# Contracts for Cross-organizational Workflows as Timed Dynamic Condition Response Graphs

Thomas Hildebrandt[a,1], Raghava Rao Mukkamala[a], Tijs Slaats[a,b], Francesco Zanitti[a]

[a] *IT University of Copenhagen, Rued Langgaardsvej 7, 2300 Copenhagen, Denmark*
[b] *Exformatics A/S, 2100 Copenhagen, Denmark*

**Abstract**

We conservatively extend the declarative Dynamic Condition Response (DCR) Graph process model, introduced in the PhD thesis of the second author, to allow for discrete time deadlines. We prove that safety and liveness properties can be verified by mapping finite timed DCR Graphs to finite state transition systems. We exemplify how deadlines can introduce time-locks and deadlocks and violate liveness. We then prove that the general technique for safe distribution of DCR Graphs provided in previous work can be extended to timed DCR Graphs. We exemplify the use of timed DCR Graphs and the distribution technique in praxis on a timed extension of a cross-organizational case management process arising from a previous case study. The example shows how a timed DCR Graph can be used to describe the global contract for a timed workflow process involving several organizations, which can then be distributed as a network of communicating timed DCR Graphs describing the local contract for each organization.

*Key words:* Time, Contracts, Declarative process models, Cross-organizational workflow, Safety, Liveness

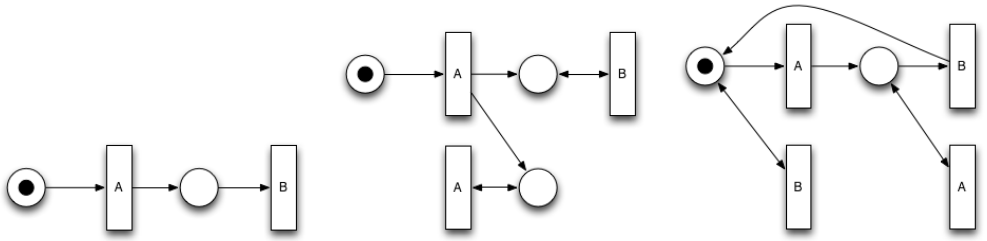
*Email addresses:* hilde@itu.dk (Thomas Hildebrandt), rao@itu.dk (Raghava Rao Mukkamala ), tslaats@itu.dk ( Tijs Slaats), frza@itu.dk (Francesco Zanitti)

[1] Authors listed alphabetically. This research is supported by the Danish Research Agency through the Trustworthy Pervasive Healthcare Services project (grant #2106-07-0019, www.trustcare.eu), an Industrial PhD grant, and the Genie Project (grant # 2106-080046)

## 1. Introduction

The idea of constructing process-aware information systems [1] to support correct execution and analysis of workflows based on explicit models of the processes dates back to the work on office-automation [2, 3, 4, 5] in the late 70ties. The early work was influenced by the Petri Net process model [6], which has also heavily influenced subsequent process modeling standards such as UML activity diagrams [7, 8] and BPMN [9] and the work on formal semantics and analysis of workflow and business processes (e.g. [10, 11, 12, 13]).

The heavy influence of the Petri Net model is quite natural. It is the first process model explicitly representing the concurrent execution of activities, it has a formal semantics supporting analysis and verification, and it has an intuitive graphical notation. To briefly recall, a Petri Net is a directed graph with nodes alternating between *transitions* and *places*, and a *marking* assigning zero or more *tokens* to each place. A transition is *enabled* if all places connected to it by an incoming edge is marked by at least one token. If an enabled transition is *fired*, one token from each place connected by an incoming edge is removed and one token is added to each place connected by an outgoing edge. Firing a transition thus represents the execution of an activity and the tokens represent resources needed to execute activities.




(a) First A, then B (b) A is a condition for B (c) B is a response to A

Figure 1: Condition and response relations as Petri Nets

However, the notion of places and tokens in the Petri Net model is intrinsically linear and imperative. It describes a way to implement dependencies between activities where the ability to redo an activity must be explicitly modeled. This has been recognized, in particular in the incarnation of BPMN, to increase the risk of over-specification and be best suited for processes that follow a strict flow of control [14, 15]. As an example consider the property *activity A is a condition for activity B*, i.e. an execution of B must be preceded by an execution of A in the

past. The Petri Net in Fig. 1(a) is likely to be the first choice of representation, but it really describes the more rigid implementation that activity A can be executed first (and only once) and then activity B can be executed (and only once). The Petri Net in Fig. 1(b) describes the general property, if all markings are allowed as accepting (terminating). The dual property, that *activity B is a response to activity A*, i.e. an execution of A must be followed by an execution of B at some point in the future, is also implemented by the Petri Net in Fig. 1(a). However, in order to allow all possible executions one needs a less restrictive Petri Net as the one in Fig. 1(c) with the initial marking being the only accepting marking.

As an alternative, declarative models such as temporal logics like LTL [16] or CTL [17] can be employed in order to maintain more flexibility of the execution and provide a specification of the dependencies between activities which abstracts from any particular implementation. This can in particular be used as a *contract* for a subsequent implementation or compliance verification [18]. Temporal logics are however in general considered to be difficult to understand by end-users and typically proposed to be replaced by *patterns* of properties, such as the condition and response patterns considered in Fig. 1(b) and Fig. 1(c) [19, 20, 18]. However, before the contract can be checked for a workflow process it is usually required that the contract is rewritten either to a particular (normal) form [21] or translated to an (imperative) automaton [20, 22, 23], which is difficult to understand and relate to the original contract.

This motivates finding a declarative process model which both can be understood by end-users of workflow management systems, e.g. by capturing patterns in a direct way, and supports verification and monitoring without rewriting the process. Dynamic Condition Response (DCR) Graphs [24, 25, 26, 27] is a candidate for such a declarative process language developed in the PhD project [28] of the second author as part of the Trustworthy Pervasive Healthcare Services (TrustCare) [29] research project.

From a practical perspective, DCR Graph model generalizes and formalizes the core primitives of the declarative Process Matrix model [30], developed, patented and used successfully for more than a decade by Resultmaker [31], the industrial partner of the project. The goal of the formalization was to provide the basis for formal verification and safe distributed execution of flexible, cross-organizational workflow processes as found in the healthcare sector.

From a technical perspective, a DCR Graph is a directed graph described by a 9-tuple $(\mathsf{E}, \mathsf{M}, \rightarrow\bullet, \bullet\rightarrow, \rightarrow\diamond, \rightarrow+, \rightarrow\%, \mathsf{L}, l)$. The nodes of the graph are given by the set $\mathsf{E}$ of *events*. The event represents that some activity, as indicated by the labeling function $l : \mathsf{E} \rightarrow \mathcal{P}(\mathsf{L})$, happens in the workflow. The edges of the


(a) DCR Graph condition relation (b) DCR Graph response relation

Figure 2: Condition and response relations in DCR Graphs

graph are given by five relations: The *condition* ($\rightarrow\bullet$), *response* ($\bullet\rightarrow$), *milestone* ($\rightarrow\diamond$), *include*($\rightarrow+$), and *exclude* ($\rightarrow\%$) relation respectively. The condition and response relations thus directly captures the condition and response patterns as shown by the DCR Graphs in Fig.1 . Before explaining the milestone relation, it is helpful to take a look the 2nd element of the DCR Graph tuple, which is the *marking* $\mathsf{M}$. The marking consists of a triple of sets of events $(\mathsf{Ex}, \mathsf{Re}, \mathsf{In})$ representing the state of the process. The set $\mathsf{Ex} \subseteq \mathsf{E}$ of *executed* events records which events have been executed in the past. The set $\mathsf{Re} \subseteq \mathsf{E}$ of *response* events records which events are required to be executed (i.e. responses) in the future in order for the entire execution to be accepting. The milestone relation means that an event can not execute while another event is pending response, it is particularly useful for descibing that a process can not continue on to a new phase while a previous phase hasn't been completed. Finally, the set $\mathsf{In} \subseteq \mathsf{E}$ of *included* events records the events which are currently included. If $e \rightarrow\% \ e'$ then $e'$ is removed from the set $\mathsf{In}$ when $e$ is executed, and if $e \rightarrow+ \ e'$ then $e'$ is added to the set $\mathsf{In}$ when $e$ is executed. This notion of dynamic inclusion and exclusion of events is the key new ingredient of DCR Graphs compared to other declarative approaches. Only currently included events in $\mathsf{Re}$ are required in order for the execution to be accepting, and only included condition events $e'$ of an event $e$ need to be in the set $\mathsf{Ex}$ in order for the event $e$ to be enabled.

The markings can also be seen as the state of a (kind of Kripke) logical program given by the DCR Graph [32]. Then $\mathsf{Ex}$ is the set of basic facts that has been proven, $\mathsf{Re}$ is what has to be proven (possibly again) or excluded from the world to complete the proof, and $\mathsf{In}$ is the set of facts that are relevant in the current world. The markings also form the states of a Büchi-automata representation of DCR Graphs [25], which allows verification of safety and liveness properties using the SPIN model checking tool [28, 33].

The main new contribution of the present paper is to conservatively extend the DCR Graph model to allow for (discrete) time deadlines, while preserving the

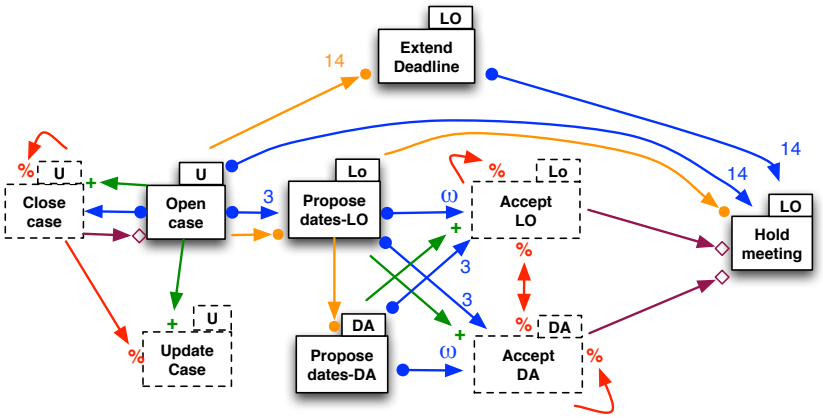



Figure 3: A timed DCR graph describing the global contract for a cross-organizational case management system.

simple operational semantics and that safety and liveness properties of the models can be formally verified by mapping the graphs to finite state Büchi-automata. Fig. 3 shows an example of a timed DCR Graph which we will use as example in the paper. The graph extends an untimed DCR Graph from a case study described in [34] which captures the *global* requirements of a distributed, cross-organizational case management system being developed by Exformatics A/S. The system spans three different types of organizations. The first is the *unions* for employees in Denmark, which creates, closes and updates cases on behalf of their members (represented by the events labelled with the U and respectively Open case, Close case and Update Case. The two other organizations are respectively the umbrella organization for the unions (LandsOrganisationen, LO) and the umbrella organization for the employers (Dansk Arbejdsgiverforening, DA). When a case is created by a union, LO and DA must agree on a date for a meeting and hold the meeting within 14 days from when the case was created. The deadline is expressed by the number 14 attached to the response relation from Open case to Hold meeting. If the meeting is not ready to be held after 14 days however, the deadline can be extended by LO by executing the event Extend Deadline. However, this event only becomes enabled 14 days after the case has been created, which is represented by the number 14 attached to the condition relation from Open case and Extend Deadline.

As described in [27], the DCR Graphs model admits a very general technique for distributing a graph describing a global contract for e.g. a cross-organizational process, as a network of synchronously communicating graphs describing the con-

tracts for each local organization, and which combined describe the same process as the global contract. Another new contribution of the present paper is to extend this technique to timed DCR Graphs and provide a detailed proof of the correctness of the distribution technique. It is the first step of the further development of the DCR Graph model and its use for specification and safe execution of distributed, cross-organizational workflow processes to be carried out in the recently initiated industrial PhD project of the third author.

The rest of the paper is structured as follows. In Sec. 2 below we briefly survey related work. In Sec. 3 we first recall the formal definition of DCR Graphs and their semantics, and then proceed to define the extension to timed DCR Graphs. We then in Sec. 4 define and exemplify timed notions of safety and liveness for timed DCR Graphs. In Sec. 5 we exemplify and provide the technique for distributing a global timed DCR Graph, extending the technique for un-timed DCR Graphs given in [27]. Finally we conclude and comment on future work in Sec. 6.

## 2. Related Work

The DCR Graph model and graphical representation is similar to the Declare workflow language [35, 20]. Indeed the graphical representation of events and condition and response relations are the same in DCR Graphs and Declare. However, the two approaches differ in a crucial way. While the Declare model includes a fairly rich set of patterns as relations between events based on the patterns identified in [19], the DCR Graph model in addition to the condition and response relations only allows the milestone relation and the dynamic inclusion and exclusion relations. Still, the DCR Graphs model is more expressive: It allows to express all $\omega$-regular languages [28], while Declare only allow expressing properties covered by finitary LTL. Moreover, Declare models are translated to either LTL or a new notion of colored automata [36] before they are verified or executed. The DCR Graph model directly supports execution of the process model based on the simple notion of markings of the graph.

There are many researchers [10, 11, 37, 12, 13, 38, 39] who have explicitly focussed on the problem of verifying the correctness of inter-organizational workflows based on variants of Petri nets. In [10], message sequence charts are used to model the interaction between local workflows modeled as Petri nets. In [11] Kindler et. al. follows a similar approach, using a set of scenarios given as sequence diagrams to specify the interactions. Criteria of local soundness guarantee the global soundness of an inter-organizational workflow. In [37], so-called *Query Nets* based on predicate/transition Petri nets are used to guarantee global termina-

tion. Besides the use of Petri nets, a key difference between the present paper and the work in [10, 11, 37] is that while we take a top down approach and *synthesize* models of the local workflows from a model of the global workflow, the latter work take a bottom up approach, assuming that the models of the local workflows are given.

A top down approach is taken in [13, 38], where a shared public view of an inter-organizational workflow given as a workflow net is partioned among the participating entities. A notion of projection inheritance is used to generate a private view that is a subclass to the relevant public view and guarantee deadlock and live-lock freedom. A more liberal and weaker notion than projection inheritance is used in [12] to guarantee weak termination in multiparty contracts based on open nets.

Modeling global behavior as a set of conversations among participating services has been studied by many researchers [40, 41, 42, 43, 44, 45] in the area business processes. An approach based on guarded automata studied in [40], for the realizability analysis of conversation protocols, whereas the authors in [41] used colored petri nets to capture the complex conversations. Similarly, but using process calculus to model service contracts, Bravetti-Zavattaro proposed conformance notion for service composition in [44] and further enhanced their correctness criteria in [45] by the notion of strong service compliance. The synthesis of local components from a global model has also been researched for process calculus formalizations of the imperative choreography language WS-CDL in the work on structured communication-centred programming for web services by Carbone, Honda and Yoshida [46]. To put it briefly, the work formalizes the core of WS-CDL as the global process calculus and defines a formal theory of end-point projections projecting the global process calculus to abstract descriptions of the behavior of each of the local ”end-points” given as pi-calculus processes typed with session types.

Also researchers [47, 48, 49, 50] in the web services community have been working on web service composition and decentralized process execution using BPEL [51] and other related technologies to model web services. A technique to partition a composite web service using program analysis is studied in [48] and [49] explore decomposition of a business process modeled in BPEL. Using a formal approach based on I/O automata representing the services, the authors in [50] study the problem of synthesizing a decentralized choreography strategy, that will have optimal overhead of service composition in terms of costs associated with each interaction. In [52, 53, 54] foundational work is presented on synthesizing distributed transition systems from global specification for the mod-

els of synchronous product and asynchronous automata[55]. In [54] structural and behavioral characterizations of the synthesis problem for synchronous and loosely cooperating communication systems are given, based on three different notions of equivalence: state space, language and bisimulation equivalence. Further Castellani et. al. [52] characterizes when an an arbitrary transition system is isomorphic to its product transition systems with a specified distribution of actions and they have shown that for finite state specifications, a finite state distributed implementation can be synthesized. Complexity results for distributed synthesis problems for the three notions of equivalences were studied in [53].

A methodology for deriving process descriptions from a business contract formalized in a formal contract language is studied in [56], while in [57] is proposed an approach to extract a distributed process model from collaborative business process. In [58, 47] is proposed a technique for flexible decentralization of a process specification with necessary synchronization between the processing entities using dependency tables, whereas the authors in [59] present a framework for optimizing the physical distribution of workflow schemas based on families of communicating flow charts.

Common to all the approaches discussed above are that they are confined to imperative process models such as Petri nets, workflow/open nets and automata. To the best of our knowledge, there exists very few works [60, 61] that have studied the synthesis problem in declarative modeling languages and none where both the global and local processes are given declaratively. Fahland [60] studies top down synthesis of Petri Net workflows from a limited subset of the declarative Declare/DecSerFlow [20] model, while Montali [61] studies the bottom-up composition of Declare [35] models with respect to conformance with a given choreography.

None of the work above study synthesis or conformance of cross-organizational workflow with time constraints. [62] provides a good overview of various temporal logics that have been used for specification of real-time systems. Similar to the extension to DCR Graph proposed in the present paper, Time Petri nets [63] extend regular Petri nets by allowing transitions to be labelled with time bounds $0 \leq a \leq b \wedge a \neq \infty$, such that a transition can only fire after a delay of $a$ time steps after it has been enabled last and must either fire or be disabled before $b$ time has passed. Time Petri nets have been studied as a formalism to model and verify time dependent concurrent systems in [64]. In [65, 66] the properties of reachability, boundedness and liveness for time Petri nets are studied, while [67] defines the semantics of time Petri nets in terms of timed automata. Time Workflow nets, a subclass of time Petri nets, have been introduced in [68] as a method for modeling

time management in workflow processes. Timed Petri nets [69] are a variation of Petri nets where a (rational) time duration is assigned to every transition and transitions are required to fire as soon as they become enabled. Timed Petri nets are mainly used for performance evaluation [70]. Timed-arc Petri nets [71, 72] are another variation of Petri nets where tokens are annotated with an age and arcs from places to transitions are labelled with a time interval. The tokens that a transition can consume are then limited to those whose age falls within these time intervals. For Timed-arc Petri nets there is no notion of urgency. Transitions are neither required to fire when adequate tokens are available nor when those tokens are about to expire.

## 3. Timed Dynamic Condition Response Graphs

We employ the following notations in the rest of the paper.

**Notation:** We write $\infty$ for the set of finite ordinals and the least infinite ordinal $\omega$. For an ordinal $k \in \infty$ we write $[k]$ for the (possibly infinite) set $\{i \mid 0 \leq i < k\}$. For a set $E$ we write $\mathcal{P}(E)$ for the power set of $E$ (i.e. set of all subsets of $E$). For a binary relation $\rightarrow \subseteq E \times E$ and a subset $\xi \subseteq E$ of $E$ we write $\rightarrow \xi$ and $\xi \rightarrow$ for the set $\{e \in E \mid (\exists e' \in \xi \mid e \rightarrow e')\}$ and the set $\{e \in E \mid (\exists e' \in \xi \mid e' \rightarrow e)\}$ respectively, and abuse notation writing $\rightarrow e$ and $e \rightarrow$ for $\rightarrow \{e\}$ and $\rightarrow \{e\}$ respectively when $e \in E$.

We first recall in Def. 3.1 the formal definition of DCR Graphs and their semantics.

**Definition 3.1.** *A Dynamic Condition Response Graph (DCR Graph) G is a tuple* $(\mathsf{E}, \mathsf{M}, \rightarrow\bullet, \bullet\rightarrow, \rightarrow\diamond, \rightarrow+, \rightarrow\%, \mathsf{L}, l)$*, where*

*(i)* $\mathsf{E}$ *is a set of* events *(or activities),*

*(ii)* $\mathsf{M} = (\mathsf{Ex}, \mathsf{Re}, \mathsf{In}) \in \mathcal{P}(\mathsf{E}) \times \mathcal{P}(\mathsf{E}) \times \mathcal{P}(\mathsf{E})$ *is the* marking

*(iii)* $\rightarrow\bullet, \bullet\rightarrow, \rightarrow\diamond, \rightarrow+, \rightarrow\% \subseteq \mathsf{E} \times \mathsf{E}$ *is the* condition*,* response*,* milestone*,* include *and* exclude *relation respectively.*

*(iv)* $\mathsf{L}$ *is the set of* labels *and* $l : \mathsf{E} \rightarrow \mathcal{P}(\mathsf{L})$ *is a labeling function mapping events to sets of labels.*

*We define that an event* $e \in \mathsf{E}$ *is* enabled*, written* $\mathsf{M} \vdash_G e$*, if* $e \in \mathsf{In} \wedge (\mathsf{In} \cap \rightarrow\bullet e) \subseteq \mathsf{Ex}$ *and* $(\mathsf{In} \cap \rightarrow\diamond e) \subseteq \mathsf{E} \backslash \mathsf{Re}$.

