blocking of the initiation switch. However, this signal is lost whenever it is transmitted (initiation cannot be performed in this phase).

When the *temp* and *humi* information appears on the respective ports, the coordination model acts like both a synchronous channel (relaying the *temp* information to the $get1$ port) and a one-place buffer (storing the *humi* information to the internal memory — the state (3)).

(iii) After filling of the buffer, the information stored in the buffer is transmitted to the $get2$ port. It ensures that the $get1$ and $get2$ ports are filled with the appropriate information in a sequence of a given order (i.e., the information of the current weather conditions can appear always in the predefined order on the display panel).

(iv) The coordination model returns to its initial phase.

The coordination behavior described above can be formally captured as a transition system labeled with cooperations. We call such a variant of transition system *cooperation machine*. The cooperation machine is declared formally by the following definition:

Definition 3.10 Let B a bus. *cooperation machine of* B, denoted $cm(B)$, is a tuple $cm(B) \stackrel{\text{df}}{=} \langle Q, T, q_0 \rangle$ where

- Q is a finite set of states,
- $q_0 \in Q$ an initial state,
- $T \subseteq Q \times (2^B \setminus \{\emptyset\}) \times Q$ a transition relation.

Remark 3.11 For a given bus B, the set of all states of the cooperation machine $cm(B)$ is denoted $Q(B)$. Further, the initial state of the cooperation machine $cm(B)$ is denoted $\Phi_B(B)$. The fact that $\langle q, \langle W/R \rangle, q' \rangle \in T$ for some $q, q' \in Q(B)$ and $\langle W/R \rangle \in B$ is denoted $q \stackrel{W/R}{\rightarrow}_B q'$. Additionally, the set of all transitions in which the bus can be engaged in some state $q \in Q(B)$ is denoted $en(q)$, $en(q) \stackrel{\text{df}}{=} \{\langle W/R \rangle \in B \parallel \exists q' \in Q(B).\, q \stackrel{W/R}{\rightarrow}_B q'\}$.

3.2.3 Networks

Definition 3.12 For each component $C = \langle S, I, G \rangle$ and each network or leaf configuration $\mathtt{S}$ define the *component configuration* $\langle \mathtt{S}, I, G \rangle$. Behavioral model of a component C is defined as a mapping $\Phi_C(\langle S, I, G \rangle) \stackrel{\text{df}}{=} \langle \mathtt{S}, I, G \rangle$, $\mathtt{S} = \Phi(S)$.

Definition 3.13 Define the set of *network configurations* as the set

$$\{\langle\langle c_1,..,c_n\rangle,\langle q_1,..,q_m\rangle\rangle \parallel \begin{array}{l}\forall n,m\in\mathcal{N},\ c_1,..,c_n \ldots \text{component configurations,}\\ q_1\in Q(B_1),..,q_m\in Q(B_m)\ \ldots \text{bus states}\ \}\end{array}$$

The behavioral model of the network $N = \langle\langle C_1,..,C_n\rangle,\langle B_1,..,B_m\rangle, L\rangle$ is defined by the mapping $\Phi(N) \overset{\text{df}}{=} \langle\langle \Phi_C(C_1),...,\Phi_C(C_n)\rangle,\langle \Phi_B(B_1),...,\Phi_B(B_m)\rangle\rangle$.

Let $\mathtt{N} = \langle\langle c_1,..,c_n\rangle,\langle q_1,...,q_m\rangle\rangle$ a network configuration. By the notation $\mathtt{N}[c_i := c']$ denote the network configuration which differs from $\mathtt{N}$ only in its ith component configuration, so that the component configuration c_i is replaced with c'_i. Similarly, by the notation $\mathtt{N}[q_j := q']$ denote the network configuration which differs from $\mathtt{N}$ only in its jth bus state, so that the bus state q_j is replaced with q'.

For some $\theta \subseteq \{1,...,n\}$ denote $\mathtt{N}[\bigwedge_{i\in\theta} c_i := c'_i]$ the network configuration which differs from $\mathtt{N}$ in all component positions included in θ, so that each component configuration c_i for some $i\in\theta$ is replaced with a new configuration c'_i. Moreover, the network configuration which differs from $\mathtt{N}$ in all component positions included in θ and additionally in jth state of the bus B_j which has moved to state q' is denoted $\mathtt{N}[\bigwedge_{i\in\theta} c_i := c'_i,\ q_j := q']$.

For an arbitrary behavioral term t of any kind denote $en(t)$ the set of all events which can be evolved from t, $en(t) \overset{\text{df}}{=} \{e \parallel \exists t'.\, t \xrightarrow{e} t'\}$ where '$\rightarrow$' represents the relevant transition relation for the particular class of behavioral terms. Note that also the internal τ-event can be included in $en(t)$.

The first rule defines the operational semantics of the bottom most components. Let $C = \langle S, I, G\rangle$ a component, and S a leaf term. The operational semantics of C is defined by the transition relation '$\rightarrow_C$' over the relevant component configurations which is given by the inference rule:

$$(1)\quad \frac{\mathtt{s} \xrightarrow{e}_s \mathtt{s}'}{\langle \mathtt{s}, I, G\rangle \xrightarrow{gate_G(e)}_C \langle \mathtt{s}', I, G\rangle}\quad \Big[e\in en(\mathtt{s})\Big]$$

Let $C = \langle N, I, G\rangle$ a component, and $N = \langle\langle C_1, C_2,...,C_n\rangle,\langle B_1,...,B_m\rangle, L\rangle$ a network. The operational semantics of C is defined by the transition relation '$\rightarrow_C$' over the relevant component configurations which is given by the following rule:

$$(2)\quad \frac{\mathtt{N} \xrightarrow{e^{\sharp_i}}_N \mathtt{N}'}{\langle \mathtt{N}, I, G\rangle \xrightarrow{gate_G(e^{\sharp_i})}_C \langle \mathtt{N}', I, G\rangle}\quad \begin{bmatrix} i\in\{1,...,n\}\\ e^{\sharp_i}\in en(\mathtt{N})\end{bmatrix}$$

Operational model of the network N is determined by the transition relation '$\rightarrow_N$' over network configurations. The following rule captures the situation when a component performs an action occurring on a free port.

$$(3)\quad \frac{c_i \xrightarrow{e}_C c'}{\mathtt{N} \xrightarrow{e^{\sharp_i}}_N \mathtt{N}[c_i := c']}\quad \begin{bmatrix} \mathtt{N} = \langle\langle c_1,..,c_i,..,c_n\rangle,\vec{q}\rangle\\ e\in en(c_i)\\ e^{\sharp_i}\text{ is a free port}\end{bmatrix}$$

Finally the cooperative behavior is defined. At a specific instant a maximal cooperation is chosen to be performed from all the enabled cooperations in the current state of the particular cooperation machine. Hence the synchronization rule is defined in the following way.

Definition 3.14 Let $\mathtt{N} = \langle\langle c_1, ..., c_n\rangle, \langle q_1, ..., q_m\rangle\rangle$ a network configuration. Further let $\langle W/R\rangle \in$ Coops a cooperation satisfying $\langle W/R\rangle \in en(q_j)$ for some $j \in \{1, ..., m\}$. We say that $\langle W/R\rangle$ is *enabled in configuration* $\mathtt{N}$ and write $enabled(\langle W/R\rangle, \mathtt{N})$ if and only if there exists a set of ports $P \subset \mathcal{P}^\sharp$ satisfying the following conditions:

- If $e^{\sharp_i} \in P$ for some $i \in \{1, ..., n\}$ then $e^{\sharp_i} \in en(c_i)$.
- $e \in P \Leftrightarrow e \in W$ or $e \in R$.