*Finally, we define the result of executing an event* $e$ *as* $(\mathsf{Ex}, \mathsf{Re}, \mathsf{In}) \oplus_G e =_{def} \big(\mathsf{Ex} \cup \{e\}, (\mathsf{Re} \setminus \{e\}) \cup e\bullet\rightarrow, (\mathsf{In} \setminus e\rightarrow\%) \cup e\rightarrow+\big)$.

Intuitively, the marking of a DCR Graph records the state. An event is then $e$ is enabled in a marking, if it is included and every event which is a condition for $e$ is either excluded or executed or, and every event which is a milestone for $e$ is either excluded or not required as a response, i.e. it is in a completed state. Below we give a simple example of a DCR Graph and execution of an event based on the graph in Fig. 3.

**Example 3.1.** *As an example consider the underlying (untimed) DCR Graph $G$ of the timed DCR Graph $G_t$ in Fig. 3: the initial marking of $G$ is* $\mathsf{M} = (\emptyset, \emptyset, \mathsf{E} \setminus \{\textsf{Accept LO}, \textsf{Accept DA}, \textsf{Close Case}, \textsf{Update Case}\})$. *The only enabled event is* Open case*, all other events are either blocked through the condition relation, or not in the set of included events. After executing* Open case*, the new marking* $\mathsf{M}'$ *of $G$ becomes:* $\mathsf{M}' = \mathsf{M} \oplus_G \textsf{Open case} = (\{\textsf{Open case}\}, \{\textsf{Propose dates-LO}, \textsf{Hold meeting}, \textsf{Close Case}\}, \mathsf{E} \setminus \{\textsf{Accept LO}, \textsf{Accept DA}\})$. Open case *is added to the set of executed events,* Close Case*,* Propose dates-LO *and* Hold meeting *are responses to* Open case *and therefore added to the set of pending responses and finally* Close case *and* Update Case *are added to the set of included events through the include relation from* Open case*. The enabled events for* $\mathsf{M}'$ *are* Extend Deadline*,* Propose dates-LO*,* Update Case *and* Close case*. Note that the event* Open case *is not enabled because there is a milestone relation from* Close case*, and* Close case *is in the response set. In other words, the case has to be closed by the union before it can be opened again.*

We now proceed to define the conservative extension of DCR Graphs to allow (discrete) time constraints. The aim is to allow modeling interesting timed systems while preserving that finite DCR Graphs have tractable finite state semantics and the technique for distribution of DCR Graphs still applies. The basic idea is to annotate response relations with discrete (possibly infinite) time constraint $k \in \infty$ and condition relations with a finite time constraint $j \in \omega$. A response relation $e \bullet\!\!\xrightarrow{k} e'$ then specifies that the response event $e'$ must happen within $k$ time steps after the last time $e$ happened. The condition relation $e \xrightarrow{j}\!\!\bullet\ e'$ specifies that the last time the condition event $e$ happened must be at least $j$ time steps before $e'$ can happen. That is, the time constraint $e \bullet\!\!\xrightarrow{\omega} e'$ means that the event $e'$ should happen eventually after $e$ happens, and the time constraint $e \xrightarrow{0}\!\!\bullet\ e'$ means that event $e$ should have happened before $e'$ can happen, corresponding to respectively the response and condition constraints in un-timed DCR Graphs. For this reason we often write $e \bullet\!\!\rightarrow e'$ for $e \bullet\!\!\xrightarrow{\omega} e'$ and $e \rightarrow\!\!\bullet\ e'$ for $e \xrightarrow{0}\!\!\bullet\ e'$.

To be able to evaluate the timed constraints we extend markings to include two functions, a function $t_{ex} : \mathsf{Ex} \to \omega$ recording the time since the event was last executed, and $t_{re} : \mathsf{Re} \to \infty$ recording the time deadline of the pending responses.

If there exists a maximal condition deadline, which is always the case if the timed DCR Graph is finite, we say that it is bounded. For bounded timed DCR Graphs we only need to record the time since last execution if it is below the maximal time constraint and otherwise record it as the maximal condition deadline.

**Definition 3.2.** *A* Timed *Dynamic Condition Response Graph (Timed DCR Graph) G is a tuple* $(\mathsf{E}, \mathsf{M_t}, \to\bullet, t_c, \bullet\to, t_r, \to\diamond, \to+, \to\%_0, \mathsf{L}, l)$*, where*

*(i)* $\mathsf{M_t} = (\mathsf{M}, t_{ex}, t_{re}) \in \mathcal{M}(G)$ *is the* timed *marking, for* $\mathcal{M}(G) =_{def} (\mathcal{P}(\mathsf{E}) \times \mathcal{P}(\mathsf{E}) \times \mathcal{P}(\mathsf{E})) \times \mathsf{Ex} \to \omega \times \mathsf{Re} \to \infty$ *if* $\mathsf{M} = (\mathsf{Ex}, \mathsf{Re}, \mathsf{In})$*,*

*(ii)* $(\mathsf{E}, \mathsf{M}, \to\bullet, \bullet\to, \to\diamond, \to+, \to\%_0, \mathsf{L}, l)$ *is a DCR Graph, referred to as the underlying DCR Graph,*

*(iii)* $t_{ex} : \mathsf{Ex} \to \omega$ *is the time since the event happened,*

*(iv)* $t_{re} : \mathsf{Re} \to \infty$ *is the maximal time deadline of the required response,*

*(v)* $t_c : (\to\bullet) \to \omega$ *is the (minimal) required time delay since the condition event happened,*

*(vi)* $t_r : (\bullet\to) \to \infty$ *is the maximal time deadline of the response event.*

*We write* $e \overset{k}{\bullet\to} e'$ *for* $e \bullet\to e'$ *and* $t_r(e, e') = k$ *and similarly write* $e \overset{k}{\to\bullet} e'$ *for* $e \to\bullet e'$ *and* $t_c(e, e') = k$*. We define the maximal condition delay as* $maxc_G =_{def} \mathsf{sup}\{k \mid \exists e, e' \in \mathsf{E}. e \overset{k}{\to\bullet} e'\}$ *and the minimal included response deadline by* $minr_G = \mathsf{min}\{t_{re}(e) \mid \exists e \in \mathsf{Re} \cap \mathsf{In}\}$.

**Notation:** We introduce the following shorthand notation for annotating the events in the executed and response sets with the their timestamp according to respectively $t_{ex}$ and $t_{re}$: For a marking $\mathsf{M_t} = ((\{e_1, e_2\}, \{e_2, e_3\}, \mathsf{In}), t_{ex}, t_{re})$ we write $(\{e_1 : t_{ex}(e_1), e_2 : t_{ex}(e_2)\}, \{e_2 : t_{re}(e_2), e_3 : t_{re}(e_3)\}, \mathsf{In})$.

In Def. 3.3 below we formalize when events are enabled and the result of executing an event in a timed DCR Graph. An event $e$ is enabled in a timed DCR Graph if it is enabled in the underlying un-timed DCR Graph and for all condition events $e' \overset{k}{\to\bullet} e$ the time $t_{ex}(e')$ since the last execution of $e'$ is greater or equal than

$k$. If an event $e$ is executed the sets $(\mathsf{Ex}, \mathsf{Re}, \mathsf{In})$ are updated as for the un-timed DCR Graphs, and in addition, the execution time for $e$ is set to $0$ and all response deadlines for events $e'$ where $e \bullet\!\!\xrightarrow{k} e'$ are set to $k$.

Note that the execution of an event does not advance time. Instead we introduce the notion of *time advance steps* $n \in \omega$, which increases the execution time of events in $\mathsf{Ex}$ and decreases the response deadlines for events in $\mathsf{Re}$. A time advance step $n$ is only enabled if $n \leq minr_G$, that is, all currently *included* response deadlines are greater than or equal to $n$. This means that time can not pass a response deadline of an included event, but it may pass a response deadline of an excluded event. In the latter case, the deadline becomes zero.

**Definition 3.3.** *For a timed Dynamic Condition Response Graph* $G = (\mathsf{E}, \mathsf{M_t}, \rightarrow\bullet, t_c, \bullet\rightarrow, t_r, \rightarrow\diamond, \rightarrow+, \rightarrow\%, \mathsf{L}, l)$*, and timed marking* $\mathsf{M_t} = \big((\mathsf{Ex}, \mathsf{Re}, \mathsf{In}), t_{ex}, t_{re}\big)$ *we define that an event* $e \in \mathsf{E}$ *is* enabled*, written* $\mathsf{M_t} \vdash_G e$*, if* $\mathsf{M} \vdash_{G'} e$*, where* $G'$ *is the underlying un-timed DCR Graph of* $G$ *and* $\forall e' \in \mathsf{In}. e' \xrightarrow{k}\bullet\ e \implies k \leq t_{ex}(e')$*. Moreover, for* $n \in \omega$ *we define that the* time advance step $n$ *is enabled, written* $\mathsf{M_t} \vdash_G n$ *if* $minr_G \geq n$*.*

*We define the result of executing an event* $e$ *as* $\mathsf{M_t} \oplus_G e =_{def} \big((\mathsf{Ex}, \mathsf{Re}, \mathsf{In}) \oplus_G e, t'_{ex}, t'_{re}\big)$*, where*

$$\textit{(i)}\ \ t'_{ex}(e') = \begin{cases} 0 & \textit{if } e' = e \\ t_{ex}(e') & \textit{otherwise} \end{cases}$$

$$\textit{(ii)}\ \ t'_{re}(e') = \begin{cases} k & \textit{if } e \bullet\!\!\xrightarrow{k} e' \\ t_{re}(e') & \textit{otherwise} \end{cases}$$

*We define the result of advancing time with* $n$ *by* $\mathsf{M_t} \oplus_G n =_{def} \big((\mathsf{Ex}, \mathsf{Re}, \mathsf{In}), t_{ex} \oplus n, t_{re} \ominus n\big)$*, where* $t_{ex} \oplus n(e) =_{def} \mathsf{min}\{t_{ex}(e) + n, maxc_G\}$ *and* $t_{re} \ominus n(e) =_{def} \mathsf{max}\{t_{re}(e) - n, 0\}$.

**Example 3.2.** *As an example, consider again the Timed DCR Graph* $G_t$ *from Fig. 3: the initial marking of* $G_t$ *is* $\mathsf{M_t} = (\emptyset, \emptyset, \mathsf{E} \setminus \{$*Accept LO*, *Accept DA*, *Close Case*, *Update Case*$\})$*, note that because of our shorthand notation and the empty* $\mathsf{Ex}$ *and* $\mathsf{Re}$ *sets the marking looks the same as the marking* $\mathsf{M}$ *of the underlying DCR Graph* $G$*. The only enabled event for the initial marking* $\mathsf{M_t}$ *is* *Open case*. *Executing* *Open case* *results in the marking* $\mathsf{M_t}' = \mathsf{M_t} \oplus_G$ *Open case* $= (\{$*Open case* $: 0\}, \{$*Propose dates-LO* $: 3,$ *Hold meeting* $:$

$14$, *Close Case* : $\omega\}, \mathsf{E} \setminus \{$*Accept LO*, *Accept DA*$\})$. *Except for the timed part* $\mathsf{M_t}$*' is the same as* $\mathsf{M}$*' that resulted from executing* *Open case* *on the underlying DCR Graph* $G$*. In the timed marking we also record the time that has passed since* *Open case* *was executed and the deadlines for which the responses on* *Propose dates-LO*, *Hold meeting* *and* *Close case* *need to be satisfied. Note that unlike the untimed example* *Extend Deadline* *is not enabled because of the time constraint on the condition.*

*The timed response on* *Propose dates-LO* *limits the time to advance from the marking* $\mathsf{M_t}$*' to a maximum of 3 steps. Doing a time advance step of size 3 results in the new marking* $\mathsf{M_t}'' = \mathsf{M_t}' \oplus_G 3 = (\{$*Open case* : $3\}, \{$*Propose dates-LO* : $0$, *Hold meeting* : $11$, *Close Case* : $\omega\}, \mathsf{E} \setminus \{$*Accept LO*, *Accept DA*$\})$. *We can now no longer advance time before* *Propose dates-LO* *has been executed or excluded.*

It follows directly from the definition above that if an event is enabled in a timed DCR Graph then it is also enabled in the underlying (un-timed) DCR Graph. Moreover, the result on the sets $(\mathsf{Ex}, \mathsf{Re}, \mathsf{In})$ in the marking when executing an event in a timed DCR Graph is the same as when it is executed in the underlying DCR Graph and time advance steps do not change the marking of the underlying DCR Graph. Conversely, a DCR Graph $G$ can be regarded as a timed DCR Graph $G_t$ having $G$ as the underlying DCR Graph and all condition delays (and execution times in the marking) $0$ and all response deadlines (in the graph and the marking) $\omega$. It then holds that, if an event is enabled in the DCR Graph $G$, then it is also enabled in the corresponding timed DCR Graph $G_t$. Moreover, the execution times and response deadlines in the marking will always be $0$ and $\omega$ respectively.

We define in Def. 3.4 timed (must) executions and the corresponding labelled transition system for timed DCR Graphs as for un-timed DCR Graphs, except that executions can now also contain time advance steps. Moreover, for an execution to be accepting it is required that it is accepting in the underlying DCR Graph and moreover contains infinitely many time advance steps. Thus, an accepting execution may contain only finitely many events, but then it will after the last event contain an infinite sequence of time advance steps. It follows directly that (accepting) executions in a timed DCR Graph correspond to (accepting) executions in the underlying DCR Graph (where the time advance steps are removed). Moreover, an (accepting) execution in a DCR Graph $G$ will have infinitely many corresponding (accepting) executions in the corresponding timed DCR Graph $G_t$ obtained by interleaving the untimed execution with any infinite sequence of time advance steps.

**Definition 3.4.** *For a timed Dynamic Condition Response Graph* $G = (\mathsf{E}, \mathsf{M_t}, \rightarrow\bullet, t_c, \bullet\rightarrow, t_r, \rightarrow\diamond, \rightarrow+, \rightarrow\%, \mathsf{L}, l)$ *we define a* timed execution $\sigma_{\mathsf{M}}$ *of* $G$ *of length* $k \in \infty$ *from* $\mathsf{M}$ *to be a (finite or infinite) sequence of tuples* $\sigma : [k] \rightarrow \mathcal{M}(G) \times (\mathsf{E} \times \mathsf{L} \cup \omega) \times \mathcal{M}(G)$ *such that if for* $i \in [k]$,

- $\sigma(i) = (\mathsf{M}_i, e_i, a_i, \mathsf{M}'_i) \wedge a_i = l(e_i) \wedge \mathsf{M}_i \vdash_G e_i \wedge \mathsf{M}'_i = \mathsf{M}_i \oplus_G e_i$ *or*
- $\sigma(i) = (\mathsf{M}_i, n, \mathsf{M}'_i) \wedge \mathsf{M}_i \vdash_G n \wedge \mathsf{M}'_i = \mathsf{M}_i \oplus_G n$

*and* $\mathsf{M} = \mathsf{M}_0$ *and* $\forall i \in [k-1].\mathsf{M}'_i = \mathsf{M}_{i+1}$.

*We say the execution* $\sigma$ *is a* must *execution if* $\forall i \in [k].\sigma(i) = (\mathsf{M}_i, e_i, a_i, \mathsf{M}'_i) \implies e_i \in \mathsf{In}_i \cap \mathsf{Re}_i$ *and* accepting *if*

*(i)* $\forall i \in [k].\big(\forall e \in \mathsf{In}_i \cap \mathsf{Re}_i.\exists j \geq i.e_j = e \vee e \notin \mathsf{In}'_j)\big)$ *and*

*(ii)* $\forall i \in \omega.\exists k \in \omega.\exists j > i.\sigma(i) = (\mathsf{M}_i, k, \mathsf{M}'_i)$

*where* $\mathsf{M}_i = ((\mathsf{Ex}_i, \mathsf{In}_i, \mathsf{Re}_i), t_{exi}, t_{rei})$ *and* $\mathsf{M}'_j = ((\mathsf{Ex}'_j, \mathsf{In}'_j, \mathsf{Re}'_j), t'_{exj}, t'_{rej})$. *Let* $\mathsf{exe}_{\mathsf{M_t}}(G)$, $\mathsf{mexe}_{\mathsf{M_t}}(G)$, $\mathsf{acc}_{\mathsf{M_t}}(G)$*and* $\mathsf{macc}_{\mathsf{M_t}}(G)$ *denote respectively the set of all executions, all must executions, all accepting executions, and all accepting must executions of* $G$ *starting in marking* $\mathsf{M_t}$.

*We say that a marking* $\mathsf{M}'$ *is reachable in* $G$ *(from the marking* $\mathsf{M}$*) if there exists a finite execution ending in* $\mathsf{M}'$ *and let* $\mathcal{M}_{\mathsf{M}\rightarrow^*}(G)$ *denote the set of all reachable markings from* $\mathsf{M}$.

*We define the corresponding labeled transition system for* $G$ *as* $\mathsf{TS}(G) = (\mathcal{M}(G), \mathsf{M_t}, \mathcal{EL}(G), \rightarrow)$ *where* $\mathcal{EL}(G) = \mathsf{E} \times \mathsf{L} \cup \omega$ *is the set of labels of the transition system,* $\mathsf{M_t}$ *is the initial marking, and* $\rightarrow \subseteq \mathcal{M}(G) \times \mathcal{EL}(G) \times \mathcal{M}(G)$ *is the transition relation defined by* $\mathsf{M} \xrightarrow{\phi} \mathsf{M}'$ *if there exists a timed execution* $\sigma_{\mathsf{M}}$ *of* $G$ *of length* $1$ *from* $\mathsf{M}$ *such that* $\sigma(0) = (\mathsf{M}, \phi, \mathsf{M}')$.

*Finally we define a zeno-run to be an infinite run with only finitely many time steps.*

It is worth noting that if the DCR Graph is finite, the reachable set of states for the corresponding labelled transition system will be finite.

**Lemma 3.1.** *The LTS for any finite timed DCR Graph* $G$ *has a finite number of reachable states.*

***Proof*** *The sets of executed events, pending responses and included events are limited to the power set of the finite set of events since* $(\mathsf{Ex}, \mathsf{Re}, \mathsf{In}) \in (\mathcal{P}(\mathsf{E}) \times \mathcal{P}(\mathsf{E}) \times \mathcal{P}(\mathsf{E}))$*. The execution times of executed events are limited to the maximum condition time, i.e.* $\forall e \in \mathsf{Ex} \mid t_{ex}(e) \in [0 \ldots max c_G]$*. The response times of pending responses are limited to the maximum response time or* $\omega$*, i.e.* $\forall e \in \mathsf{Re} \mid t_{re}(e) \in [0 \ldots \mathsf{max}\{k \mid \exists e, e' \in \mathsf{E}.e \bullet\!\!\xrightarrow{k} e'\}] \cup \omega$.

## 4. Safety and Liveness Properties

In this section we define safety and liveness properties of timed DCR Graphs and prove they are decidable for finite, bounded timed DCR Graphs.

First we identify the unwanted markings, referred to as time-locked, from where time can no longer progress. Note that zeno runs may still exist from a time-locked marking.

**Definition 4.1.** *For a timed Dynamic Condition Response Graph* $G = (\mathsf{E}, \mathsf{M_t}, \rightarrow\bullet, t_c, \bullet\rightarrow, t_r, \rightarrow\diamond, \rightarrow+, \rightarrow\%, \mathsf{L}, l)$ *with a timed marking* $\mathsf{M_t} = \big((\mathsf{Ex}, \mathsf{Re}, \mathsf{In}), t_{ex}, t_{re}\big)$ *we define that* $\mathsf{M_t}$ *is time-locked , written* $\mathsf{TL}(\mathsf{M_t})$*, if* $\forall \mathsf{M_t}' \in \mathcal{M}_{\mathsf{M_t}\rightarrow^*}.\neg \mathsf{M_t}' \vdash_G 1$*, meaning that there is no reachable marking from which time can progress. We define that G is* timelock free*, if* $\forall \mathsf{M}' \in \mathcal{M}_{\mathsf{M}\rightarrow^*}.\neg \mathsf{TL}(\mathsf{M}')$*, meaning that there exists no reachable time-locked marking.*

**Proposition 4.1.** *Time-lock and time-lock freedom is decidable for finite bounded DCR Graphs.*

***Proof*** *To determine if a marking* $\mathsf{M_t}$ *is time-locked, we must check for the existence of a transition labelled by a time-step from any of the reachable markings from* $\mathsf{M_t}$*, which is finite by Lem. 3.1. To determine time-lock freedom for a graph* $G$ *we just have to check every of the finitely many reachable markings from the initial marking if it is time-locked.*

A timed DCR Graph is said to be *deadlock free* if and only if for any reachable marking, there is either an enabled event or no included required responses. Furthermore, it is to be *strongly deadlock free* if and only if for any reachable marking, there is either an enabled event which is also a required response or no included required responses.