We say that $\langle W/R\rangle$ is *maximal enabled cooperation in* q_j *of* $\mathtt{N}$ and write $maxenabled(\langle W/R\rangle, \mathtt{N}, q_j)$ if and only if $enabled(\langle W/R\rangle, \mathtt{N})$ and $\forall \langle W'/R'\rangle \in en(q_j). (W' \subseteq W \wedge R' \subseteq R) \vee (W' \cap W = \emptyset \wedge R' \cap R = \emptyset)$.

$$(4)\quad \frac{q_j \xrightarrow{W/R}_{B_j} q'_j \quad \forall e^{\sharp_i} \in \mathcal{M}_{B_j}.\, c_i \xrightarrow{e}_C c'_i}{\mathtt{N} \xrightarrow{\tau}_N \mathtt{N}[\bigwedge_{i\in\theta} c_i := c'_i, q_j := q'_j]} \quad \left[\begin{array}{c} \mathtt{N} = \langle\langle c_1, .., c_n\rangle, \langle q_1, .., q_j, .., q_m\rangle\rangle \\ maxenabled(\langle W/R\rangle, \mathtt{N}, q_j) \\ \mathcal{M}_{B_j} \stackrel{\mathrm{df}}{=} W \cup R \\ \theta \stackrel{\mathrm{df}}{=} \{i \parallel e^{\sharp_i} \in \mathcal{M}_{B_j}\} \end{array}\right]$$

4 Network Interoperability Correctness

Our formal solution for checking of interoperability correctness of VCN behavioral model revisits and extends the approach previously presented and proved by Bernardo et.al. in [1]. Inspired by that work, we establish the solution for checking of arbitrary interoperability-critical property which is preserved by weak bisimulation equivalence of VCN terms. As weak bisimulation equivalence preserves validity of all μ-calculus formulae with weak versions of diamond and box operators, the interoperability checking includes an exhaustive set of interesting properties including deadlock freedom. The principle of interoperability checking relies on the idea of searching for a situation in a particular network hierarchy in which validity of a property being checked is violated by interaction of some components and buses otherwise satisfying the considered property as stand-alone entities. With respect to the inductive definition of VCN diagrams the particular interoperability check traverses the hirerachy from leaves up to the network for which checking is performed.

Note that the approach of Bernardo et.al. deals with design architectures composed of uniform components connected by links which can be of one-to-many character. Anyway, those links are static (stateless) connectors. There is no explicit notion of a connector like in Wright or in our approach. However, connectors can be there still modeled explicitly as components which are logically treated differently than common components. What is not possible there is modeling of connector dynamism concerning atomic many-to-many cooperations, as our approach allows by the behavioral model of buses. This is the reason why the results of [1] cannot be directly applied for developing the interoperability checking framework for the behavioral model of VCN. We follow the way of utilizing and extending those results to fit the character of behavioral model of buses.

Remark 4.1 For a component C_i and a bus B_j of some network $\langle\vec{C},\vec{B},L\rangle \in \mathbf{T}_{\mathrm{st}}$ denote $Llinks(C_i, B_j)$ the set of all links between C_i and B_j:

$$Llinks(C_i, B_j) \stackrel{\mathrm{df}}{=} \{l \in links(L, B_j) \,\|\, \exists p \in I_i.\, port(l) = p^{\sharp_i}\}$$

Definition 4.2 Let $N = \langle\langle C_1, ..., C_n\rangle, \langle B_1, ..., B_m\rangle, L\rangle$ a network. Define the *dependency graph of* N, denoted $\mathcal{G}(N)$, as a bipartite graph $\mathcal{G}(N) \stackrel{\mathrm{df}}{=} \langle\{C_1, ..., C_n\} \cup \{B_1, ..., B_m\}, E\rangle$ where E is defined in the following way:

$$E \stackrel{\mathrm{df}}{=} \{\langle C_i, B_j\rangle \,\|\, Llinks(C_i, B_j) \neq \emptyset\}$$

In the following part, we will assume φ a formula of modal μ-calculus expressing some interoperability safety property. An example of such a property can be deadlock freedom expressed by the formula $\nu Z.\langle\langle -\rangle\rangle\mathtt{tt} \wedge [[-]]Z$. A significant feature of this formula is that it does not refer to any particular event and hence its validity is independent of behavioral model alphabet.