**Definition 4.2.** *For a timed Dynamic Condition Response Graph* $G = (\mathsf{E}, \mathsf{M_t}, \rightarrow\bullet, t_c, \bullet\rightarrow, t_r, \rightarrow\diamond, \rightarrow+, \rightarrow\%, \mathsf{L}, l)$ *with a timed marking* $\mathsf{M_t} = \big((\mathsf{Ex}, \mathsf{Re}, \mathsf{In}), t_{ex}, t_{re}\big)$

*we define that* $\mathsf{M_t}$ *is in* deadlock*, written* $\mathsf{DL}(\mathsf{M_t})$*, if* $\forall e \in \mathsf{E}.\mathsf{M_t} \not\models_G e \wedge (\mathsf{In} \cap \mathsf{Re} \neq \emptyset)$ *and that* $\mathsf{M_t}$ *is in* strong deadlock*, written* $\mathsf{SDL}(\mathsf{M_t})$*, if* $\forall e \in \mathsf{Re}.\mathsf{M_t} \not\models_G e \wedge (\mathsf{In} \cap \mathsf{Re} \neq \emptyset)$ *We define that G is* deadlock free*, if* $\forall \mathsf{M_t}' \in \mathcal{M}_{\mathsf{M_t} \rightarrow^*}.\neg \mathsf{DL}(\mathsf{M_t}')$ *and we say that G is* strongly deadlock free*, if* $\forall \mathsf{M_t}' \in \mathcal{M}_{\mathsf{M_t} \rightarrow^*}.\neg \mathsf{SDL}(\mathsf{M_t}')$.

**Proposition 4.2.** *Deadlock and strong deadlock freedom is decidable for finite DCR Graphs.*

***Proof*** *Follows easily from the definition and Lem. 3.1.*

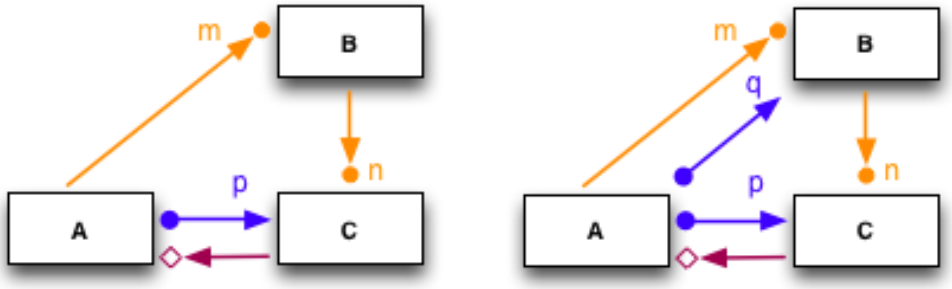
A timed DCR Graph is said to be *live* if and only if, in every reachable marking, it is always possible to continue along an accepting run. We say it is *strongly live* if and only if, from any reachable marking there exists an accepting must execution.

**Definition 4.3.** *For a timed Dynamic Condition Response Graph* $G = (\mathsf{E}, \mathsf{M_t}, \rightarrow\bullet, t_c, \bullet\rightarrow, t_r, \rightarrow\diamond, \rightarrow+, \rightarrow\%, \mathsf{L}, l)$ *we define that the G is* live*, if* $\forall \mathsf{M}' \in \mathcal{M}_{\mathsf{M} \rightarrow^*}.$ $\mathsf{acc}_{\mathsf{M}'}(G) \neq \emptyset$*, and* strongly live*, if* $\forall \mathsf{M}' \in \mathcal{M}_{\mathsf{M} \rightarrow^*}.\mathsf{macc}_{\mathsf{M}'}(G) \neq \emptyset$*,*

**Proposition 4.3.** *Liveness and strong liveness is decidable for finite DCR Graphs.*

***Proof*** *(Outline) First of all note that it is sufficient to consider the sub graph where all time steps increment the time with* 1*. In [25, 28] is given a construction of a Büchi-automaton from an un-timed DCR Graph which accepts the same executions as the DCR Graph and essentially having markings of the DCR Graph as states. This construction can be extended to timed DCR Graphs by adding the time functions to the markings (increasing the number of states by a constant factor), adding time steps and adding new marked copies of all states which are states reached by a time advance step from a state which would have been accepting for the underlying un-timed DCR Graph. These marked states will be the accepting states of the automata. The definition guarantees that an execution is accepting if and only if does not leave a response pending and included continuously (as for the un-timed DCR Graphs), and it does make a time step infinitely often. This is the definition of acceptance for timed DCR Graphs. Decidability of liveness and strong liveness then follows from the decidability of language emptiness of finite Büchi-automata.*

Fig. 4(a) shows a timed DCR Graph which, depending on the time constraints as shown in table 1, may have time-locks, deadlocks and violate liveness. The




(a) (b)

Figure 4: Examples of time-lock and (strongly) deadlock freedom in timed DCR Graphs

| | Time-lock free | Deadlock free | Live |
|---|---|---|---|
| $m > p$ | | | |
| $m \leq p \wedge n > 0 \wedge p < \omega$ | | x | |
| $(m \leq p \wedge n = 0) \vee p = \omega$ | x | x | x |

Table 1: Values for p, m and n in Fig. 4(a)

graph consist of three events: $A$, $B$, and $C$. The event $A$ is a condition for $B$, and the time deadline requires that the last execution of $A$ must have happened $m$ time steps before $B$ can happen. Similarly, the $B$ is a condition for $C$, and the time deadline requires that the last execution of $B$ must have happened $n$ time steps before $B$ can happen. The event $C$ is a response to the event $A$, and the deadline requires that it must happen within $p$ time steps after the last execution of $A$. The milestone relation from $C$ to $A$ is a kind of alternation pattern, it enforces that $A$ can not happen as long as $C$ is required as a response, i.e. two $A$s can not happen without at least one $C$ in between.

Now, if $m > p$ then the graph enters both a marking which is both time-locked and deadlocked if $A$ happens and time advances with $p$ steps. Since $B$ is a condition for $C$ it must be executed before $C$ can be executed, but this is not possible since the delay constraint $m$ is greater than $p$. The event $A$ can not be executed before $C$ has been executed due to the milestone relation, thus we have a deadlock. And since $C$ is required urgently, i.e. within deadline $0$, time can not progress either, i.e. we have a time-lock.

This deadlock can be resolved if $m \leq p$. Then $B$ can be executed before the deadline $p$ expires, and we can repeat doing $B$ infinitely often. However, the graph may still contain a time-lock, and thus also violate liveness: If $n > 0$ and $p < \omega$ and we do $B$ exactly $p$ time steps after $A$, then we can not do $C$ since it requires a delay of $n > 0$ after the execution of $B$, and time can not progress because $C$ is required urgently, i.e. within deadline $0$. The time-lock is resolved if $n = 0$ or $p = \omega$, since then we can either do $C$ just after $B$ (without advancing time) or the deadline on $C$ will never expire and lead to time-lock.

The graph in Fig. 4(a) is neither strongly deadlock free nor strongly live for any of the parameters. If $A$ is executed it is not possible to execute $C$ if only required events are executed. This can be resolved by making $B$ a response to $A$ as shown in Fig. 4(b). Strongly deadlock and live DCR Graphs may be needed if the process is distributed between different roles as considered in the next section. If $A$ and $B$ are carried out by role $N$, and $C$ by role $M$, then role $N$ may believe that it is not necessary to do more after doing $A$. However, role $M$ will then be stuck with a required action $C$ which is blocked because $B$ has not been executed.

## 5. Timed DCR Graphs as Global Contracts

The DCR Graphs model admits a very general technique for distributing a graph describing a global contract for e.g. a cross-organizational process, as a network of synchronously communicating graphs describing the contracts for each

local organization [27]. The technique is based on a notion of projection, restricting the graph to a subset of the events and labels. The projection introduces a notion of *interface* events, that can be regarded as a subscription to executions of events in other components.

In Fig. 5 it is shown how the technique can be used to project the global graph in Fig. 3 to three local graphs, representing the part of the process to be carried out by respectively the unions (U), the umbrella organization for the unions (LandsOrganisationen, LO) and the umbrella organization for the employers (Dansk Arbejdsgiverforening, DA). The Open case event with the double border in the LO local graph indicate that it needs to subscribe to the event Open case in order to know when it can and must propose dates and hold a meeting. The unions on the other hand need not subscribe to any external events, they only need to consider their own events Open case, Close case and Update case.

### *5.1. Projection*

First we define how to project a DCR Graph $G$ with respect to a *projection parameter* $\delta = (\delta_{\mathsf{E}}, \delta_{\mathsf{L}})$ where $\delta_{\mathsf{E}} \subseteq \mathsf{E}$ is a subset of the events of $G$ and $\delta_{\mathsf{L}} \subseteq \mathsf{L}$ is a subset of the labels.

Intuitively, the projection $G_{|\delta}$ contains only those events and relations that are relevant for the execution of events in $\delta_{\mathsf{E}}$ and the labeling is restricted to the set $\delta_{\mathsf{L}}$. This includes both the events in $\delta_{\mathsf{E}}$ and any other event that can affect the marking, or the ability to execute an event in $\delta_{\mathsf{E}}$. The technical difficulty is to infer the events and relations not in $\delta_{\mathsf{E}}$, referred to as *external events* below, that should be included in the projection because they influence the execution of the workflow restricted to the events in $\delta_{\mathsf{E}}$. The external events are never executed by the local component but can be executed by other components, in which case the local component must be notified so that it can update its marking accordingly.

**Definition 5.1.** *If* $G = (\mathsf{E}, \mathsf{M_t}, \rightarrow\bullet, t_c, \bullet\rightarrow, t_r, \rightarrow\diamond, \rightarrow+, \rightarrow\%, \mathsf{L}, l)$ *then* $G_{|\delta} = (\mathsf{E}_{|\delta}, \mathsf{M}_{\mathsf{t}|\delta}, \rightarrow\bullet_{|\delta}, t_{c|\delta}, \bullet\rightarrow_{|\delta}, t_{r|\delta}, \rightarrow\diamond_{|\delta}, \rightarrow+_{|\delta}, \rightarrow\%_{|\delta}, \delta_{\mathsf{L}}, l_{|\delta})$ *is the projection of* $G$ *with respect to* $\delta = (\delta_{\mathsf{E}}, \delta_{\mathsf{L}})$, $\delta_{\mathsf{E}} \subseteq \mathsf{E}$ *and* $\delta_{\mathsf{L}} \subseteq \mathsf{L}$ *where:*

*(i)* $\mathsf{E}_{|\delta} = \rightarrow \delta_{\mathsf{E}}$, *for* $\rightarrow = \bigcup_{c \in C} c$, *and* $C = \{\mathsf{id}, \rightarrow\bullet, \bullet\rightarrow, \rightarrow\diamond, \rightarrow+, \rightarrow\%, \bullet\rightarrow\rightarrow\diamond, \rightarrow+\rightarrow\bullet, \rightarrow\%\rightarrow\bullet, \rightarrow+\rightarrow\diamond, \rightarrow\%\rightarrow\diamond\}$

*(ii)* $l_{|\delta}(e) = \begin{cases} l(e) \cap \delta_{\mathsf{L}} & \text{if } e \in \delta_{\mathsf{E}} \\ \emptyset & \text{otherwise} \end{cases}$

*(iii)* $\mathsf{M}_{\mathsf{t}|\delta} = ((\mathsf{Ex}_{|\delta}, \mathsf{Re}_{|\delta}, \mathsf{In}_{|\delta}), t_{ex|\delta}, t_{re|\delta})$ *where:*

*(a)* $\mathsf{Ex}_{|\delta} = \mathsf{Ex} \cap \mathsf{E}_{|\delta}$

*(b)* $\mathsf{Re}_{|\delta} = \mathsf{Re} \cap (\delta_{\mathsf{E}} \cup \rightarrow\!\diamond\ \delta_{\mathsf{E}})$

*(c)* $\mathsf{In}_{|\delta} = \mathsf{In} \cap (\delta_{\mathsf{E}} \cup \rightarrow\!\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\!\diamond\ \delta_{\mathsf{E}})$

*(d)* $t_{ex|\delta}\ (e) = t_{ex}\ (e)$ *if* $e \in \mathsf{Ex}_{|\delta}$

*(e)* $t_{re|\delta}\ (e) = t_{re}\ (e)$ *if* $e \in \mathsf{Re}_{|\delta}$

*(iv)* $\rightarrow\!\bullet_{|\delta} = \rightarrow\!\bullet \cap ((\rightarrow\!\bullet\ \delta_{\mathsf{E}}) \times \delta_{\mathsf{E}})$

*(v)* $t_{c|\delta} : (\rightarrow\!\bullet_{|\delta}) \rightarrow \omega$
$t_{c|\delta}\ (e, e') = t_c\ (e, e')$

*(vi)* $\bullet\!\rightarrow_{|\delta} = \bullet\!\rightarrow \cap ((\bullet\!\rightarrow\!\rightarrow\!\diamond\ \delta_{\mathsf{E}}) \times (\rightarrow\!\diamond\ \delta_{\mathsf{E}})) \cup ((\bullet\!\rightarrow\ \delta_{\mathsf{E}}) \times \delta_{\mathsf{E}}))$

*(vii)* $t_{r|\delta} : (\bullet\!\rightarrow_{|\delta}) \rightarrow \infty$
$t_{r|\delta}\ (e, e') = t_r\ (e, e')$

*(viii)* $\rightarrow\!\diamond_{|\delta} = \rightarrow\!\diamond \cap ((\rightarrow\!\diamond\ \delta_{\mathsf{E}}) \times \delta_{\mathsf{E}})$

*(ix)* $\rightarrow\!+_{|\delta} = \rightarrow\!+ \cap \Big( \big((\rightarrow\!+\ \delta_{\mathsf{E}}) \times \delta_{\mathsf{E}}\big) \cup \big((\rightarrow\!+\!\rightarrow\!\bullet\ \delta_{\mathsf{E}}) \times (\rightarrow\!\bullet\ \delta_{\mathsf{E}}\big) \cup \big((\rightarrow\!+\!\rightarrow\!\diamond\ \delta_{\mathsf{E}}) \times (\rightarrow\!\diamond\ \delta_{\mathsf{E}})\big)\Big)$

*(x)* $\rightarrow\!\%_{|\delta} = \rightarrow\!\% \cap \Big( \big((\rightarrow\!\%\ \delta_{\mathsf{E}}) \times \delta_{\mathsf{E}}\big) \cup \big((\rightarrow\!\%\!\rightarrow\!\bullet\ \delta_{\mathsf{E}}) \times (\rightarrow\!\bullet\ \delta_{\mathsf{E}}\big) \cup \big((\rightarrow\!\%\!\rightarrow\!\diamond\ \delta_{\mathsf{E}}) \times (\rightarrow\!\diamond\ \delta_{\mathsf{E}})\big)\Big)$

In *(i)* we define the set of events as the union of the set $\delta_{\mathsf{E}}$ of events that we project over, any event that has a direct relation towards an event in $\delta_{\mathsf{E}}$ and events that exclude or include an event which is either a condition or a milestone for an event in $\delta_{\mathsf{E}}$. The additional events will be included in the projection without labels, as can be seen from the definition of the labeling function in *(**??**)*. This means that the events can not be executed locally. However, when composed in a network containing other processes that can execute these events, their execution will be communicated to the process. For this reason we refer to these events as the (additional) external events of the projection. As proven in Prop. A.1-A.3 the communication of the execution of this set of external events in addition to the

local events shared by others ensures that the local state of the projection stays consistent with the global state.

Further *(iii)* defines the projection of the marking: The executed events remain the same, but are limited to the events in $\mathsf{E}_{|\delta}$. The responses are restricted to events in $\delta_{\mathsf{E}}$ and events that have a milestone relation to an event in $\delta_{\mathsf{E}}$ because these are the only responses that will affect the local execution of the projected graph. Note that these events will by definition be events in $\mathsf{E}_{|\delta}$ but may be external events. In case of set of included events, we take the actual included status of the events in projection parameter along with the events that are conditions and milestones to the events in projection parameter, as the include status of those events will have an influence on the execution of events in local graph. All other external events of the projected graph are not included in the projected marking regardless of their included status in the marking of the global graph, because their include/exclude status will have no influence on the execution of events in local graph.

Finally, *(iv)*, *(vi)*, *(viii)*, *(ix)* and *(x)* state which relations should be included in the projection. For the events in $\delta_{\mathsf{E}}$ all incoming relations should be included. Additionally inclusion and exclusion relations to events that are either a condition or a milestone for an event in $\delta_{\mathsf{E}}$ are included as well.

Fig. 5 shows how the case management example from fig. 3 can be projected over the three roles. The projection parameters for these projections are: $\delta_U = (\{\mathsf{Open\ case}, \mathsf{Update\ Case}\}, \{(\mathsf{Open\ case}, \mathsf{U}), (\mathsf{Update\ Case}, \mathsf{U})\})$, $\delta_{LO} = (\{\mathsf{Propose\ Dates\ -\ LO}, \mathsf{Accept\ LO}, \mathsf{Extend\ Deadline}, \mathsf{Hold\ Meeting}\}$, $\{(\mathsf{Propose\ Dates\ -\ LO}, \mathsf{LO}), (\mathsf{Accept\ LO}, \mathsf{LO}), (\mathsf{Extend\ Deadline}, \mathsf{LO}),$ $(\mathsf{Hold\ Meeting}, \mathsf{LO})\})$ and $\delta_{DA} = (\{\mathsf{Propose\ Dates\ -\ DA}, \mathsf{Accept\ DA}\}$, $\{(\mathsf{Propose\ Dates\ -\ DA}, \mathsf{DA}), (\mathsf{Accept\ DA}, \mathsf{DA})\})$. From the figure one can see that the union needs to know nothing about the global contract. It can only open, close and update the case and because these events are not depending on any of the events of LO and DA, the union does not have to be aware of any other events. LO on the other hand needs to know about most of the events and relations in the contract because many of its internal events depend directly on events of the union or DA. This reflects the role of LO as an intermediary between the union and DA, which gives it a central role in the process. DA is only involved in arranging a meeting and needs to be aware of a few of the events of LO to properly play its role in this, but all other events that do not directly affect the meeting arrangement part of the contract are not relevant for DA and therefore do not have to be projected to the local graph for DA.

Prop. 5.1 below states the key correspondence between global execution of events and the local execution of events in a projection. We have provided the

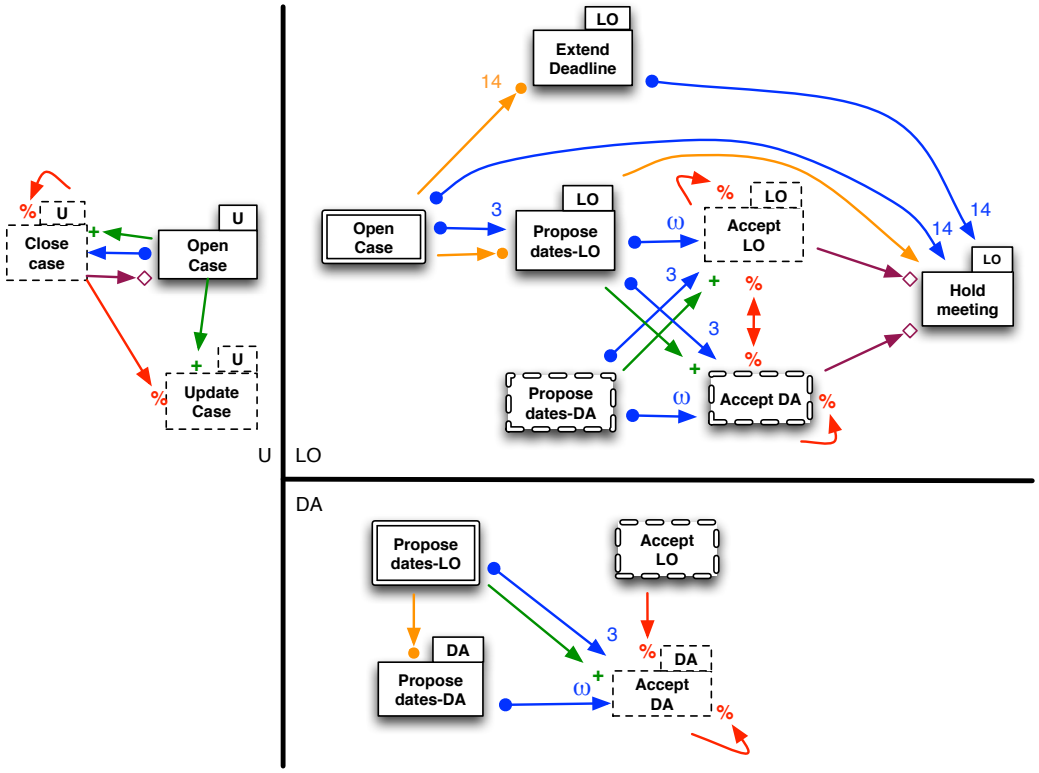



Figure 5: Projection of the timed DCR graph in fig. 3 over the roles U, LO and DA

details of the proof in the appendix. In order to prove the time extension, we first show the correspondence between global execution of events and the local execution of events in a projection of the underlying DCR Graph. We then use this result to prove the correspondence between the global and local execution of events for the time extension to the marking, $t_{ex}$ and $t_{re}$.

**Proposition 5.1.** *Let* $G = (\mathsf{E}, \mathsf{M_t}, \rightarrow\bullet, t_c, \bullet\rightarrow, t_r, \rightarrow\diamond, \rightarrow+, \rightarrow\%, \mathsf{L}, l)$ *be a Timed DCR Graph and* $G_{|\delta}$ *its projection with respect to a projection parameter* $\delta = (\delta_{\mathsf{E}}, \delta_{\mathsf{L}})$. *Then,*

1. *if* $e \in \delta_{\mathsf{E}}$ *and* $l(e) \cap \delta_{\mathsf{L}} \neq \emptyset$ *then* $\mathsf{M_t} \vdash_G e \wedge \mathsf{M_t} \oplus_G e = \mathsf{M_t}' \wedge \mathsf{M}'_{\mathsf{t}|\delta} = \mathsf{M_t}''$ *if and only if* $\mathsf{M}_{\mathsf{t}|\delta} \vdash_{G_{|\delta}} e \wedge \mathsf{M}_{\mathsf{t}|\delta} \oplus_{G_{|\delta}} e = \mathsf{M_t}''$

2. *if* $e \notin \mathsf{E}_{|\delta}$ *then* $\mathsf{M_t} \vdash_G e \wedge \mathsf{M_t} \oplus_G e = \mathsf{M_t}'$ *implies* $\mathsf{M}_{\mathsf{t}|\delta} = \mathsf{M}'_{\mathsf{t}|\delta}$

3. *if* $e \in \mathsf{E}_{|\delta}$ *(and* $l(e) \cap \delta_{\mathsf{L}} = \emptyset$*) then* $\mathsf{M_t} \vdash_G e \wedge \mathsf{M_t} \oplus_G e = \mathsf{M_t}'$ *implies* $\mathsf{M_{t|\delta}} \oplus_{G_{|\delta}} e = \mathsf{M'_{t|\delta}}$.

*Proof can be found in App. 1.*

**Example 5.1.** *We consider the projection in Fig. 5. Recall from Ex. 3.2 that the execution of* Open case *in the initial marking*

$$\mathsf{M_t} = (\emptyset, \emptyset, \mathsf{E} \setminus \{\textit{Accept LO}, \textit{Accept DA}, \textit{Close Case}, \textit{Update Case}\})$$

*gave rise to the marking*

$$\mathsf{M_t}' = (\{\textit{Open case} : 0\},\\ \{\textit{Propose dates-LO} : 3, \textit{Hold meeting} : 14, \textit{Close case} : \omega\},\\ \mathsf{E} \setminus \{\textit{Accept LO}, \textit{Accept DA}\})$$

*We now give the projected markings of* $\mathsf{M_t}$ *and* $\mathsf{M_t}$*' for the three projection parameters* $\delta_U = (\delta_E^U, \delta_L^U)$*,* $\delta_{LO} = (\delta_E^{LO}, \delta_L^{LO})$*, and* $\delta_{DA} = (\delta_E^{DA}, \delta_L^{DA})$*:*

$$\mathsf{M_{t|\delta_U}} = (\emptyset, \emptyset, \{\textit{Open case}\})$$

$$\mathsf{M'_{t|\delta_U}} = (\{\textit{Open case} : 0\}, \{\textit{Close Case} : \omega\},\\ \{\textit{Open case}, \textit{Close case}, \textit{Update case}\})$$

$$\mathsf{M_{t|\delta_{LO}}} = (\emptyset, \emptyset, \{\textit{Open case}, \textit{Propose dates-LO},\\ \textit{Extend deadline}, \textit{Hold meeting}\})$$

$$\mathsf{M'_{t|\delta_{LO}}} = (\{\textit{Open case} : 0\}, \{\textit{Propose dates-LO} : 3, \textit{Hold meeting} : 14\},\\ \{\textit{Open case}, \textit{Propose dates-LO}, \textit{Extend deadline}, \textit{Hold meeting}\})$$

$$\mathsf{M_{t|\delta_{DA}}} = \mathsf{M'_{t|\delta_{DA}}} = (\emptyset, \emptyset, \{\textit{Propose dates-LO}, \textit{Propose dates-DA}\})$$

*Let us verify that each of the properties 1.-3. in Prop. 5.1 holds for the projection parameters* $\delta_U$*,* $\delta_{LO}$ *and* $\delta_{DA}$ *when executing the event* $e =$ Open case *in the initial marking* $\mathsf{M_t}$.

1. *This property applies to the projection over the union because* $e \in \delta_{\mathsf{E}}^{\mathsf{U}}$ *and* $l(e) \cap \delta_{\mathsf{L}}^{\mathsf{U}} \neq \emptyset$. *First of all the property requires us to verify that* $e$ *is only enabled in the global graph if and only if it is enabled in the local graph. This is the case because both* $\mathsf{M_t} \vdash_G e$ *and* $\mathsf{M}_{\mathsf{t}|\delta_{\mathsf{U}}} \vdash_G e$. *Secondly the property requires us to verify that the result of executing* $e$ *on the global graph and then projecting the resulting marking over* $\delta_U$ *is the same as executing* $e$ *on the projected graph. This is the case because:* $\mathsf{M}_{\mathsf{t}|\delta_{\mathsf{U}}} \oplus_{G_{|\delta_{\mathsf{U}}}} e = \mathsf{M}'_{\mathsf{t}|\delta_{\mathsf{U}}}$.

2. *This property applies to the projection over DA because* $e \notin \mathsf{E}_{|\delta_{\mathsf{DA}}}$. *It requires us to verify that if* $e$ *does not occur in the events of the projected graph* $G_{|\delta_{\mathsf{DA}}}$, *then the marking projected over* $\delta_{DA}$ *will be the same before and after the execution of* $e$. *This is the case because* $\mathsf{M}_{\mathsf{t}|\delta_{\mathsf{DA}}} = \mathsf{M}'_{\mathsf{t}|\delta_{\mathsf{DA}}}$.

3. *This property applies to the projection over LO because* $e \in \mathsf{E}_{|\delta_{\mathsf{LO}}}$ *and* $l(e) \cap \delta_L^{\mathsf{LO}} = \emptyset$. *It requires us to show that if* $e$ *is an external event in the projected graph* $G_{|\delta_{\mathsf{LO}}}$, *then it must be the case that if we execute* $e$ *on* $G_{|\delta_{\mathsf{LO}}}$, *the resulting marking must be the same as if we had first executed* $e$ *on* $G$ *and then projected the result over* $\delta_{LO}$. *This is the case because* $\mathsf{M}_{\mathsf{t}|\delta_{\mathsf{LO}}} \oplus_{G_{|\delta_{\mathsf{LO}}}} e = \mathsf{M}'_{\mathsf{t}|\delta_{\mathsf{LO}}}$.

### *5.2. Composition*

Now we define the binary composition of two DCR Graphs. The *composition* of $G_1$ and $G_2$ is simply the component-wise union of the respective components.

**Definition 5.2.** $G_1 \cup G_2 = (\mathsf{E}_1 \cup \mathsf{E}_2, \mathsf{M_t}, \rightarrow\bullet_1 \cup \rightarrow\bullet_2, t_{c1} \cup t_{c2}, \bullet\rightarrow_1 \cup \bullet\rightarrow_2, t_{r1} \cup t_{r2}, \rightarrow\diamond_1 \cup \rightarrow\diamond_2, \rightarrow+_1 \cup \rightarrow+_2, \rightarrow\%_1 \cup \rightarrow\%_2, \mathsf{L}_1 \cup \mathsf{L}_2, l_1 \cup l_2)$, *where*
$\mathsf{M_t} = ((\mathsf{Ex}_1 \cup \mathsf{Ex}_2, \mathsf{Re}_1 \cup \mathsf{Re}_2, \mathsf{In}_1 \cup \mathsf{In}_2), t_{ex1} \cup t_{ex2}, t_{re1} \cup t_{re2})$

**Definition 5.3.** *The composition* $G_1 \cup G_2$ *is* well-defined *when:*

*(i)* $\forall(e \in \mathsf{E}_1 \cap \mathsf{E}_2 \mid (e \in \mathsf{Ex}_1 \Leftrightarrow e \in \mathsf{Ex}_2))$

*(ii)* $\forall(e \in (\mathsf{E}_1^{\mathsf{i}} \cup \rightarrow\bullet \mathsf{E}_1^{\mathsf{i}} \cup \rightarrow\diamond \mathsf{E}_1^{\mathsf{i}}) \cap (\mathsf{E}_2^{\mathsf{i}} \cup \rightarrow\bullet \mathsf{E}_2^{\mathsf{i}} \cup \rightarrow\diamond \mathsf{E}_2^{\mathsf{i}}) \mid (e \in \mathsf{In}_1 \Leftrightarrow e \in \mathsf{In}_2)$

*(iii)* $\forall(e \in (\mathsf{E}_1^{\mathsf{i}} \cup \rightarrow\diamond \mathsf{E}_1^{\mathsf{i}}) \cap (\mathsf{E}_2^{\mathsf{i}} \cup \rightarrow\diamond \mathsf{E}_2^{\mathsf{i}}) \mid (e \in \mathsf{Re}_1 \Leftrightarrow e \in \mathsf{Re}_2)$

*(iv)* $\forall(e \in \mathsf{Ex}_1 \cap \mathsf{Ex}_2 \mid (t_{c1}\ e = t_{c2}\ e))$

*(v)* $\forall(e \in \mathsf{Re}_1 \cap \mathsf{Re}_2 \mid (t_{r1}\ e = t_{r2}\ e))$

*(vi)* $\forall(e, e' \in \rightarrow\bullet_1 \cap \rightarrow\bullet_2 |\ t_{c1}(e, e') = t_{c2}(e, e'))$

*(vii)* $\forall(e, e' \in \bullet\rightarrow_1 \cap \bullet\rightarrow_2 |\ t_{r1}(e, e') = t_{r2}(e, e'))$

*Where:* $\mathsf{E}_i^{\mathsf{i}} = \{e \in \mathsf{E}_i \mid l(e) \neq \emptyset\}$ *for* $i \in \{1, 2\}$

*(i)* ensures that those events that will be glued together have the same execution marking. *(ii)* ensures that events that will be glued together and in both DCR Graphs belong to either the set of internal events or the set of events that have a condition/milestone relation towards an internal event, have the same inclusion marking. *(iii)* ensures that events that will be glued together and in both DCR Graphs belong to the set of internal events have the same pending response marking. *(iv)* ensures that executed events that will be glued together have the same timed execution marking. *(v)* ensures that responses on events that will be glued together have the same deadline. *(vi)* ensures that shared conditions have the same required time delay. *(vii)* ensures that shared responses have the same maximal deadline. If $G_1 \cup G_2$ is well-defined, then we also say that $G_1$ and $G_2$ are *composable* with respect to each other.

**Lemma 5.1.** *The composition operator* $\cup$ *is commutative and associative.*

**Proof** According to definition 5.2, elements of the tuple defining the graph $G = G_1 \cup G_2$ are constructed from the union of the same elements in $G_1$ and $G_2$. Composition is therefore commutative and associative, because the union operator is commutative and associative.

**Definition 5.4.** *We call a vector* $\Delta = \delta_1 \ldots \delta_k$ *of projection parameters* covering *for some DCR Graph* $G = (\mathsf{E}, \mathsf{M_t}, \rightarrow\bullet, t_c, \bullet\rightarrow, t_r, \rightarrow\diamond, \rightarrow+, \rightarrow\%, \mathsf{L}, l)$ *if:*

1. $\bigcup_{i \in [k]} \delta_{\mathsf{E}i} = \mathsf{E}$ *and*
2. $(\forall a \in \mathsf{L}. \forall e \in \mathsf{E}. a \in l(e) \Rightarrow (\exists i \in [k]. e \in \delta_{\mathsf{E}i} \wedge a \in \delta_{\mathsf{L}i})$

**Proposition 5.2.** *If some vector* $\Delta = \delta_1 \ldots \delta_k$ *of projection parameters is covering for some DCR Graph* $G$ *then:* $\bigcup_{i \in [k]} G_{|\delta_i} = G$

***Proof*** *Since the vector of projection parameters is covering, every event and label is covered in at least one of the projections. Moreover the definition of composition 5.2, is defined over union of individual components. Hence when all projections are composed, we will get the same graph and hence* $\bigcup_{i \in [k]} G_{|\delta_i} = G$.

*5.3. Safe Distributed Synchronous Execution of Timed DCR Graphs*

Networks of timed DCR Graphs are defined exactly as networks of un-timed DCR Graphs introduced in [27].

**Definition 5.5.** *A* network of timed DCR Graphs *is a finite vector of timed DCR Graphs* $\overline{G}$ *sometimes written as* $\Pi_{i\in[n]}G_i$ *or* $G_0\|G_2\|\ldots\|G_{n-1}$*. Assuming* $G_i = (\mathsf{E}_i, \mathsf{M}_i, \rightarrow\bullet_i, t_{ci}, \bullet\rightarrow_i, t_{ri}, \rightarrow\diamond_i, \rightarrow+_i, \rightarrow\%_i, \mathsf{L}_i, l_i)$*, we define the set of events of the network by* $\mathcal{E}(\Pi_{i\in[n]}G_i) = \cup_{i\in[n]}\mathsf{E}_i$ *and the set of labels of the network by* $\mathcal{L}(\Pi_{i\in[n]}G_i) = \cup_{i\in[n]}\mathsf{L}_i$ *and we write the network marking as* $\overline{\mathsf{M}} = \Pi_{i\in[n]}\mathsf{M}_i$*. Finally, let* $\mathcal{M}(\overline{G})$ *denote the set of network markings of* $\overline{G}$*.*

We define when an event is locally enabled in one of the components and the result of executing an event locally as for un-timed DCR Graphs, except that we instead use the definition of timed enabled and timed update of the marking. That is, an event can be executed if it is locally enabled, and the result of executing it is that it is synchronously executed in all components of the network sharing the event. A time advance event is however defined to be enabled only if it is enabled in *all* components and the result of advancing the time is to advance it in all components. This ensures that time advances globally in the network.

**Definition 5.6.** *For a network of timed DCR Graphs* $\overline{G} = \Pi_{i\in[n]}G_i$ *where* $G_i = (\mathsf{E}_i, \mathsf{M}_i, \rightarrow\bullet_i, t_{ci}, \bullet\rightarrow_i, t_{ri}, \rightarrow\diamond_i, \rightarrow+_i, \rightarrow\%_i, \mathsf{L}_i, l_i)$*, an event* $e \in \mathcal{E}(\Pi_{i\in[n]}G_i)$ *is enabled at a location* $i$ *in the distributed marking* $\overline{\mathsf{M}} = \Pi_{i\in[n]}\mathsf{M}_i$*, written* $\overline{\mathsf{M}} \vdash_{\overline{G},i} e$*, if* $e \in \mathsf{E}_i \wedge \mathsf{M}_i \vdash_{G_i} e$*, i.e. it is locally enabled in the* $i$*th timed dynamic condition response graph. The result of executing an event* $e \in \mathcal{E}(\Pi_{i\in[n]}G_i)$ *in a marking* $\overline{\mathsf{M}} = \Pi_{i\in[n]}\mathsf{M}_i$ *is the new marking* $\overline{\mathsf{M}} \oplus_{\overline{G}_i} e = \Pi_{i\in[n]}\mathsf{M}'_i$ *where* $\mathsf{M}'_i = \mathsf{M}_i \oplus_G e$ *if* $e \in \mathsf{E}_i$ *and* $\mathsf{M}'_i = \mathsf{M}_i$ *otherwise. For a network marking* $\overline{\mathsf{M}} = \Pi_{i\in[n]}\mathsf{M}_i$ *we define that the time advance event* $n$ *is enabled, written* $\overline{\mathsf{M}} \vdash_{\overline{G}} n$*, if* $\mathsf{M}_i \vdash_{G_i} n$ *for all* $i \in [n]$ *and the result of advancing time with* $n$ *by* $\overline{\mathsf{M}} \oplus_{\overline{G}} n = \Pi_{i\in[n]}\mathsf{M}_i \oplus_{G_i} n$*.*

We define executions of networks as follows. As for un-timed DCR Graphs, an event can be executed if it is locally enabled in a component where it has assigned at least one label. Time can be advanced globally if the time advance event is enabled as defined above.

**Definition 5.7.** *For a network of timed DCR Graphs* $\overline{G} = \Pi_{i\in[n]}G_i$ *where* $l_i$ *is the labeling function of* $G_i$*, we define a* timed execution $\sigma_{\overline{\mathsf{M}}}$ *of* $\overline{G}$ *of length* $k \in \infty$ *from marking* $\overline{\mathsf{M}}$ *to be a (finite or infinite) sequence of tuples* $\sigma : [k] \rightarrow \mathcal{M}(\overline{G}) \times ([n] \times \mathcal{E}(\Pi_{i\in[n]}G_i) \times \mathcal{L}(\Pi_{i\in[n]}G_i) \cup \omega) \times \mathcal{M}(\overline{G})$ *such that for* $i \in [k]$

- $\sigma(i) = (\overline{\mathsf{M}}_i, h_i, e_i, a_i, \overline{\mathsf{M}}_i') \wedge a_i \in l_{h_i}(e_i) \wedge \overline{\mathsf{M}}_i \vdash_{\overline{G},h_i} e_i \wedge \overline{\mathsf{M}}_i' = \overline{\mathsf{M}}_i \oplus_{\overline{G}} e_i$ *or*
- $\sigma(i) = (\overline{\mathsf{M}}_i, n, \overline{\mathsf{M}}_i') \wedge \overline{\mathsf{M}}_i \vdash_{\overline{G}} n \wedge \overline{\mathsf{M}}_i' = \overline{\mathsf{M}} \oplus_{\overline{G}} n$

*and* $\overline{\mathsf{M}}_0 = \overline{\mathsf{M}}$ *and* $\forall i \in [k-1].\overline{\mathsf{M}}_i' = \overline{\mathsf{M}}_{i+1}$.

*We say the execution is* accepting *if*

*(i)* $\forall i \in [k], h \in [n].\big(\forall e \in \mathsf{In}_{h,i} \cap \mathsf{Re}_{h,i}.\exists j \geq i.e_j = e \vee e \notin \mathsf{In}'_{h,j}\big)$, *where* $\overline{\mathsf{M}}_i = \Pi_{h\in[n]}(\mathsf{Ex}_{h,i}, \mathsf{In}_{h,i}, \mathsf{Re}_{h,i})$ *and* $\overline{\mathsf{M}}_j' = \Pi_{h\in[n]}(\mathsf{Ex}'_{h,j}, \mathsf{In}'_{h,j}, \mathsf{Re}'_{h,j})$

*(ii)* $\forall i \in \omega.\exists h \in \omega.\exists j > i.\sigma(i) = (\overline{\mathsf{M}}_i, h, \overline{\mathsf{M}}_i')$

We define the global transition system for a network of timed DCR Graphs as follows.