In the previous section cooperation machines have been introduced as specific transition systems with cooperations appearing in transition labels. Hence we at first utilize the traditional (weak) bisimulation equivalence in order to establish the notion of so-called cooperation-labeled weak bisimulation.

Remark 4.3 Let $\gamma \in \mathrm{Coops}^*$ a sequence of cooperations. Denote $\hat{\gamma}$ the following sequences of cooperations:

- $\hat{\gamma} \stackrel{\mathrm{df}}{=} \epsilon$, if $\gamma = \langle\emptyset/\emptyset\rangle$;
- $\hat{\gamma} \stackrel{\mathrm{df}}{=} \hat{\gamma}'$, if $\gamma = \langle\emptyset/\emptyset\rangle^* \cdot \gamma' \cdot \langle\emptyset/\emptyset\rangle^*$ where $\gamma' \neq \epsilon$.

Denote $\stackrel{\gamma}{\Rightarrow}_B$ the sequence of succeeding transitions $\stackrel{\gamma}{\Rightarrow}_B \stackrel{\mathrm{df}}{=} (\stackrel{\emptyset/\emptyset}{\longrightarrow}_B)^* \stackrel{\gamma}{\longrightarrow}_B (\stackrel{\emptyset/\emptyset}{\longrightarrow}_B)^*$.

Definition 4.4 Let B_1 and B_2 buses. Further let $q \in Q(B_1), b \in Q(B_2)$ states of the respective cooperation machines. We say q and b are *(weakly) bisimilar* and write $q \approx^{cl} b$ if and only if for each $\gamma \in B_1 \cup B_2$ both of the following holds:

(i) If $q \stackrel{\gamma}{\longrightarrow}_{B_1} q'$ then $\exists b' \in Q(B_2).\, b \stackrel{\hat{\gamma}}{\Rightarrow}_{B_2} b'$ and $q' \approx^{cl} b'$.

(ii) If $b \stackrel{\gamma}{\longrightarrow}_{B_2} \mathtt{b}'$ then $\exists q' \in Q(B_1).\, q \stackrel{\hat{\gamma}}{\Rightarrow}_{B_1} q'$ and $b' \approx^{cl} q'$.

The fundamental idea of interoperability checking is based on the notion of so-called compatibility which is based on pairwise comparison of behavior of components and buses in the network. More particularly, the behavior observed on links between a component and a bus is considered. In order to enable such comparison we define a projection of a bus behavior to a subset of the bus links.

Definition 4.5 Let $N = \langle\langle C_1, ..., C_n\rangle, \langle B_1, ..., B_m\rangle, L\rangle$ a network, and let B_j for some $j \in \{1, ..., m\}$ its bus, $cm(B_j) = \langle Q, T, q_0\rangle$ its cooperation machine, and $L' \subseteq L$ a link relation. Define *L'-projection of B_j*, denoted $\pi(B_j, L')$, as the bus B' with semantics determined by the cooperation machine $cm(B') \stackrel{\mathrm{df}}{=} \langle Q, T', q_0\rangle$ where T' is defined in the following way:

$$\langle q, \langle W'/R'\rangle, q'\rangle \in T' \stackrel{\mathrm{df}}{\Leftrightarrow} \langle q, \langle W/R\rangle, q'\rangle \in T$$

where $W' \stackrel{\mathrm{df}}{=} W \cap ports(links(L', B_j))$ and $R' \stackrel{\mathrm{df}}{=} R \cap ports(links(L', B_j))$.
The bus B' is defined as the set of all cooperations appearing in labels of T'.