**Definition 5.8.** *For a network of timed DCR Graphs* $\overline{G} = \Pi_{i\in[n]}G_i$ *where* $G_i = (\mathsf{E}_i, \mathsf{M}_i, \rightarrow\bullet_i, t_{ci}, \bullet\rightarrow_i, t_{ri}, \rightarrow\diamond_i, \rightarrow+_i, \rightarrow\%_i, \mathsf{L}_i, l_i)$ *and* $\overline{\mathsf{M}} = \Pi_{i\in[n]}\mathsf{M}_i$*, we define the corresponding global transition system* $\mathsf{TS}(\overline{G})$ *to be the tuple*

$$(\mathcal{M}(\overline{G}), \overline{\mathsf{M}}, \mathcal{EL}(\overline{G}), \rightarrow_N)$$

*where* $\mathcal{EL}(\overline{G}) = (\mathcal{E}(\Pi_{i\in[n]}G_i) \times \mathcal{L}(\Pi_{i\in[n]}G_i)) \cup \omega$ *is the set of labels of the transition system,* $\overline{\mathsf{M}}$ *is the initial marking, and* $\rightarrow_N \subseteq \mathcal{M}(\overline{G}) \times \mathcal{EL}(\overline{G}) \times \mathcal{M}(\overline{G})$ *is the transition relation defined by* $\overline{\mathsf{M}}' \xrightarrow{(e,a)}_N \overline{\mathsf{M}}''$ *if there is a timed execution* $\sigma_{\overline{\mathsf{M}}'}$ *from* $\overline{\mathsf{M}}'$ *of length* 1 *such that* $\sigma_{\overline{\mathsf{M}}'}(0) = (\overline{\mathsf{M}}', i, e, a, \overline{\mathsf{M}}'')$ *for some* $i \in [n]$ *and* $\overline{\mathsf{M}}' \xrightarrow{h}_N \overline{\mathsf{M}}''$ *if there is a timed execution* $\sigma_{\overline{\mathsf{M}}'}$ *from* $\overline{\mathsf{M}}'$ *of length* 1 *such that* $\sigma_{\overline{\mathsf{M}}'}(0) = (\overline{\mathsf{M}}', h, \overline{\mathsf{M}}'')$*. (Accepting) executions of length* $k$ *of the transition system is defined as sequences transitions obtained similarly from (accepting) executions* $\sigma$ *of length* $k$ *of the graph.*

We are now ready to state the main theorem which follows from Def 3.4, Def. 5.8 and Prop. 5.1.

**Theorem 5.1.** *For a timed DCR Graph* $G$*, a covering vector of projection parameters* $\Delta = \delta_1 \ldots \delta_n$ *and* $\overline{G}_\Delta = \Pi_{i\in[n]}G_{|\delta_i}$ *it holds that the relation* $\mathcal{R} = \{(\mathsf{M}, \overline{\mathsf{M}}_\Delta) \mid \mathsf{M} \in \mathcal{M}(G)$ *and* $\overline{\mathsf{M}}_\Delta = \Pi_{i\in[n]}\mathsf{M}_{|\delta_i}\}$ *is a bisimulation between* $\mathsf{TS}(G)$ *and* $\mathsf{TS}(\overline{G}_\Delta)$ *such that an execution is accepting in* $\mathsf{TS}(G)$ *if and only if the bisimilar execution is accepting in* $\mathsf{TS}(\overline{G}_\Delta)$.

**Proof** (outline) In order to prove bisimilarity, we show (1) $\exists \mathsf{M} \xrightarrow{(e,a)} \mathsf{M}'$ in $TS(G)$ if and only if $\exists \overline{\mathsf{M}}_{\Delta} \xrightarrow{(e,a)} \overline{\mathsf{M}'}_{\Delta}$ in $\mathsf{TS}(\overline{G}_{\Delta})$ and (2) $\exists \mathsf{M} \xrightarrow{h} \mathsf{M}'$ in $TS(G)$ if and only if $\exists \overline{\mathsf{M}}_{\Delta} \xrightarrow{h} \overline{\mathsf{M}'}_{\Delta}$ in $\mathsf{TS}(\overline{G}_{\Delta})$.

1. According to Def 3.4, $\mathsf{M} \xrightarrow{(e,a)} \mathsf{M}'$ in $TS(G)$ implies $\mathsf{M} \xrightarrow{(e,a)} \mathsf{M} \oplus_G e$, $\mathsf{M} \vdash_G e$ and $a \in \mathrm{l}(e)$. From Prop. 5.1 it then follows that $\overline{\mathsf{M}}_{\Delta} \xrightarrow{(e,a)} \overline{\mathsf{M}'}_{\Delta}$. Similarly, if $\overline{\mathsf{M}}_{\Delta} \xrightarrow{(e,a)} \overline{\mathsf{M}'}_{\Delta}$ in $\mathsf{TS}(\overline{G}_{\Delta})$ then again using Prop. 5.1, there exists in $TS(G)$ a transition $\mathsf{M} \xrightarrow{(e,a)} \mathsf{M}'$.

2. Since the time advances globally and synchronously in the network, it easily follows that time advance steps can be mutually simulated and result in consistent updates of the deadlines in the markings.

Now we have to prove that a timed execution in $\mathsf{TS}(G)$ is accepting if and only if the corresponding execution is accepting in $\mathsf{TS}(\overline{G}_{\Delta})$. If an execution in $\mathsf{TS}(\overline{G}_{\Delta})$ is accepting, then

1. according to Def. 5.8, in the network of projected graphs, $\forall i \in [\underline{k}], h \in [n].\big(\forall e \in \mathsf{In}_{|\delta_{h,i}} \cap \mathsf{Re}_{|\delta_{h,i}}.\exists j \geq i.e_j = e \vee e \notin \mathsf{In}'_{|\delta_{h,i}}\big)$, where $\overline{\mathsf{M}}_{\Delta_i} = \Pi_{h\in[n]}(\mathsf{Ex}_{|\delta_{h,i}}, \mathsf{In}_{|\delta_{h,i}}, \mathsf{Re}_{|\delta_{h,i}})$ and $\overline{\mathsf{M}}'_{\Delta_j} = \Pi_{h\in[n]}(\mathsf{Ex}'_{|\delta_{h,i}}, \mathsf{In}'_{|\delta_{h,i}}, \mathsf{Re}'_{|\delta_{h,i}})$. According to proposition 5.1 and since $\mathsf{TS}(G) \sim \mathsf{TS}(\overline{G}_{\Delta})$, if there exists an execution where an included response is eventually executed or excluded in $\mathsf{TS}(\overline{G}_{\Delta})$, then we will also have the corresponding execution satisfying that an included response is eventually executed or excluded in $\mathsf{TS}(G)$.

2. As the time advances globally in the network, $\forall i \in \omega.\exists h \in \omega.\exists j > i.\sigma(i) = (\overline{\mathsf{M}}_i, h, \overline{\mathsf{M}}'_i)$ in $\mathsf{TS}(\overline{G}_{\Delta})$ implies that $\forall i \in \omega.\exists h \in \omega.\exists j > i.\sigma(i) = (\mathsf{M}_i, h, \mathsf{M}'_i)$ in $\mathsf{TS}(G)$.

Therefore, if a timed execution is accepting in $\mathsf{TS}(\overline{G}_{\Delta})$ then it is also accepting in $\mathsf{TS}(G)$. On the similar lines, it trivially follows that if a timed execution in $\mathsf{TS}(G)$ is accepting the corresponding execution in $\mathsf{TS}(\overline{G}_{\Delta})$ is also accepting.

## 6. Conclusion

We have conservatively extended the declarative Dynamic Condition Response (DCR) Graph process model introduced in the PhD thesis of the second author [24,

28] to allow for (discrete) time deadlines. In particular, the simple operational semantics of DCR Graphs is conservatively extended with time steps, preserving that safety and liveness properties of the models can be formally verified by mapping bounded timed DCR Graphs to finite state automata. We proceeded to extend to timed DCR Graphs the general technique provided in [27] for safe distribution of a global DCR Graph as a network of communicating DCR Graphs which has the same behavior as the global DCR Graph. We have exemplified timed DCR Graphs and the distribution technique on a timed extension of the cross-organizational case management process studied in [34, 27]. The example shows how a timed DCR Graph can be used to describe the global contract for a timed workflow process involving several organizations, which can then be distributed as a network of communicating timed DCR Graphs, having a graph describing the local contract for each organization. Finally we have exemplified how the time deadlines may introduce both deadlocks and so-called time-locks where time cannot proceed in the model.

We plan for future work to study the application of the techniques in the present paper for contract-oriented programming of distributed event-based systems and context-sensitive services. In particular, we are currently developing an event-based programming language based on an extension of DCR Graphs with data and sub processes. We will also study the formal relation between timed DCR Graphs and other timed process models, in particular timed LTL [21], variants of Petri Net with time [63, 68, 71, 72] and timed automata [], and explore verification of timed DCR Graphs using existing tools for these models. Finally we intend to investigate the relation to the work on structured communication-centred programming for web services by Carbone, Honda and Yoshida [46].

## References


[1] M. Dumas, W. M. van der Aalst, A. H. ter Hofstede, Process Aware Information Systems: Bridging People and Software Through Process Technology, Wiley-Interscience, 2005.

[2] M. D. Zisman, Representation, Specification and Automation of Office Procedures, Ph.D. thesis, Wharton School, University of Pennsylvania, 1977.

[3] J. C. BURNS, The evolution of office information systems, Datamation vol. 23,no. 4 (1977) 60–64.

[4] C. A. Ellis, Information control nets: A mathematical model of office information flow, Proceedings of the Conference on Simulation, Measurement and Modeling of Computer Systems, ACM SIGMETRICS Performance Evaluation Review 8 (1979) 225–240.

[5] C. A. Ellis, G. J. Nutt, Office information systems and computer science, ACM Comput. Surv. 12 (1980) 27–60.

[6] C. A. Petri, Kommunikation mit Automaten, Ph.D. thesis, Universitet Hamburg, 1962.

[7] H. Eshuis, Semantics and Verification of UML Activity Diagrams for Workflow Modelling, Ph.D. thesis, Univ. of Twente, 2002. CTIT Ph.D.-thesis series No. 02-44.

[8] R. Eshuis, R. Wieringa, Tool support for verifying uml activity diagrams, Software Engineering, IEEE Transactions on 30 (2004) 437 – 447.

[9] Object Management Group BPMN Technical Committee, Business Process Model and Notation, version 2.0, Webpage, 2011. `http://www.omg.org/spec/BPMN/2.0/PDF`.

[10] W. M. P. van der Aalst, Interorganizational workflows: An approach based on message sequence charts and petri nets., Systems Analysis - Modelling - Simulation 34 (1999) 335–367.

[11] E. Kindler, A. Martens, W. Reisig, Inter-operability of workflow applications: Local criteria for global soundness, in: Business Process Management, Models, Techniques, and Empirical Studies, Springer-Verlag, London, UK, 2000, pp. 235–253.

[12] W. M. P. van der Aalst, N. Lohmann, P. Massuthe, C. Stahl, K. Wolf, Multiparty Contracts: Agreeing and Implementing Interorganizational Processes, The Computer Journal 53 (2010) 90–106.

[13] W. M. P. v. d. Aalst, M. Weske, The p2p approach to interorganizational workflows, in: Proceedings of the 13th International Conference on Advanced Information Systems Engineering, CAiSE '01, pp. 140–156.

[14] W. Vanderaalst, M. Weske, D. Grunbauer, Case handling: a new paradigm for business process support, Data & Knowledge Engineering 53 (2005) 129–162.

[15] M. Pesic, M. H. Schonenberg, N. Sidorova, W. M. P. Van Der Aalst, Constraint-based workflow models: change made easy, in: Proceedings of the 2007 OTM Confederated international conference on On the move to meaningful internet systems: CoopIS, DOA, ODBASE, GADA, and IS - Volume Part I, OTM'07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 77–94.

[16] A. Pnueli, The temporal logic of programs, in: Proceedings of 18th IEEE FOCS, pp. 46–57.

[17] E. M.Clarke, O. Grumberg, D. A.Peled, Model Checking, MIT Press, 1999.

[18] M. Papazoglou, Making business processes compliant to standards and regulations, in: Enterprise Distributed Object Computing Conference (EDOC), 2011 15th IEEE International, pp. 3 –13.

[19] M. Dwyer, G. Avrunin, J. Corbett, Patterns in property specifications for finite-state verification, in: Software Engineering, 1999. Proceedings of the 1999 International Conference on, pp. 411 –420.

[20] W. M. van der Aalst, M. Pesic, DecSerFlow: Towards a truly declarative service flow language, in: M. Bravetti, M. Nunez, G. Zavattaro (Eds.), Proceedings of Web Services and Formal Methods (WS-FM 2006), volume 4184 of *LNCS*, Springer Verlag, 2006, pp. 1–23.

[21] K. J. Kristoffersen, C. Pedersen, H. R. Andersen, Runtime verification of timed ltl using disjunctive normalized equation systems, Electronic Notes in Theoretical Computer Science 89 (2003) 210 – 225.

[22] G. J. Holzmann, The model checker spin, IEEE Trans. Softw. Eng. 23 (1997) 279–295.

[23] M. Y. Vardi, An automata-theoretic approach to linear temporal logic, in: Logics for Concurrency: Structure versus Automata, volume 1043 of Lecture Notes in Computer Science, Springer-Verlag, 1996, pp. 238–266.

[24] T. T. Hildebrandt, R. R. Mukkamala, Declarative event-based workflow as distributed dynamic condition response graphs, in: K. Honda, A. Mycroft (Eds.), PLACES, volume 69 of *EPTCS*, pp. 59–73.

[25] R. Mukkamala, T. Hildebrandt, From dynamic condition response structures to büchi automata, in: Theoretical Aspects of Software Engineering (TASE), 2010 4th IEEE International Symposium on, pp. 187 –190.

[26] T. Hildebrandt, R. R. Mukkamala, T. Slaats, Nested dynamic condition response graphs, in: Proceedings of Fundamentals of Software Engineering (FSEN).

[27] T. Hildebrandt, R. R. Mukkamala, T. Slaats, Safe distribution of declarative processes, in: Proceedings of the 9th international conference on Software engineering and formal methods, SEFM'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 237–252.

[28] R. R. Mukkamala, A Formal Model For Declarative Workflows - Dynamic Condition Response Graphs, Ph.D. thesis, IT University of Copenhagen, 2012. Forthcomming.

[29] T. Hildebrandt, Trustworthy pervasive healthcare processes (TrustCare) research project, Webpage, 2008. `http://www.trustcare.dk/`.

[30] R. R. Mukkamala, T. Hildebrandt, J. B. Tøth, The resultmaker online consultant: From declarative workflow management in practice to ltl, in: Proceedings of the 2008 12th Enterprise Distributed Object Computing Conference Workshops, EDOCW '08, IEEE Computer Society, Washington, DC, USA, 2008, pp. 135–142.

[31] Resultmaker, 2008. `http://www.resultmaker.com/`.

[32] T. Hildebrandt, Dynamic condition response graphs - a dynamic temporal logic for event-based processes, in: 8th Scandinavian Logic Symposium, pp. 52–54.

[33] T. Hildebrandt, R. R. Mukkamala, T. Slaats, Verifying liveness and safety for declarative event-based workflows in spin, 2012. Sumitted for publication.

[34] T. Hildebrandt, R. Mukkamala, T. Slaats, Designing a cross-organizational case management system using dynamic condition response graphs, in: Enterprise Distributed Object Computing Conference (EDOC), 2011 15th IEEE International, pp. 161 –170.

[35] M. Pesic, W. M. P. van der Aalst, A declarative approach for flexible business processes management, in: Proceedings of the 2006 international conference on Business Process Management Workshops, BPM'06, Springer-Verlag, Berlin, Heidelberg, 2006, pp. 169–180.

[36] F. Maria Maggi, M. Montali, M. Westergaard, W. M. P. van der Aalst, Monitoring business constraints with linear temporal logic: An approach based on colored automata, in: Business Process Management (BPM) 2011, volume 6896 of *Lecture Notes in Computer Science*, pp. 32–147.

[37] A. ter Hofstede, R. van Glabbeek, D. Stork, Query nets: Interacting workflow modules that ensure global termination, in: Business Process Management, Springer Berlin / Heidelberg, 2003, pp. 184–199.

[38] W. van der Aalst, Inheritance of interorganizational workflows: How to agree to disagree without loosing control?, Information Technology and Management 4 (2003) 345–389.

[39] A. Martens, Analyzing web service based business processes, in: Proceedings of the 8th international conference, held as part of the joint European Conference on Theory and Practice of Software conference on Fundamental Approaches to Software Engineering, FASE'05, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 19–33.

[40] X. Fu, T. Bultan, J. Su, Realizability of conversation protocols with message contents, in: Proceedings of the IEEE International Conference on Web Services, ICWS '04, IEEE Computer Society, Washington, DC, USA, 2004, pp. 96–.

[41] X. Yi, K. Kochut, Process composition of web services with complex conversation protocols., in: Design, Analysis, and Simulation of Distributed Systems Symposium at Adavanced Simulation Technology.

[42] S. Rinderle, A. Wombacher, M. Reichert, Evolution of process choreographies in dychor, in: On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE, volume 4275 of *LNCS*, Springer, 2006, pp. 273–290.

[43] D. Wodtke, G. Weikum, A formal foundation for distributed workflow execution based on state charts, in: Proceedings of the 6th International Conference on Database Theory, Springer-Verlag, London, UK, 1997, pp. 230–246.

[44] M. Bravetti, G. Zavattaro, Contract based multi-party service composition, in: International Symposium on Fundamentals of Software Engineering (FSEN), volume 4767, Springer, 2007, pp. 207–222.

[45] M. Bravetti, G. Zavattaro, A theory of contracts for strong service compliance, Mathematical. Structures in Comp. Sci. 19 (2009) 601–638.

[46] M. Carbone, K. Honda, N. Yoshida, Structured Communication-Centred Programming for Web Services, in: 16th European Symposium on Programming (ESOP'07), LNCS, Springer, 2007, pp. 2–17.

[47] W. Fdhila, C. Godart, Toward synchronization between decentralized orchestrations of composite web services., in: CollaborateCom'09, pp. 1–10.

[48] M. G. Nanda, S. Chandra, V. Sarkar, Decentralizing execution of composite web services, SIGPLAN Not. 39 (2004) 170–187.

[49] R. Khalaf, F. Leymann, Role-based decomposition of business processes using BPEL, in: Web Services, 2006. ICWS '06. International Conference on, pp. 770 –780.

[50] S. Mitra, R. Kumar, S. Basu, Optimum decentralized choreography for web services composition, in: Proceedings of the 2008 IEEE International Conference on Services Computing - Volume 2, pp. 395 –402.

[51] OASIS WSBPEL Technical Committee, Web Services Business Process Execution Language, version 2.0, 2007. `http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf`.

[52] I. Castellani, M. Mukund, P. Thiagarajan, Synthesizing distributed transition systems from global specifications, in: Foundations of Software Technology and Theoretical Computer Science, volume 1738, Springer Berlin / Heidelberg, 1999, pp. 219–231.

[53] K. Heljanko, A. Stefanescu, Complexity results for checking distributed implementability, in: Proceedings of the Fifth International Conference on Application of Concurrency to System Design, pp. 78–87.

[54] M. Mukund, From global specifications to distributed implementations, in: Synthesis and Control of Discrete Event Systems, Springer, 2002, pp. 19–35.

[55] W. Zielonka, Notes on finite asynchronous automata., Informatique Théorique et Applications 21(2) (1987) 99–135.

[56] Z. Milosevic, S. Sadiq, M. Orlowska, Towards a methodology for deriving contract-compliant business processes, in: Business Process Management, volume 4102 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2006, pp. 395–400.

[57] W. Sadiq, S. Sadiq, K. Schulz, Model driven distribution of collaborative business processes, in: Services Computing, 2006. SCC '06. IEEE International Conference on, pp. 281 –284.

[58] W. Fdhila, U. Yildiz, C. Godart, A flexible approach for automatic process decentralization using dependency tables, International Conference on Web Services (2009).

[59] G. Dong, R. Hull, B. Kumar, J. Su, G. Zhou, A framework for optimizing distributed workflow executions, in: Revised Papers from the 7th International Workshop on Database Programming Languages: Research Issues in Structured and Semistructured Database Programming, DBPL '99, Springer-Verlag, London, UK, 2000, pp. 152–167.

[60] D. Fahland, Towards analyzing declarative workflows, in: J. Koehler, M. Pistore, A. P. Sheth, P. Traverso, M. Wirsing (Eds.), Autonomous and Adaptive Web Services, volume 07061 of *Dagstuhl Seminar Proceedings*, Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2007, p. 6.

[61] M. Montali, Specification and Verification of Declarative Open Interaction Models: A Logic-Based Approach, volume 56 of *Lecture Notes in Business Information Processing*, Springer, 2010.

[62] P. Bellini, R. Mattolini, P. Nesi, Temporal logics for real-time system specification, ACM Comput. Surv. 32 (2000) 12–42.

[63] P. M. Merlin, A study of the recoverability of computing systems., Ph.D. thesis, University of California, Irvine, 1974. AAI7511026.

[64] B. Berthomieu, M. Diaz, Modeling and verification of time dependent systems using time petri nets, Software Engineering, IEEE Transactions on 17 (1991) 259 –273.

[65] L. Popova-Zeugmann, On time petri nets, Elektronische Informationsverarbeitung und Kybernetik 27 (1991) 227–244.

[66] L. Popova-Zeugmann, D. Schlatter, Analyzing paths in time petri nets, Fundam. Inform. 37 (1999) 311–327.

[67] F. Cassez, O. H. Roux, Structural translation from time petri nets to timed automata, Electr. Notes Theor. Comput. Sci. 128 (2005) 145–160.

[68] S. Ling, H. Schmidt, Time petri nets for workflow modelling and analysis, in: Systems, Man, and Cybernetics, 2000 IEEE International Conference on, volume 4, pp. 3039 –3044 vol.4.

[69] C. Ramchandani, Analysis of asynchronous concurrent systems by timed petri nets, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1974.

[70] W. M. Zuberek, Timed petri nets and preliminary performance evaluation, in: Proceedings of the 7th annual symposium on Computer Architecture, ISCA '80, ACM, New York, NY, USA, 1980, pp. 88–96.

[71] B. Walter, Timed petri-nets for modelling and alalyzing protocols with real-time characteristics, in: Protocol Specification, Testing, and Verification, pp. 149–159.

[72] H.-M. Hanisch, Analysis of place/transition nets with timed arcs and its application to batch process control, in: M. Ajmone Marsan (Ed.), Application and Theory of Petri Nets 1993, volume 691 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 1993, pp. 282–299.

## A. Proofs to propositions in Sec. 5

**Lemma A.1.** $e \bullet\!\!\rightarrow_{|\delta} = \{e' \mid e \bullet\!\!\rightarrow e' \wedge e' \in (\delta_{\mathsf{E}} \cup \rightarrow\!\!\diamond\, \delta_{\mathsf{E}})\}$

***Proof*** *According to def 5.1-vi, the response relation in projected graph is*
$\bullet\!\!\rightarrow_{|\delta} = \bullet\!\!\rightarrow \cap ((\bullet\!\!\rightarrow \rightarrow\!\!\diamond\, \delta_{\mathsf{E}}) \times (\rightarrow\!\!\diamond\, \delta_{\mathsf{E}})) \cup ((\bullet\!\!\rightarrow\, \delta_{\mathsf{E}}) \times \delta_{\mathsf{E}}))$.
*Informally it contains relations which can cause a response on an event which is either included in the set of events in the project parameter* $(\delta_{\mathsf{E}})$ *or in a set of events which are milestones to events in project parameter* $(\rightarrow\!\!\diamond\, \delta_{\mathsf{E}})$.
$\bullet\!\!\rightarrow_{|\delta} = \{(e'', e') \mid e'' \bullet\!\!\rightarrow e' \wedge e' \in (\delta_{\mathsf{E}} \cup \rightarrow\!\!\diamond\, \delta_{\mathsf{E}})\}$ *and hence*
$e \bullet\!\!\rightarrow_{|\delta} = \{e' \mid e \bullet\!\!\rightarrow e' \wedge e' \in (\delta_{\mathsf{E}} \cup \rightarrow\!\!\diamond\, \delta_{\mathsf{E}})\}$

**Lemma A.2.** $e{\rightarrow}{+}_{|\delta}{=}e{\rightarrow}{+} \cap(\delta_{\mathsf{E}} \cup {\rightarrow}{\bullet}\ \delta_{\mathsf{E}} \cup {\rightarrow}{\diamond}\ \delta_{\mathsf{E}})$

***Proof*** *According to def 5.1-ix, the include relation in projected graph is*
${\rightarrow}{+}_{|\delta}{=}{\rightarrow}{+} \cap \Big( \big( ({\rightarrow}{+}\ \delta_{\mathsf{E}}) \times \delta_{\mathsf{E}} \big) \cup \big( ({\rightarrow}{+}{\rightarrow}{\bullet}\ \delta_{\mathsf{E}}) \times ({\rightarrow}{\bullet}\ \delta_{\mathsf{E}}) \big) \cup \big( ({\rightarrow}{+}{\rightarrow}{\diamond}\ \delta_{\mathsf{E}}) \times ({\rightarrow}{\diamond}\ \delta_{\mathsf{E}}) \big) \Big)$
${\rightarrow}{+}_{|\delta} = \{(e'', e') \mid e'' {\rightarrow}{+}\ e' \wedge e' \in (\delta_{\mathsf{E}} \cup {\rightarrow}{\bullet}\ \delta_{\mathsf{E}} \cup {\rightarrow}{\diamond}\ \delta_{\mathsf{E}})\}$
$e{\rightarrow}{+}_{|\delta} = \{e' \mid e {\rightarrow}{+}\ e' \wedge e' \in (\delta_{\mathsf{E}} \cup {\rightarrow}{\bullet}\ \delta_{\mathsf{E}} \cup {\rightarrow}{\diamond}\ \delta_{\mathsf{E}})\}$
$e{\rightarrow}{+}_{|\delta}{=}e{\rightarrow}{+} \cap(\delta_{\mathsf{E}} \cup {\rightarrow}{\bullet}\ \delta_{\mathsf{E}} \cup {\rightarrow}{\diamond}\ \delta_{\mathsf{E}})$

**Lemma A.3.** $e{\rightarrow}{\%}_{|\delta}{=}e{\rightarrow}{\%} \cap(\delta_{\mathsf{E}} \cup {\rightarrow}{\bullet}\ \delta_{\mathsf{E}} \cup {\rightarrow}{\diamond}\ \delta_{\mathsf{E}})$

***Proof*** *According to def 5.1-x, the exclude relation in projected graph is*
${\rightarrow}{\%}_{|\delta}{=}{\rightarrow}{\%} \cap \Big( \big( ({\rightarrow}{\%}\ \delta_{\mathsf{E}}) \times \delta_{\mathsf{E}} \big) \cup \big( ({\rightarrow}{\%}{\rightarrow}{\bullet}\ \delta_{\mathsf{E}}) \times ({\rightarrow}{\bullet}\ \delta_{\mathsf{E}}) \big) \cup \big( ({\rightarrow}{\%}{\rightarrow}{\diamond}\ \delta_{\mathsf{E}}) \times ({\rightarrow}{\diamond}\ \delta_{\mathsf{E}}) \big) \Big)$
${\rightarrow}{\%}_{|\delta} = \{(e'', e') \mid e'' {\rightarrow}{\%}\ e' \wedge e' \in (\delta_{\mathsf{E}} \cup {\rightarrow}{\bullet}\ \delta_{\mathsf{E}} \cup {\rightarrow}{\diamond}\ \delta_{\mathsf{E}})\}$
$e{\rightarrow}{\%}_{|\delta} = \{e' \mid e {\rightarrow}{\%}\ e' \wedge e' \in (\delta_{\mathsf{E}} \cup {\rightarrow}{\bullet}\ \delta_{\mathsf{E}} \cup {\rightarrow}{\diamond}\ \delta_{\mathsf{E}})\}$
$e{\rightarrow}{\%}_{|\delta}{=}e{\rightarrow}{\%} \cap(\delta_{\mathsf{E}} \cup {\rightarrow}{\bullet}\ \delta_{\mathsf{E}} \cup {\rightarrow}{\diamond}\ \delta_{\mathsf{E}})$

**Proposition A.1.** *Let* $G = (\mathsf{E}, \mathsf{M}, {\rightarrow}{\bullet}, {\bullet}{\rightarrow}, {\rightarrow}{\diamond}, {\rightarrow}{+}, {\rightarrow}{\%}, \mathsf{L}, l)$ *be a DCR Graph and* $G_{|\delta}$ *its projection with respect to a projection parameter* $\delta = (\delta_{\mathsf{E}}, \delta_{\mathsf{L}})$*. Then, for* $e \in \delta_{\mathsf{E}}$ *and* $a \in \delta_{\mathsf{L}}$ *it holds that* $\mathsf{M} \vdash_G e \wedge \mathsf{M} \oplus_G e = \mathsf{M}' \wedge \mathsf{M}'_{|\delta} = \mathsf{M}''$ *if and only if* $\mathsf{M}_{|\delta} \vdash_{G_{|\delta}} e \wedge \mathsf{M}_{|\delta} \oplus_{G_{|\delta}} e = \mathsf{M}''$.

***Proof*** *In order to prove the proposition, we have to show that the proposition in both directions.*

*(G→P)* *for* $e \in \delta_{\mathsf{E}}$ *and* $a \in \delta_{\mathsf{L}}$. $\mathsf{M} \vdash_G e \wedge \mathsf{M} \oplus_G e = \mathsf{M}' \wedge \mathsf{M}'_{|\delta} = \mathsf{M}'' \implies \mathsf{M}_{|\delta} \vdash_{G_{|\delta}} e \wedge \mathsf{M}_{|\delta} \oplus_{G_{|\delta}} e = \mathsf{M}''$.
*We will split the proof into 2 steps:*

*(A)* $\mathsf{M} \vdash_G e \implies \mathsf{M}_{|\delta} \vdash_{G_{|\delta}} e$
*From def 3.1, we have* $\mathsf{M} \vdash_G e \implies e \in \mathsf{In} \wedge (\mathsf{In} \cap {\rightarrow}{\bullet}\, e) \subseteq \mathsf{Ex}$ *and* $(\mathsf{In} \cap {\rightarrow}{\diamond} e) \subseteq \mathsf{E} \backslash \mathsf{Re}$.
*In order to prove that* $\mathsf{M}_{|\delta} \vdash_{G_{|\delta}} e$*, we have to show that* $e \in \mathsf{In}_{|\delta} \wedge (\mathsf{In}_{|\delta} \cap {\rightarrow}{\bullet}_{|\delta}\, e) \subseteq \mathsf{Ex}_{|\delta} \wedge (\mathsf{In}_{|\delta} \cap {\rightarrow}{\diamond}_{|\delta}\, e) \subseteq \mathsf{E}_{|\delta} \backslash \mathsf{Re}_{|\delta}$*. We will prove each part individually as follows,*

*(i) To prove:* $e \in \mathsf{In}_{|\delta}$.
*From def 5.1-iiic we have,*
$\mathsf{In}_{|\delta} = \mathsf{In} \cap (\delta_{\mathsf{E}} \cup \rightarrow\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\diamond\ \delta_{\mathsf{E}})$ *therefore*
$e \in \mathsf{In} \wedge e \in \delta_{\mathsf{E}} \implies e \in \mathsf{In}_{|\delta}$.

*(ii) To prove:* $(\mathsf{In}_{|\delta} \cap \rightarrow\bullet_{|\delta}\, e) \subseteq \mathsf{Ex}_{|\delta}$.
$\forall e' \in (\mathsf{In}_{|\delta} \cap \rightarrow\bullet_{|\delta}\, e)$,
*(a)* $e' \in \mathsf{In}_{|\delta} \implies e' \in \mathsf{In}$
*(b)* $e' \in \rightarrow\bullet_{|\delta}\, e \implies e' \in \rightarrow\bullet\, e$ *from def 5.1-iv*
*Using above 2 statements and from* $\mathsf{M} \vdash_G e$
$\forall e'.e' \in (\mathsf{In}_{|\delta} \cap \rightarrow\bullet_{|\delta}\, e) \implies e' \in (\mathsf{In} \cap \rightarrow\bullet\, e) \implies e' \in \mathsf{Ex}$,
*Further,* $(\mathsf{In}_{|\delta} \cap \rightarrow\bullet_{|\delta}\, e) \subseteq \mathsf{Ex}_{|\delta}$, *from def 5.1-iiia*

*(iii) To prove:* $(\mathsf{In}_{|\delta} \cap \rightarrow\diamond_{|\delta}\, e) \subseteq \mathsf{E}_{|\delta} \backslash \mathsf{Re}_{|\delta}$
$\forall e' \in (\mathsf{In}_{|\delta} \cap \rightarrow\diamond_{|\delta}\, e)$
*(a)* $e' \in \mathsf{In}_{|\delta} \implies e' \in \mathsf{In}$
*(b)* $e' \in \rightarrow\diamond_{|\delta}\, e \implies e' \in \rightarrow\diamond\, e$, *from def 5.1-viii*
*Using above 2 statements and from* $\mathsf{M} \vdash_G e$
$e' \in (\mathsf{In}_{|\delta} \cap \rightarrow\diamond_{|\delta}\, e) \implies e' \in (\mathsf{In} \cap \rightarrow\diamond\, e) \implies e' \in \mathsf{E} \backslash \mathsf{Re} \implies e' \notin \mathsf{Re}$,
*According to def 5.1 iiib, we have* $\mathsf{Re}_{|\delta} = \mathsf{Re} \cap (\delta_{\mathsf{E}} \cup \rightarrow\diamond\ \delta_{\mathsf{E}})$. *and so* $e' \notin \mathsf{Re} \implies e' \notin \mathsf{Re}_{|\delta}$.
*Further,* $e' \in \mathsf{E}_{|\delta} \wedge e' \notin \mathsf{Re}_{|\delta} \implies e' \in \mathsf{E}_{|\delta} \setminus \mathsf{Re}_{|\delta}$.
*Hence we can conclude that* $(\mathsf{In}_{|\delta} \cap \rightarrow\diamond_{|\delta}\, e) \subseteq \mathsf{E}_{|\delta} \backslash \mathsf{Re}_{|\delta}$

*From (G→P)-A-i, (G→P)-A-ii and (G→P)-A-iii, we have proved that* $e \in \mathsf{In}_{|\delta} \wedge (\mathsf{In}_{|\delta} \cap \rightarrow\bullet_{|\delta}\, e) \subseteq \mathsf{Ex}_{|\delta} \wedge (\mathsf{In}_{|\delta} \cap \rightarrow\diamond_{|\delta}\, e) \subseteq \mathsf{E}_{|\delta} \backslash \mathsf{Re}_{|\delta}$ *is valid. Therefore we can conclude that* $\mathsf{M} \vdash_G e \implies \mathsf{M}_{|\delta} \vdash_{G_{|\delta}} e$.

*(B) To prove:* $\mathsf{M} \oplus_G e = \mathsf{M}' \wedge \mathsf{M}'_{|\delta} = \mathsf{M}'' \implies \mathsf{M}_{|\delta} \oplus_{G_{|\delta}} e = \mathsf{M}''$.
*We have* $\mathsf{M} \oplus_G e = \mathsf{M}'$ *where* $\mathsf{M} = (\mathsf{Ex}, \mathsf{Re}, \mathsf{In})$ *and* $\mathsf{M}' = (\mathsf{Ex}', \mathsf{Re}', \mathsf{In}')$ *and from the def 3.1, we can infer* $\mathsf{Ex}' = \mathsf{Ex} \cup \{e\}, \mathsf{Re}' = (\mathsf{Re} \setminus \{e\}) \cup e\bullet\rightarrow$, *and* $\mathsf{In}' = (\mathsf{In} \cup e \rightarrow +) \setminus e \rightarrow \%$.
*In projected graph, we have* $\mathsf{M}_{|\delta} = (\mathsf{Ex}_{|\delta}, \mathsf{Re}_{|\delta}, \mathsf{In}_{|\delta})$, $\mathsf{M}'' = (\mathsf{Ex}''_{|\delta}, \mathsf{Re}''_{|\delta}, \mathsf{In}''_{|\delta})$ *and from above result we know that* $\mathsf{M}_{|\delta} \vdash_{G_{|\delta}} e$. *Hence we can infer that* $\mathsf{Ex}''_{|\delta} = \mathsf{Ex}_{|\delta} \cup \{e\}, \mathsf{Re}''_{|\delta} = (\mathsf{Re}_{|\delta} \setminus \{e\}) \cup e\bullet\rightarrow_{|\delta}$, *and* $\mathsf{In}''_{|\delta} = (\mathsf{In}_{|\delta} \cup e \rightarrow +_{|\delta}) \setminus e \rightarrow \%_{|\delta}$.
*We have to prove that* $\mathsf{M}'_{|\delta} = \mathsf{M}''$. *In order to prove this equivalence,*

*we will show that* $\mathsf{Ex}'_{|\delta} = \mathsf{Ex}''_{|\delta}$, $\mathsf{Re}'_{|\delta} = \mathsf{Re}''_{|\delta}$ *and* $\mathsf{In}'_{|\delta} = \mathsf{In}''_{|\delta}$ *individually as follows,*

*(i) To prove:* $\mathsf{Ex}'_{|\delta} = \mathsf{Ex}''_{|\delta}$.
$\mathsf{Ex}'_{|\delta} = (\mathsf{Ex} \cup \{e\}) \cap \mathsf{E}_{|\delta}$ *from def 5.1-iiia*
$= (\mathsf{Ex} \cap \mathsf{E}_{|\delta}) \cup (\{e\} \cap \mathsf{E}_{|\delta})$ *distributive law of sets*
$= \mathsf{Ex}_{|\delta} \cup \{e\}$ *according to def 5.1-iiia and* $e \in \delta_{\mathsf{E}} \subseteq \mathsf{E}_{|\delta}$.
$= \mathsf{Ex}''_{|\delta}$.
*Hence we can conclude that* $\mathsf{Ex}'_{|\delta} = \mathsf{Ex}''_{|\delta}$.

*(ii) To prove:* $\mathsf{Re}'_{|\delta} = \mathsf{Re}''_{|\delta}$.
$\mathsf{Re}'_{|\delta} = \big((\mathsf{Re} \setminus \{e\}) \cup e \bullet\!\!\rightarrow\big) \cap (\delta_{\mathsf{E}} \cup \rightarrow\!\!\diamond\ \delta_{\mathsf{E}})$ *from def 5.1-iiib*
$= \big((\mathsf{Re} \setminus \{e\}) \cap (\delta_{\mathsf{E}} \cup \rightarrow\!\!\diamond\ \delta_{\mathsf{E}})\big) \cup \big(\ e \bullet\!\!\rightarrow \cap (\delta_{\mathsf{E}} \cup \rightarrow\!\!\diamond\ \delta_{\mathsf{E}})\big)$ *distributive law*
$= \big(\mathsf{Re} \cap (\delta_{\mathsf{E}} \cup \rightarrow\!\!\diamond\ \delta_{\mathsf{E}}) \setminus (\{e\} \cap (\delta_{\mathsf{E}} \cup \rightarrow\!\!\diamond\ \delta_{\mathsf{E}})\big) \cup \big(\ e \bullet\!\!\rightarrow \cap (\delta_{\mathsf{E}} \cup \rightarrow\!\!\diamond\ \delta_{\mathsf{E}})\big)$ *set intersection distributes over set difference*
$= (\mathsf{Re}_{|\delta} \setminus \{e\}) \cup \big(\ e \bullet\!\!\rightarrow \cap (\delta_{\mathsf{E}} \cup \rightarrow\!\!\diamond\ \delta_{\mathsf{E}})\big)$
$= (\mathsf{Re}_{|\delta} \setminus \{e\}) \cup \{e' \mid e \bullet\!\!\rightarrow e' \wedge e' \in (\delta_{\mathsf{E}} \cup \rightarrow\!\!\diamond\ \delta_{\mathsf{E}})\}$
$= (\mathsf{Re}_{|\delta} \setminus \{e\}) \cup e \bullet\!\!\rightarrow_{|\delta}$ *using lemma A.1*
$= \mathsf{Re}''_{|\delta}$
*Hence we can conclude that* $\mathsf{Re}'_{|\delta} = \mathsf{Re}''_{|\delta}$.

*(iii) To prove:* $\mathsf{In}'_{|\delta} = \mathsf{In}''_{|\delta}$
$\mathsf{In}'_{|\delta} = \mathsf{In}' \cap (\delta_{\mathsf{E}} \cup \rightarrow\!\!\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\!\!\diamond\ \delta_{\mathsf{E}})$, *from def 5.1-iiic*
*But we know that* $\mathsf{In}' = (\mathsf{In} \cup e \rightarrow\!\!+) \setminus e \rightarrow\!\!\%$
$\mathsf{In}'_{|\delta} = \big((\mathsf{In} \cup e \rightarrow\!\!+) \setminus e \rightarrow\!\!\%\ \big) \cap (\delta_{\mathsf{E}} \cup \rightarrow\!\!\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\!\!\diamond\ \delta_{\mathsf{E}})$
$\mathsf{In}'_{|\delta} = \big((\mathsf{In} \cup e \rightarrow\!\!+) \cap (\delta_{\mathsf{E}} \cup \rightarrow\!\!\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\!\!\diamond\ \delta_{\mathsf{E}})\big) \setminus \big(\ e \rightarrow\!\!\% \cap (\delta_{\mathsf{E}} \cup \rightarrow\!\!\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\!\!\diamond\ \delta_{\mathsf{E}})\big)$ *set intersection distributes over set difference*
$\mathsf{In}'_{|\delta} = \big((\mathsf{In} \cap (\delta_{\mathsf{E}} \cup \rightarrow\!\!\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\!\!\diamond\ \delta_{\mathsf{E}})\big) \cup \big(\ e \rightarrow\!\!+ \cap (\delta_{\mathsf{E}} \cup \rightarrow\!\!\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\!\!\diamond\ \delta_{\mathsf{E}})\big) \setminus \big(\ e \rightarrow\!\!\% \cap (\delta_{\mathsf{E}} \cup \rightarrow\!\!\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\!\!\diamond\ \delta_{\mathsf{E}})\big)$ *distributive law*
$\mathsf{In}'_{|\delta} = \Big(\big(\mathsf{In} \cap (\delta_{\mathsf{E}} \cup \rightarrow\!\!\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\!\!\diamond\ \delta_{\mathsf{E}})\big) \cup e \rightarrow\!\!+_{|\delta}\ \Big) \setminus e \rightarrow\!\!\%_{|\delta}$ *using lemmas A.2 and A.3*
*But we know that the marking in projected graph before executing event* $e$ *is* $\mathsf{In}_{|\delta} = \mathsf{In} \cap (\delta_{\mathsf{E}} \cup \rightarrow\!\!\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\!\!\diamond\ \delta_{\mathsf{E}})$. *Using this fact, we can rewrite the above statement as follows,*
$\mathsf{In}'_{|\delta} = (\mathsf{In}_{|\delta} \cup e \rightarrow\!\!+_{|\delta}) \setminus e \rightarrow\!\!\%_{|\delta}$
$\mathsf{In}'_{|\delta} = \mathsf{In}''_{|\delta}$

*Hence we can conclude that* $\mathsf{In}'_{|\delta} = \mathsf{In}''_{|\delta}$.