Note that the cooperation-labeled transition relation of cooperation machines can be comprehended formally as an extension of the classical transition relation which is used for determining the operational semantics of VCN terms. This classical transition relation can be lifted to the format of cooperation-labeled transition relation. More precisely, we can consider behavioral model of a VCN term in the form of so-called saturated cooperation machine (defined below). This way we achieve uniform framework for behavioral analysis of VCN terms. In such a setting, we represent the internal τ-event as an empty cooperation $\langle\emptyset/\emptyset\rangle$.

Definition 4.6 The *cooperation machine $cm(B)$* is called *saturated* if each transition of $cm(B)$ has the form $\langle q, b, q'\rangle$ satisfying just one of the following possibilities:

- $b \equiv \langle\{w\}/\emptyset\rangle$ for some $w \in \mathcal{W}^\sharp$;
- $b \equiv \langle\emptyset/\{r\}\rangle$ for some $r \in \mathcal{R}^\sharp$;
- $b \equiv \langle\emptyset/\emptyset\rangle$.

To analyze interoperability of buses and particular components in a network we need to look into the network internal behavior. More specifically, cooperations cannot be hidden in such analysis. In particular, we need to observe cooperations occurring on the network link relation or on its specific subset. To capture this kind of observation of network behavior with respect to some link relation L, we define L-observable model of the network. This operational model is based on the cooperation-labeled transition relation. In the following definition the notion of L-observable model is established and consequently used to formalize the notion of compatibility.

Definition 4.7 Let $N = \langle\langle C_1, ..., C_n\rangle, \langle B_1, ..., B_m\rangle, L\rangle \in \mathbf{T}_{\mathrm{st}}$ a network. Define *L-observable model of network N* by a transition relation $\rightarrow_L$ defined over network configurations by the following rules, which are modification of rules $(3, 4)$:

$$(3') \quad \frac{c_i \xrightarrow{e}_C c'}{\mathtt{N} \xrightarrow{\gamma}_L \mathtt{N}[c_i := c']} \quad \left[\begin{array}{c} \mathtt{N} = \langle\langle c_1, .., c_i, .., c_n\rangle, \vec{q}\rangle \\ e^{\sharp_i} \text{ is a free port} \\ \gamma \stackrel{\text{df}}{=} \begin{cases} \langle e^{\sharp_i}/\emptyset\rangle, \text{ if } e \in \mathcal{W}, \\ \langle \emptyset/e^{\sharp_i}\rangle, \text{ if } e \in \mathcal{R}. \end{cases} \end{array}\right]$$

$$(4') \quad \frac{q_j \xrightarrow{W/R}_{B_j} q'_j \quad \forall e^{\sharp_i} \in \mathcal{M}_{B_j}.\, c_i \xrightarrow{e}_C c'_i}{\mathtt{N} \xrightarrow{W/R}_L \mathtt{N}[\bigwedge_{i\in\theta} c_i := c'_i, q_j := q'_j]} \quad \left[\begin{array}{c} \mathtt{N} = \langle\langle c_1, .., c_n\rangle, \langle q_1, .., q_j, .., q_m\rangle\rangle \\ \mathit{maxenabled}(\langle W/R\rangle, \mathtt{N}, q_j) \\ \mathcal{M}_{B_j} \stackrel{\text{df}}{=} W \cup R \\ \theta \stackrel{\text{df}}{=} \{i \parallel e^{\sharp_i} \in \mathcal{M}_{B_j}\} \end{array}\right]$$

The L-*observation of network* N, denoted $\Phi_L(N)$, is determined as in the case of the common behavioral model by the initial network configuration $\Phi(N)$, but the transition relation $\rightarrow_L$ is taken instead of $\rightarrow_N$.

Definition 4.8 Let $N = \langle\langle C_1, ..., C_n\rangle, \langle B_1, ..., B_m\rangle, L\rangle$ a network, and let $C_i = \langle T, I, G\rangle$ and B_j (for $i \in \{1, ..., n\}, j \in \{1, ..., m\}$) adjacent nodes of $\mathcal{G}(N)$. Further let $L' \subseteq L$ defined as a link relation $L' \stackrel{\text{df}}{=} \{l \in \mathit{links}(L, B_j) \| \exists p \in I.\, p^{\sharp_i} = \mathit{port}(l)\}$.

We say C_i is *compatible in* N *with* B_j, and write $C_i \bowtie^N B_j$, if and only if $\Phi_B(B') \approx^{cl} \Phi_{L'}(\langle\langle T, I', G'\rangle, \langle\pi(B_j, L')\rangle, L'\rangle)$ where

- $I' \stackrel{\text{df}}{=} I \cap P$ where $P \stackrel{\text{df}}{=} \{p \in I \parallel \exists l \in L'.\, p^{\sharp_i} = \mathit{port}(l)\}$
- G' is defined by G restricted to ports of I'

An important subpart of an acyclic network topology is a so-called star topology. Intuitively, the star topology is a group of all components connected to the same bus in a particular acyclic network. Significance of this structure relies on the fact that for checking of interoperability it suffices to check compatibility of the central bus with every component included in the particular star topology. This result is included in the general theorem 4.11. In order to establish the notion of the star topology, we introduce formally notion of subnetwork at first.

Definition 4.9 Let $N = \langle\langle C_1, ..., C_n\rangle, \langle B_1, ..., B_m\rangle, L\rangle \in \mathbf{T}_{\text{st}}$ a network. For some $n' \leq n$ and $m' \leq m$ define the *subnetwork* N' *of* N as a network $N' = \langle\langle C'_1, ..., C'_{n'}\rangle, \langle B'_1, ..., B'_{m'}\rangle, L'\rangle \in \mathbf{T}_{\text{st}}$ satisfying:

- $\{C'_1, ..., C'_{n'}\} \subseteq \{C_1, ..., C_n\}$ and $\{B'_1, ..., B'_{m'}\} \subseteq \{B_1, ..., B_m\}$
- $L' \stackrel{\text{df}}{=} \{l \in L \parallel l \in \mathit{Llinks}(C, B) \wedge C \in \{C'_1, ..., C'_{n'}\} \wedge B \in \{B'_1, ..., B'_{m'}\}\}$
- Each $B \in \{B'_1, ..., B'_{m'}\}$ is defined as $\pi(B_j, L')$ for some $j \in \{1, ..., m\}$.

Further define for some $j \in \{1, ..., m\}$ *star topology of* B_j *in* N as a subnetwork N' of N having the form $N' \stackrel{\text{df}}{=} \langle\langle C'_1, ..., C'_{n'}\rangle, \langle B'\rangle, L'\rangle$ where each $C'_i \in \{C'_1, ..., C'_{n'}\}$ satisfies $\mathit{Llinks}(C'_i, B_j) \neq \emptyset$.

In checking of network interoperability, the potential cyclicity of network topology requires a special care. An example of the interoperability correctness violation in the cyclic topology is depicted in Figure 3. Note that if we take only the star topology

$$\langle\langle C_1', C_2'\rangle, \langle\pi(B_1, L)\rangle, L\rangle,\ L \stackrel{\text{df}}{=} \{\langle in^{\sharp_1}, B_1\rangle, \langle out^{\sharp_1}, B_1\rangle, \langle in^{\sharp_2}, B_1\rangle, \langle out^{\sharp_2}, B_1\rangle\}$$

where the components C_1', C_2' are C_1, C_2 projected to links in L then such a subnetwork is deadlock free. The similar situation holds for the star topology of the bus B_2. However, considering the entire cyclic topology a deadlock situation can arise, as is depicted by the darkened states of respective transition systems.