*From (G→P)-B-i, (G→P)-B-ii and (G→P)-B-iii, we have proved that* $\mathsf{Ex}'_{|\delta} = \mathsf{Ex}''_{|\delta}$, $\mathsf{Re}'_{|\delta} = \mathsf{Re}''_{|\delta}$ *and* $\mathsf{In}'_{|\delta} = \mathsf{In}''_{|\delta}$. *and there by we can conclude that* $\mathsf{M}'_{|\delta} = \mathsf{M}''$.

*Since we have proved both parts: ( (G→P)-A and (G→P)-B ), the proposition* $\mathsf{M} \oplus_G e = \mathsf{M}' \wedge \mathsf{M}'_{|\delta} = \mathsf{M}'' \implies \mathsf{M}_{|\delta} \oplus_{G_{|\delta}} e = \mathsf{M}''$ *holds.*

*(P→G) for* $e \in \delta_{\mathsf{E}}$ *and* $a \in \delta_{\mathsf{L}}$. $\mathsf{M}_{|\delta} \vdash_{G_{|\delta}} e \wedge \mathsf{M}_{|\delta} \oplus_{G_{|\delta}} e = \mathsf{M}'' \implies \mathsf{M} \vdash_G e \wedge \mathsf{M} \oplus_G e = \mathsf{M}' \wedge \mathsf{M}'_{|\delta} = \mathsf{M}''$
*Again, we will split the proof into 2 parts.*

*(A)* $\mathsf{M}_{|\delta} \vdash_{G_{|\delta}} e \implies \mathsf{M} \vdash_G e$
*From def 3.1, we have* $\mathsf{M}_{|\delta} \vdash_{G_{|\delta}} e \implies e \in \mathsf{In}_{|\delta} \wedge (\mathsf{In}_{|\delta} \cap \rightarrow\bullet_{|\delta} e) \subseteq \mathsf{Ex}_{|\delta} \wedge (\mathsf{In}_{|\delta} \cap \rightarrow\diamond_{|\delta} e) \subseteq \mathsf{E}_{|\delta} \backslash \mathsf{Re}_{|\delta}$
*In order to prove that* $\mathsf{M} \vdash_G e$, *we have to show that* $e \in \mathsf{In} \wedge (\mathsf{In} \cap \rightarrow\bullet e) \subseteq \mathsf{Ex}$ *and* $(\mathsf{In} \cap \rightarrow\diamond e) \subseteq \mathsf{E} \backslash \mathsf{Re}$

*(i) To prove:* $e \in \mathsf{In}$
*From def 5.1-iiic we have:* $\mathsf{In}_{|\delta} = \mathsf{In} \cap (\delta_{\mathsf{E}} \cup \rightarrow\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\diamond\ \delta_{\mathsf{E}})$
$e \in \mathsf{In}_{|\delta} \wedge \left(\mathsf{In}_{|\delta} = \mathsf{In} \cap (\delta_{\mathsf{E}} \cup \rightarrow\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\diamond\ \delta_{\mathsf{E}})\right) \implies e \in \mathsf{In}$.

*(ii) To prove:* $(\mathsf{In} \cap \rightarrow\bullet e) \subseteq \mathsf{Ex}$
*From def 5.1-iv, we have* $\rightarrow\bullet_{|\delta} = \rightarrow\bullet \cap ((\rightarrow\bullet\ \delta_{\mathsf{E}}) \times \delta_{\mathsf{E}})$
$\forall e'.e' \in \rightarrow\bullet_{|\delta} e \implies (e', e) \in \rightarrow\bullet_{|\delta} \implies (e', e) \in \rightarrow\bullet \implies e' \in \rightarrow\bullet e$ *and therefore* $\rightarrow\bullet_{|\delta} e = \rightarrow\bullet e$.
$\forall e'.e' \in (\mathsf{In}_{|\delta} \cap \rightarrow\bullet_{|\delta} e) \implies (e' \in \mathsf{In}_{|\delta}) \cap (e' \in \rightarrow\bullet_{|\delta} e) \implies (e' \in \mathsf{In}) \cap (e' \in \rightarrow\bullet e) \implies e' \in (\mathsf{In} \cap \rightarrow\bullet e)$, *and hence* $(\mathsf{In}_{|\delta} \cap \rightarrow\bullet_{|\delta} e) = (\mathsf{In} \cap \rightarrow\bullet e)$.
$(\mathsf{In}_{|\delta} \cap \rightarrow\bullet_{|\delta} e) \subseteq \mathsf{Ex}_{|\delta} \implies (\mathsf{In} \cap \rightarrow\bullet e) \subseteq \mathsf{Ex}_{|\delta}$.
*according to def 5.1-iiia :* $\mathsf{Ex}_{|\delta} = \mathsf{Ex} \cap \mathsf{E}_{|\delta}$.
*Hence* $(\mathsf{In} \cap \rightarrow\bullet e) \subseteq \mathsf{Ex}_{|\delta} \implies (\mathsf{In} \cap \rightarrow\bullet e) \subseteq \mathsf{Ex}$

*(iii) To prove:* $(\mathsf{In} \cap \rightarrow\diamond e) \subseteq \mathsf{E} \backslash \mathsf{Re}$
*From def 5.1-viii, we have* $\rightarrow\diamond_{|\delta} = \rightarrow\diamond \cap ((\rightarrow\diamond\ \delta_{\mathsf{E}}) \times \delta_{\mathsf{E}})$,
$\forall e'.e' \in \rightarrow\diamond_{|\delta} e \implies (e', e) \in \rightarrow\diamond_{|\delta} \implies (e', e) \in \rightarrow\diamond \implies e' \in \rightarrow\diamond e$ *and therefore* $\rightarrow\diamond_{|\delta} e = \rightarrow\diamond e$.
$\forall e'.e' \in (\mathsf{In}_{|\delta} \cap \rightarrow\diamond_{|\delta} e) \implies (e' \in \mathsf{In}_{|\delta}) \cap (e' \in \rightarrow\diamond_{|\delta} e) \implies (e' \in \mathsf{In}) \cap (e' \in \rightarrow\diamond e) \implies e' \in (\mathsf{In} \cap \rightarrow\diamond e)$, *and hence* $(\mathsf{In}_{|\delta} \cap \rightarrow\diamond_{|\delta} e) = (\mathsf{In} \cap \rightarrow\diamond e)$.

$(\mathsf{In}_{|\delta} \cap \rightarrow\diamond_{|\delta} e) \subseteq \mathsf{E}_{|\delta} \backslash \mathsf{Re}_{|\delta} \implies (\mathsf{In} \cap \rightarrow\diamond e) \subseteq \mathsf{E}_{|\delta} \backslash \mathsf{Re}_{|\delta} \implies \forall e' \in (\mathsf{In} \cap \rightarrow\diamond e).e' \notin \mathsf{Re}_{|\delta}$.
*according to def 5.1-iiib :* $\mathsf{Re}_{|\delta} = \mathsf{Re} \cap (\delta_{\mathsf{E}} \cup \rightarrow\diamond \delta_{\mathsf{E}})$,
$\forall e' \in (\mathsf{In} \cap \rightarrow\diamond e).e' \notin \mathsf{Re}_{|\delta} \implies e' \notin (\mathsf{Re} \cap (\delta_{\mathsf{E}} \cup \rightarrow\diamond \delta_{\mathsf{E}}))$.
*Further, as* $e' \rightarrow\diamond e$, *we know that* $e' \in (\delta_{\mathsf{E}} \cup \rightarrow\diamond \delta_{\mathsf{E}})$. *The only way* $e' \notin (\mathsf{Re} \cap (\delta_{\mathsf{E}} \cup \rightarrow\diamond \delta_{\mathsf{E}}))$ *becomes true is when* $e' \notin \mathsf{Re}$.
*Hence* $(\mathsf{In}_{|\delta} \cap \rightarrow\diamond_{|\delta} e) \subseteq \mathsf{E}_{|\delta} \backslash \mathsf{Re}_{|\delta} \implies (\mathsf{In} \cap \rightarrow\diamond e) \subseteq \mathsf{E} \backslash \mathsf{Re}$.

*Form (P→G)-A-(i), (P→G)-A-(ii) and (P→G)-A-(iii), we can conclude that* $\mathsf{M}_{|\delta} \vdash_{G_{|\delta}} e \implies \mathsf{M} \vdash_G e$ .

*(B)* $\mathsf{M}_{|\delta} \oplus_{G_{|\delta}} e = \mathsf{M}'' \implies \mathsf{M} \oplus_G e = \mathsf{M}' \wedge \mathsf{M}'_{|\delta} = \mathsf{M}''$
*We have* $\mathsf{M}_{|\delta} \oplus_{G_{|\delta}} e = \mathsf{M}''$ *in the local graph where* $\mathsf{M}_{|\delta} = (\mathsf{Ex}_{|\delta}, \mathsf{Re}_{|\delta}, \mathsf{In}_{|\delta})$, $\mathsf{M}'' = (\mathsf{Ex}''_{|\delta}, \mathsf{Re}''_{|\delta}, \mathsf{In}''_{|\delta})$ *and from the def 3.1, we can infer* $\mathsf{Ex}''_{|\delta} = \mathsf{Ex}_{|\delta} \cup \{e\}, \mathsf{Re}''_{|\delta} = (\mathsf{Re}_{|\delta} \backslash \{e\}) \cup e \bullet\!\!\rightarrow_{|\delta}$, *and* $\mathsf{In}''_{|\delta} = (\mathsf{In}_{|\delta} \cup e \rightarrow +_{|\delta}) \backslash e \rightarrow \%_{|\delta}$.
*In main graph, we know* $\mathsf{M} \vdash_G e$ *where* $\mathsf{M} = (\mathsf{Ex}, \mathsf{Re}, \mathsf{In})$ *and hence we can workout the new marking as* $\mathsf{M} \oplus_G e = \mathsf{M}'$ *where* $\mathsf{M}' = (\mathsf{Ex}', \mathsf{Re}', \mathsf{In}')$ *with* $\mathsf{Ex}' = \mathsf{Ex} \cup \{e\}, \mathsf{Re}' = (\mathsf{Re} \setminus \{e\}) \cup e \bullet\!\!\rightarrow$, *and* $\mathsf{In}' = (\mathsf{In} \cup e \rightarrow +) \backslash e \rightarrow \%$.
*We have to prove that* $\mathsf{M}'' = \mathsf{M}'_{|\delta}$.

*(i) To prove:* $\mathsf{Ex}''_{|\delta} = \mathsf{Ex}'_{|\delta}$
*Let us start with* $\mathsf{Ex}''_{|\delta}$
$\mathsf{Ex}''_{|\delta} = \mathsf{Ex}_{|\delta} \cup \{e\}$
$= (\mathsf{Ex} \cap \mathsf{E}_{|\delta}) \cup \{e\}$ *from def 5.1-iiia*
$= (\mathsf{Ex} \cup \{e\}) \cap (\mathsf{E}_{|\delta} \cup \{e\})$
$= \mathsf{Ex}' \cap \mathsf{E}_{|\delta}$
$= \mathsf{Ex}'_{|\delta}$
*Hence we can conclude that* $\mathsf{Ex}''_{|\delta} = \mathsf{Ex}'_{|\delta}$.

*(ii) To prove:* $\mathsf{Re}''_{|\delta} = \mathsf{Re}'_{|\delta}$
*Let us start with* $\mathsf{Re}''_{|\delta}$
$\mathsf{Re}''_{|\delta} = (\mathsf{Re}_{|\delta} \setminus \{e\}) \cup e \bullet\!\!\rightarrow_{|\delta}$
$= \big((\mathsf{Re} \cap (\delta_{\mathsf{E}} \cup \rightarrow\diamond \delta_{\mathsf{E}})) \setminus \{e\}\big) \cup e \bullet\!\!\rightarrow_{|\delta}$ *from def 5.1-iiib*
$= \big((\mathsf{Re} \backslash \{e\}) \cap (\delta_{\mathsf{E}} \cup \rightarrow\diamond \delta_{\mathsf{E}})\big) \cup e \bullet\!\!\rightarrow_{|\delta}$ *(set relative complements)*
$= \big((\mathsf{Re} \setminus \{e\}) \cap (\delta_{\mathsf{E}} \cup \rightarrow\diamond \delta_{\mathsf{E}})\big) \cup (e \bullet\!\!\rightarrow \cap (\delta_{\mathsf{E}} \cup \rightarrow\diamond \delta_{\mathsf{E}}))$ *using lemma A.1*
$= \big((\mathsf{Re} \setminus \{e\}) \cup e \bullet\!\!\rightarrow\big) \cap (\delta_{\mathsf{E}} \cup \rightarrow\diamond \delta_{\mathsf{E}})$
$= \mathsf{Re}' \cap (\delta_{\mathsf{E}} \cup \rightarrow\diamond \delta_{\mathsf{E}})$

$= \mathsf{Re}'_{|\delta}$ *according to def 5.1-iiib.*
*Hence we can conclude that* $\mathsf{Re}''_{|\delta} = \mathsf{Re}'_{|\delta}$.

*(iii) To prove:* $\mathsf{In}''_{|\delta} = \mathsf{In}'_{|\delta}$

*Let us starts with* $\mathsf{In}''_{|\delta}$ *and show that it will be equal to the projection over included set from global graph* $(\mathsf{In}'_{|\delta})$.
$\mathsf{In}''_{|\delta} = (\mathsf{In}_{|\delta} \cup e \rightarrow +_{|\delta}) \setminus e \rightarrow \%_{|\delta}$
$\mathsf{In}''_{|\delta} = \big((\mathsf{In} \cap (\delta_{\mathsf{E}} \cup \rightarrow\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\diamond\ \delta_{\mathsf{E}})) \cup e \rightarrow +_{|\delta}\ \big) \setminus e \rightarrow \%_{|\delta}$, *from def 5.1-iiic.*
$\mathsf{In}''_{|\delta} = \big((\mathsf{In} \cap (\delta_{\mathsf{E}} \cup \rightarrow\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\diamond\ \delta_{\mathsf{E}})) \cup (e \rightarrow + \cap (\delta_{\mathsf{E}} \cup \rightarrow\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\diamond\ \delta_{\mathsf{E}}))\big) \setminus (e \rightarrow \% \cap (\delta_{\mathsf{E}} \cup \rightarrow\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\diamond\ \delta_{\mathsf{E}}))$ *using lemmas A.2 and A.3*
$\mathsf{In}''_{|\delta} = \big((\mathsf{In} \cup e \rightarrow +) \cap (\delta_{\mathsf{E}} \cup \rightarrow\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\diamond\ \delta_{\mathsf{E}})\big) \setminus (e \rightarrow \% \cap (\delta_{\mathsf{E}} \cup \rightarrow\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\diamond\ \delta_{\mathsf{E}}))$.
$\mathsf{In}''_{|\delta} = \big((\mathsf{In} \cup e \rightarrow +) \setminus e \rightarrow \%\ \big) \cap (\delta_{\mathsf{E}} \cup \rightarrow\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\diamond\ \delta_{\mathsf{E}})$.
$\mathsf{In}''_{|\delta} = (\mathsf{In}') \cap (\delta_{\mathsf{E}} \cup \rightarrow\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\diamond\ \delta_{\mathsf{E}})$.
$\mathsf{In}''_{|\delta} = \mathsf{In}'_{|\delta}$.
*Hence we can conclude that* $\mathsf{In}''_{|\delta} = \mathsf{In}'_{|\delta}$.

*From (P→G)-B-i, (P→G)-B-ii and (P→G)-B-iii, we have proved that* $\mathsf{Ex}''_{|\delta} = \mathsf{Ex}'_{|\delta}$, $\mathsf{Re}''_{|\delta} = \mathsf{Re}'_{|\delta}$ *and* $\mathsf{In}''_{|\delta} = \mathsf{In}'_{|\delta}$ *and there by we can conclude that* $\mathsf{M}'' = \mathsf{M}'_{|\delta}$.

*Since we have proved both parts: ( (P→G)-A and (P→G)-B ), the proposition for* $e \in \delta_{\mathsf{E}}$ *and* $a \in \delta_{\mathsf{L}}$. $\mathsf{M}_{|\delta} \vdash_{G_{|\delta}} e \wedge \mathsf{M}_{|\delta} \oplus_{G_{|\delta}} e = \mathsf{M}'' \implies \mathsf{M} \vdash_G e \wedge \mathsf{M} \oplus_G e = \mathsf{M}' \wedge \mathsf{M}'_{|\delta} = \mathsf{M}''$ *holds.*

*Finally, we have proved the proposition in both ways* $\big($*(G→P) and (P→G)*$\big)$*, therefore the proposition: for* $e \in \delta_{\mathsf{E}}$ *and* $a \in \delta_{\mathsf{L}}$ *it holds that* $\mathsf{M} \vdash_G e \wedge \mathsf{M} \oplus_G e = \mathsf{M}' \wedge \mathsf{M}'_{|\delta} = \mathsf{M}''$ *if and only if* $\mathsf{M}_{|\delta} \vdash_{G_{|\delta}} e \wedge \mathsf{M}_{|\delta} \oplus_{G_{|\delta}} e = \mathsf{M}''$ *holds.*

**Proposition A.2.** *Let* $G = (\mathsf{E}, \mathsf{M}, \rightarrow\bullet, \bullet\rightarrow, \rightarrow\diamond, \rightarrow+, \rightarrow\%, \mathsf{L}, l)$ *be a DCR Graph and* $G_{|\delta}$ *its projection with respect to a projection parameter* $\delta = (\delta_{\mathsf{E}}, \delta_{\mathsf{L}})$. *Then, for* $e \notin \mathsf{E}_{|\delta}$ *it holds that* $\mathsf{M} \vdash_G e \wedge \mathsf{M} \oplus_G e = \mathsf{M}'$ *implies* $\mathsf{M}_{|\delta} = \mathsf{M}'_{|\delta}$.

***Proof*** *According to projection definition 5.1,* $e \notin \mathsf{E}_{|\delta} \implies e \notin G_{|\delta}$, *therefore there will not be any change in the marking. Hence* $\mathsf{M}_{|\delta} = \mathsf{M}'_{|\delta}$.