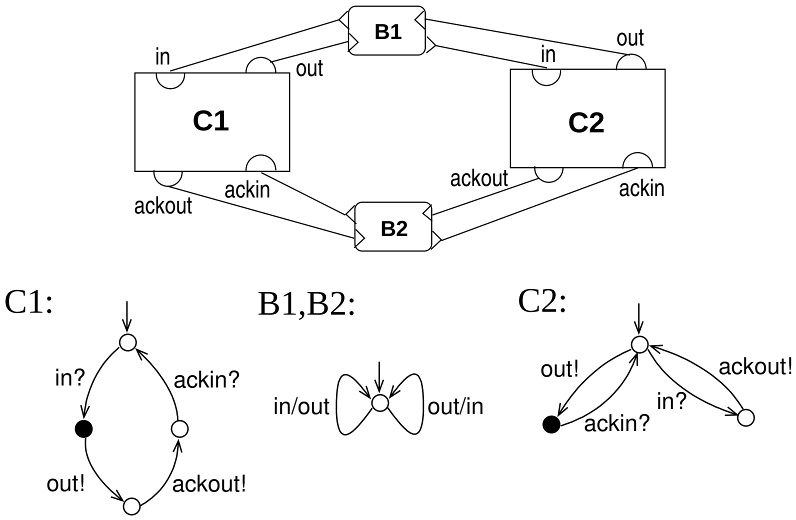

Fig. 3. Violation of network interoperability correctness by deadlock

To capture the above stated problem, we present the main result of this paper declaring a set of conditions which are both necessary and sufficient to ensure an arbitrary network topology to satisfy some property φ. In the following theorem, by maximal cycle we mean each subgraph of a network dependency graph which is not strictly contained in some larger cycle of the graph. Moreover, for each maximal cycle Ω of a network dependency graph we consider the respective subnetwork, so-called Ω-subnetwork, containing all the buses and components of the cycle and the link relation restricted to them. Additionally, we define the border of a cycle to denote the set of components in the cycle which have some free port or a link leading to a bus outside the cycle and buses in the cycle linked to some component outside the cycle.

Definition 4.10 Let $N = \langle\langle C_1, ..., C_n\rangle, \langle B_1, ..., B_m\rangle, L\rangle$ a network and Ω some (maximal) cycle of $\mathcal{G}(N)$. Define *border* of Ω, denoted $\beta(\Omega)$, as the set

$$\begin{aligned}\beta(\Omega) \stackrel{\text{df}}{=} &\{C \in \Omega \parallel \exists p \in I(C), \forall B \in \Omega.\, p \notin ports(Llinks(B, C))\} \cup \\ &\{B \in \Omega \parallel \exists \gamma \in B, p^\sharp \in \gamma, \forall C \in \Omega.\, p \notin ports(Llinks(B, C))\}\end{aligned}$$

Theorem 4.11 *Let $N = \langle\langle C_1, ..., C_n\rangle, \langle B_1, ..., B_m\rangle, L\rangle$ a network with $\mathcal{G}(N)$ which is a connected graph of arbitrary shape, and let φ a property. N satisfies φ if and only if the following conditions hold:*

(i) *Each bus $B_j \in \{B_1, ..., B_m\}$ satisfies φ.*

(ii) *Each component $C_i \in \{C_1, ..., C_n\}$ satisfies φ.*

(iii) *Each bus B which is not included in any cycle of $\mathcal{G}(N)$ satisfies $C \bowtie^N B$ for each C such that $Llinks(C, B) \neq \emptyset$.*

(iv) *Each bus $B \in \beta(\Omega)$ of each maximal cycle Ω of $\mathcal{G}(N)$ satisfies $C \bowtie^N B$ for each C such that $Llinks(C, B) \neq \emptyset$ and C is not included in Ω.*

(v) *For each maximal cycle Ω of $\mathcal{G}(N)$ the Ω-subnetwork of N satisfies φ.*

To realize interoperability checking for arbitrary network topology and that way constructively prove the above stated theorem, we apply the idea of step-by-step reduction of the entire potentially cyclic topology to a smaller acyclic topology which has the behavioral model observationally equivalent to the original topology. The intuition about such an observational behavior preserving reduction of cyclic topologies to acyclic topologies is the following. Each cycle in the entire network is replaced with a star topology which has the behavior model equivalent to the cyclic topology. This star topology is defined in such a way that all the relevant links leading from components and buses of the cycle to the buses and components outside the cycle are remapped to equivalent links leading from components and buses of the star topology. The important property that must be satisfied by such a replacement is mutual compatibility of those components and buses. Moreover, the star topology must itself satisfy compatibility of its components with the bus forming its center. Then by replacing all the cyclic subnetworks with such compatible star topologies the acyclicity of the entire network is achieved and interoperability checking then relies on checking mutual compatibility of buses and components. The formal proof of the theorem is based on the intuition described in this paragraph and is given precisely in the full version of the paper [13].

5 Conclusion and Future Work

In this paper we have presented simplified version of the language VCN for hierarchical specification of component-based concurrent systems. The key concept of the language are buses which represent coordination models used in system architectures. We have utilized the process algebraic approach of [3] and proved its extension which has been introduced due to the specific features of VCN which are not incorporated directly in traditional process algebraic approaches for architectural description.

We are currently implementing a graphical tool which allows VCN diagrams to be simply created and modified. In our future work, we would like to discuss endogenous and exogenous extensions of the VCN models in the style of [1]. We also aim at extending the architectural interoperability checks to VCN bus classes which allow generalized definitions of buses [14].

References

[1] Aldini, A. and M. Bernardo, *On the usability of process algebra: An architectural view*, Theor. Comput. Sci. **335** (2005), pp. 281–329.

[2] Allen, R. and D. Garlan, *A formal basis for architectural connection*, ACM Trans. Softw. Eng. Methodol. **6** (1997), pp. 213–249.

[3] Bernardo, M., P. Ciancarini and L. Donatiello, *Architecting families of software systems with process algebras*, ACM Trans. Softw. Eng. Methodol. **11** (2002), pp. 386–426.

[4] Bouvry, P. and F. Arbab, *Visifold: A Visual Environment for a Coordination Language*, in: *Coordination Languages and Models* (1996).

[5] Ciancarini, P., *Coordination Models and Languages as Software Integrators*, ACM Computing Surveys **28(2)** (1996), p. 300.

[6] Cleaveland, R., X. Du and S. A. Smolka, *GCCS: A Graphical Coordination Language for System Specification*, in: *Proceedings of COORD'00* (2000).

[7] de Alfaro, L. and T. A. Henzinger, *Interface automata*, in: *Proceedings of the Ninth Annual Symposium on Foundations of Software Engineering*, ACM Press, 2001.

[8] Harel, D., *Statecharts: A Visual Formalism for Complex Systems*, Technical report, The Weizmann Institute (1987).

[9] Papadopoulos, G. A. and F. Arbab, *Coordination models and languages*, in: *761*, Centrum voor Wiskunde en Informatica (CWI), 1998 p. 55.

[10] Ray, A. and R. Cleaveland, *Architectural Interaction Diagrams: AIDs for System Modeling*, in: *Proc. of ICSE 2003* (2003).

[11] Rowson, J. A. and A. Sangiovanni-Vincentelli, *Interface-based design*, in: *DAC '97: Proceedings of the 34th annual conference on Design automation* (1997), pp. 178–183.

[12] Šafránek, D., *SGCCS: A Graphical Language for Real-Time Coordination*, in: *Proceedings of FOCLASA'02*, ENTCS **68** (2002).

[13] Šafránek, D., *Architectural Interoperability Checking in Visual Coordination Networks*, Technical Report FIMU-RS-2006, Faculty of Informatics, Masaryk University Brno (2006).

[14] Šafránek, D. and J. Šimša, *VCD: A Visual Formalism for Specification of Heterogeneous Software Architectures*, in: *Theory and Practice of Computer Science: 31st Conference on Current Trends in Theory and Practice of Computer Science*, LNCS **3381** (2005).