**Proposition A.3.** *Let* $G = (\mathsf{E}, \mathsf{M}, \rightarrow\bullet, \bullet\rightarrow, \rightarrow\diamond, \rightarrow+, \rightarrow\%, \mathsf{L}, l)$ *be a DCR Graph and* $G_{|\delta}$ *its projection with respect to a projection parameter* $\delta = (\delta_{\mathsf{E}}, \delta_{\mathsf{L}})$*. Then for* $e \in \mathsf{E}_{|\delta}$ *(and* $a \notin \delta_{\mathsf{L}}$*) it holds that* $\mathsf{M} \vdash_G e \wedge \mathsf{M} \oplus_G e = \mathsf{M}'$ *implies* $\mathsf{M}_{|\delta} \oplus_{G_{|\delta}} e = \mathsf{M}'_{|\delta}$*.*

***Proof*** *The proof for this proposition is more or less similar to proof in the part (P→G)-(B) of proposition A.1 with minor changes.*

*We have* $\mathsf{M} \oplus_G e = \mathsf{M}'$ *where* $\mathsf{M} = (\mathsf{Ex}, \mathsf{Re}, \mathsf{In})$ *and* $\mathsf{M}' = (\mathsf{Ex}', \mathsf{Re}', \mathsf{In}')$ *and from the def 3.1, we can infer* $\mathsf{Ex}' = \mathsf{Ex} \cup \{e\}, \mathsf{Re}' = (\mathsf{Re} \setminus \{e\}) \cup e\bullet\rightarrow$*, and* $\mathsf{In}' = (\mathsf{In} \cup e\rightarrow+) \setminus e\rightarrow\%$*.*
*In projected graph, we have marking projected from global graph, according to def 5.1 as* $\mathsf{M}_{|\delta} = (\mathsf{Ex}_{|\delta}, \mathsf{Re}_{|\delta}, \mathsf{In}_{|\delta})$*. The result of executing event* $e$ *in projected graph will be a marking, let us say* $\mathsf{M}''_{|\delta} = \mathsf{M}_{|\delta} \oplus_{G_{|\delta}} e$*, then we have to prove that* $\mathsf{M}''_{|\delta} = \mathsf{M}'_{|\delta}$*.*
*Let us say that* $\mathsf{M}''_{|\delta} = (\mathsf{Ex}''_{|\delta}, \mathsf{Re}''_{|\delta}, \mathsf{In}''_{|\delta})$*, and since in the projected graph we have* $\mathsf{M}''_{|\delta} = \mathsf{M}_{|\delta} \oplus_{G_{|\delta}} e$*, we can infer from the def 3.1,* $\mathsf{Ex}''_{|\delta} = \mathsf{Ex}_{|\delta} \cup \{e\}, \mathsf{Re}''_{|\delta} = (\mathsf{Re}_{|\delta} \setminus \{e\}) \cup e\bullet\rightarrow_{|\delta}$*, and* $\mathsf{In}''_{|\delta} = (\mathsf{In}_{|\delta} \cup e\rightarrow+_{|\delta}) \setminus e\rightarrow\%_{|\delta}$*.*
*In order to prove this equivalence of* $\mathsf{M}''_{|\delta} = \mathsf{M}'_{|\delta}$*, we will show that* $\mathsf{Ex}''_{|\delta} = \mathsf{Ex}'_{|\delta}$*,* $\mathsf{Re}''_{|\delta} = \mathsf{Re}'_{|\delta}$ *and* $\mathsf{In}''_{|\delta} = \mathsf{In}'_{|\delta}$ *individually as follows,*

*(i) To prove:* $\mathsf{Ex}''_{|\delta} = \mathsf{Ex}'_{|\delta}$
*Let us start with* $\mathsf{Ex}''_{|\delta}$
$\mathsf{Ex}''_{|\delta} = \mathsf{Ex}_{|\delta} \cup \{e\}$
$= (\mathsf{Ex} \cap \mathsf{E}_{|\delta}) \cup \{e\}$ *from def 5.1-iiia*
$= (\mathsf{Ex} \cup \{e\}) \cap (\mathsf{E}_{|\delta} \cup \{e\})$
$= \mathsf{Ex}' \cap \mathsf{E}_{|\delta}$
$= \mathsf{Ex}'_{|\delta}$
*Hence we can conclude that* $\mathsf{Ex}''_{|\delta} = \mathsf{Ex}'_{|\delta}$*.*

*(ii) To prove:* $\mathsf{Re}''_{|\delta} = \mathsf{Re}'_{|\delta}$
*Let us start with* $\mathsf{Re}''_{|\delta}$
$\mathsf{Re}''_{|\delta} = (\mathsf{Re}_{|\delta} \setminus \{e\}) \cup e\bullet\rightarrow_{|\delta}$
$= \big((\mathsf{Re} \cap (\delta_{\mathsf{E}} \cup \rightarrow\diamond\ \delta_{\mathsf{E}})) \setminus \{e\}\big) \cup e\bullet\rightarrow_{|\delta}$ *from def 5.1-iiib*
$= \big((\mathsf{Re} \setminus \{e\}) \cap (\delta_{\mathsf{E}} \cup \rightarrow\diamond\ \delta_{\mathsf{E}})\big) \cup e\bullet\rightarrow_{|\delta}$ *(set relative complements)*
$= \big((\mathsf{Re} \setminus \{e\}) \cap (\delta_{\mathsf{E}} \cup \rightarrow\diamond\ \delta_{\mathsf{E}})\big) \cup (e\bullet\rightarrow \cap (\delta_{\mathsf{E}} \cup \rightarrow\diamond\ \delta_{\mathsf{E}}))$ *using lemma A.1*
$= \big((\mathsf{Re} \setminus \{e\}) \cup e\bullet\rightarrow\big) \cap (\delta_{\mathsf{E}} \cup \rightarrow\diamond\ \delta_{\mathsf{E}})$
$= \mathsf{Re}' \cap (\delta_{\mathsf{E}} \cup \rightarrow\diamond\ \delta_{\mathsf{E}})$

$= \mathsf{Re}'_{|\delta}$ *according to def 5.1-iiib.*
*Hence we can conclude that* $\mathsf{Re}''_{|\delta} = \mathsf{Re}'_{|\delta}$.

*(iii) To prove:* $\mathsf{In}''_{|\delta} = \mathsf{In}'_{|\delta}$
*Let us starts with* $\mathsf{In}''_{|\delta}$ *and show that it will be equal to the projection over included set from global graph* $(\mathsf{In}'_{|\delta})$.
$\mathsf{In}''_{|\delta} = (\mathsf{In}_{|\delta} \cup e \rightarrow +_{|\delta}) \setminus e \rightarrow \%_{|\delta}$
$\mathsf{In}''_{|\delta} = \big((\mathsf{In} \cap (\delta_{\mathsf{E}} \cup \rightarrow\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\diamond\ \delta_{\mathsf{E}})) \cup e \rightarrow +_{|\delta}\ \big) \setminus e \rightarrow \%_{|\delta}$ *from def 5.1-iiic.*
$\mathsf{In}''_{|\delta} = \big((\mathsf{In} \cap (\delta_{\mathsf{E}} \cup \rightarrow\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\diamond\ \delta_{\mathsf{E}})) \cup (e \rightarrow + \cap (\delta_{\mathsf{E}} \cup \rightarrow\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\diamond\ \delta_{\mathsf{E}}))\big) \setminus (e \rightarrow \% \cap (\delta_{\mathsf{E}} \cup \rightarrow\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\diamond\ \delta_{\mathsf{E}}))$ *using lemmas A.2 and A.3*
$\mathsf{In}''_{|\delta} = \big((\mathsf{In} \cup e \rightarrow +) \cap (\delta_{\mathsf{E}} \cup \rightarrow\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\diamond\ \delta_{\mathsf{E}})\big) \setminus (e \rightarrow \% \cap (\delta_{\mathsf{E}} \cup \rightarrow\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\diamond\ \delta_{\mathsf{E}}))$.
$\mathsf{In}''_{|\delta} = \big((\mathsf{In} \cup e \rightarrow +) \setminus e \rightarrow \%\ \big) \cap (\delta_{\mathsf{E}} \cup \rightarrow\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\diamond\ \delta_{\mathsf{E}})$.
$\mathsf{In}''_{|\delta} = (\mathsf{In}') \cap (\delta_{\mathsf{E}} \cup \rightarrow\bullet\ \delta_{\mathsf{E}} \cup \rightarrow\diamond\ \delta_{\mathsf{E}})$.
$\mathsf{In}''_{|\delta} = \mathsf{In}'_{|\delta}$.
*Hence we can conclude that* $\mathsf{In}''_{|\delta} = \mathsf{In}'_{|\delta}$.

*From (i), (ii) and (iii), we have proved that* $\mathsf{Ex}''_{|\delta} = \mathsf{Ex}'_{|\delta}$, $\mathsf{Re}''_{|\delta} = \mathsf{Re}'_{|\delta}$ *and* $\mathsf{In}''_{|\delta} = \mathsf{In}'_{|\delta}$ *and there by we can conclude that* $\mathsf{M}'' = \mathsf{M}'_{|\delta}$.
*Therefore the proposition: for* $e \in \mathsf{E}_{|\delta}$ *(and* $a \notin \delta_{\mathsf{L}}$*) it holds that* $\mathsf{M} \vdash_G e \wedge \mathsf{M} \oplus_G e = \mathsf{M}'$ *implies* $\mathsf{M}_{|\delta} \oplus_{G_{|\delta}} e = \mathsf{M}'_{|\delta}$ *is proved.*

**Proof of Proposition 5.1.1.** Let $G = (\mathsf{E}, \mathsf{M_t}, \rightarrow\bullet, t_c, \bullet\rightarrow, t_r, \rightarrow\diamond, \rightarrow+, \rightarrow\%, \mathsf{L}, \mathsf{l})$ be a Timed DCR Graph and $G_{|\delta}$ its projection with respect to a projection parameter $\delta = (\delta_{\mathsf{E}}, \delta_{\mathsf{L}})$. Then, for $e \in \delta_{\mathsf{E}}$ and $a \in \delta_{\mathsf{L}}$ it holds that $\mathsf{M_t} \vdash_G e \wedge \mathsf{M_t} \oplus_G e = \mathsf{M_t}' \wedge \mathsf{M_t}'_{|\delta} = \mathsf{M_t}''$ if and only if $\mathsf{M_t}_{|\delta} \vdash_{G_{|\delta}} e \wedge \mathsf{M_t}_{|\delta} \oplus_{G_{|\delta}} e = \mathsf{M_t}''$.
In order to prove the proposition, we have to show that the proposition in both directions.

(G→P) for $e \in \delta_{\mathsf{E}}$ and $a \in \delta_{\mathsf{L}}$. $\mathsf{M_t} \vdash_G e \wedge \mathsf{M_t} \oplus_G e = \mathsf{M_t}' \wedge \mathsf{M_t}'_{|\delta} = \mathsf{M_t}'' \implies \mathsf{M_t}_{|\delta} \vdash_{G_{|\delta}} e \wedge \mathsf{M_t}_{|\delta} \oplus_{G_{|\delta}} e = \mathsf{M_t}''$.
We will split the proof into 2 steps:

(A) $\mathsf{M_t} \vdash_G e \implies \mathsf{M_t}_{|\delta} \vdash_{G_{|\delta}} e$
From def 3.3, we have
$\mathsf{M_t} \vdash_G e \implies \mathsf{M} \vdash_{G'} e \wedge \forall e' \in \mathsf{In}.e' \overset{k}{\rightarrow}\bullet\ e \implies k \leq t_{ex}(e')$.
$\mathsf{M} \vdash_{G'} e$ has been proved in the (G→P)(A) part of the proof of proposition A.1, therefore we have to show that $\forall e' \in \mathsf{In}.e' \overset{k}{\rightarrow}\bullet\ e \implies k \leq$

$t_{ex}(e')$ implies that $\forall e' \in \mathsf{In}_{|\delta}.e' \overset{k}{\rightarrow}\bullet_{|\delta}\ e \implies k \leq t_{ex|\delta}(e')$, where $e' \overset{k}{\rightarrow}\bullet_{|\delta}\ e =_{def} e' \rightarrow\bullet_{|\delta}\ e$ and $t_{c|\delta}(e, e') = k$.

$\forall e' \in \mathsf{In}.e' \overset{k}{\rightarrow}\bullet\ e \implies k \leq t_{ex}(e')$
$= \forall e' \in \mathsf{In}.e' \overset{k}{\rightarrow}\bullet_{|\delta}\ e \implies k \leq t_{ex}(e')$, from def 5.1-iv
$= \forall e' \in \mathsf{In}_{|\delta}.e' \overset{k}{\rightarrow}\bullet_{|\delta}\ e \implies k \leq t_{ex}(e')$, from def 5.1-iiic
$= \forall e' \in \mathsf{In}_{|\delta}.e' \overset{k}{\rightarrow}\bullet_{|\delta}\ e \implies k \leq (t_{ex}(e')$ if $e' \in \mathsf{Ex})$, by definition of $t_{ex}$
$= \forall e' \in \mathsf{In}_{|\delta}.e' \overset{k}{\rightarrow}\bullet_{|\delta}\ e \implies k \leq (t_{ex}(e')$ if $e' \in \mathsf{Ex} \cap \delta_{\mathsf{E}})$, since $\mathsf{M}_{|\delta} \vdash_{G'} e \wedge e' \in \mathsf{In}_{|\delta}$, it preserves the $t_{ex}$ behavior
$= \forall e' \in \mathsf{In}_{|\delta}.e' \overset{k}{\rightarrow}\bullet_{|\delta}\ e \implies k \leq (t_{ex}(e')$ if $e' \in \mathsf{Ex}_{|\delta})$, from def 5.1-iiia
$= \forall e' \in \mathsf{In}_{|\delta}.e' \overset{k}{\rightarrow}\bullet_{|\delta}\ e \implies k \leq t_{ex|\delta}(e')$ from def 5.1-iiid.
Hence proved.

(B) To prove: $\mathsf{M_t} \oplus_G e = \mathsf{M_t}' \wedge \mathsf{M_t}'_{|\delta} = \mathsf{M_t}'' \implies \mathsf{M_{t|\delta}} \oplus_{G_{|\delta}} e = \mathsf{M_t}''$
From def. 3.3 we have $\mathsf{M_t} \oplus_G e =_{def} \big((\mathsf{Ex}, \mathsf{Re}, \mathsf{In}) \oplus_G e, t'_{ex}, t'_{re}\big)$, where

(i) $t'_{ex}(e') = \begin{cases} 0 & \text{if } e' = e \\ t_{ex}(e') & \text{otherwise} \end{cases}$

(ii) $t'_{re}(e') = \begin{cases} k & \text{if } e \overset{k}{\bullet\rightarrow} e' \\ t_{re}(e') & \text{otherwise} \end{cases}$

We proved the $(\mathsf{Ex}, \mathsf{Re}, \mathsf{In}) \oplus_G e$ part in the (G→P)(B) part of the proof of proposition A.1. We then need to show that $({t_{ex}}')_{|\delta} \implies {t_{ex}}''$ and that $({t_{re}}')_{|\delta} \implies {t_{re}}''$

$({t_{ex}}')_{|\delta}(e')$
$= t'_{ex}(e')$ if $e' \in \mathsf{Ex}'_{|\delta}$, according to def 5.1-iiid
$= \left(\begin{cases} 0 & \text{if } e' = e \\ t_{ex}(e') & \text{otherwise} \end{cases}\right)$ if $e' \in (\mathsf{Ex} \cup \{e\}) \cap \mathsf{E}_{|\delta}$, according to def 3.3, and def 5.1-iiia
$= \begin{cases} 0 & \text{if } e' = e \wedge e' \in (\mathsf{Ex} \cup \{e\}) \cap \mathsf{E}_{|\delta} \\ t_{ex}(e') & \text{if } e' \in (\mathsf{Ex} \cup \{e\}) \cap \mathsf{E}_{|\delta} \end{cases}$, by moving the projection inward

But we know that if $e' = e$, trivially $e \in \mathsf{Ex} \cup \{e\}$ and $e \in \mathsf{E}_{|\delta}$, by def 5.1-i. Moreover, the second branch will be used only if $e' \neq e$, therefore

$= \begin{cases} 0 & \text{if } e' = e \\ t_{ex}(e') & \text{if } e' \in (\mathsf{Ex} \cap \mathsf{E}_{|\delta}) \end{cases}$

$= {t_{ex}}''(e')$, according to def 5.1-iiia, and def 3.3

$({t_{re}}')_{|\delta}(e')$

$= t'_{re}(e')$ if $e' \in \mathsf{Re}'_{|\delta}$, according to def 5.1-iiie

$= \left( \begin{cases} k & \text{if } e \overset{k}{\bullet\!\!\rightarrow} e' \\ t_{re}(e') & \text{otherwise} \end{cases} \right)$ if $e' \in \mathsf{Re}'_{|\delta}$, according to def 3.3

$= \begin{cases} k & \text{if } e \overset{k}{\bullet\!\!\rightarrow}_{|\delta} e' \wedge e' \in \mathsf{Re}'_{|\delta} \\ t_{re}(e') & \text{if } e' \in \mathsf{Re}'_{|\delta} \end{cases}$, according to def 5.1-vi and moving the projection inward

$= \begin{cases} k & \text{if } e \overset{k}{\bullet\!\!\rightarrow}_{|\delta} e' \wedge e' \in ((\mathsf{Re} \setminus \{e\}) \cup e\bullet\!\!\rightarrow) \cap (\delta_{\mathsf{E}} \cup \rightarrow\!\diamond\, \delta_{\mathsf{E}}) \\ t_{re}(e') & \text{if } e' \in ((\mathsf{Re} \setminus \{e\}) \cup e\bullet\!\!\rightarrow) \cap (\delta_{\mathsf{E}} \cup \rightarrow\!\diamond\, \delta_{\mathsf{E}}) \end{cases}$, according to def 5.1-iiib and def 3.1

But we know that if $e \overset{k}{\bullet\!\!\rightarrow}_{|\delta} e'$, trivially $e' \in e\bullet\!\!\rightarrow$, and also that the second branch will be used only if $(e, e') \notin \bullet\!\!\rightarrow_{|\delta}$, therefore

$= \begin{cases} k & \text{if } e \overset{k}{\bullet\!\!\rightarrow}_{|\delta} e' \\ t_{re}(e') & \text{if } e' \in \mathsf{Re} \cap (\delta_{\mathsf{E}} \cup \rightarrow\!\diamond\, \delta_{\mathsf{E}}) \end{cases}$

$= \begin{cases} k & \text{if } e \overset{k}{\bullet\!\!\rightarrow}_{|\delta} e' \\ t_{re|\delta}(e') & \text{otherwise} \end{cases}$, by def 5.1-vi, and def 5.1-iiie

$= {t_{re}}''(e')$, according to def 3.3

Since we have proved both parts: ( (G→P)-A and (G→P)-B ), the proposition $\mathsf{M_t} \oplus_G e = \mathsf{M_t}' \wedge \mathsf{M}'_{\mathsf{t}|\delta} = \mathsf{M_t}'' \implies \mathsf{M}_{\mathsf{t}|\delta} \oplus_{G_{|\delta}} e = \mathsf{M_t}''$ holds.

(P→G) for $e \in \delta_{\mathsf{E}}$ and $a \in \delta_{\mathsf{L}}$. $\mathsf{M}_{\mathsf{t}|\delta} \vdash_{G_{|\delta}} e \wedge \mathsf{M}_{\mathsf{t}|\delta} \oplus_{G_{|\delta}} e = \mathsf{M_t}'' \implies \mathsf{M_t} \vdash_G e \wedge \mathsf{M_t} \oplus_G e = \mathsf{M_t}' \wedge \mathsf{M}'_{\mathsf{t}|\delta} = \mathsf{M_t}''$
Again, we will split the proof into 2 parts.

(A) $\mathsf{M}_{\mathsf{t}|\delta} \vdash_{G_{|\delta}} e \implies \mathsf{M_t} \vdash_G e$

From def 3.3, we have
$\mathsf{M_t} \vdash_G e \implies \mathsf{M} \vdash_{G'} e \wedge \forall e' \in \mathsf{In}.e' \overset{k}{\rightarrow}\bullet\ e \implies k \leq t_{ex}(e')$.
$\mathsf{M}_{|\delta} \vdash_{G'} e$ has been proved in the (P→G)(A) part of the proof of proposition A.1, therefore we have to show that $\forall e' \in \mathsf{In}_{|\delta}.e' \overset{k}{\rightarrow}\bullet_{|\delta}\ e \implies k \leq t_{ex|\delta}(e')$ implies that $\forall e' \in \mathsf{In}.e' \overset{k}{\rightarrow}\bullet\ e \implies k \leq t_{ex}(e')$, where $e' \overset{k}{\rightarrow}\bullet\ e =_{def} e' \rightarrow\bullet\ e$ and $t_c(e, e') = k$.

$\forall e' \in \mathsf{In}_{|\delta}.e' \overset{k}{\rightarrow}\bullet_{|\delta}\ e \implies k \leq t_{ex|\delta}(e')$
$= \forall e' \in \mathsf{In}.e' \overset{k}{\rightarrow}\bullet_{|\delta}\ e \implies k \leq t_{ex|\delta}(e')$ from A.1-i
$= \forall e' \in \mathsf{In}.e' \overset{k}{\rightarrow}\bullet\ e \implies k \leq t_{ex|\delta}(e')$ from A.1-ii
$= \forall e' \in \mathsf{In}.e' \overset{k}{\rightarrow}\bullet\ e \implies k \leq (t_{ex}(e')$ if $e' \in \mathsf{Ex}_{|\delta})$, according to def 5.1-iiid
$= \forall e' \in \mathsf{In}.e' \overset{k}{\rightarrow}\bullet\ e \implies k \leq t_{ex}(e')$ since $\mathsf{M} \vdash_{G'} e \wedge e' \in \rightarrow\bullet\ e \wedge e' \in \mathsf{In} \implies e' \in \mathsf{Ex}_{|\delta}$.
Hence proved.

(B) $\mathsf{M_{t|\delta}} \oplus_{G_{|\delta}} e = \mathsf{M_t}'' \implies \mathsf{M_t} \oplus_G e = \mathsf{M_t}' \wedge \mathsf{M}'_{\mathsf{t}|\delta} = \mathsf{M_t}''$
From def. 3.3 we have
$\mathsf{M_t} \oplus_G e =_{def} \big((\mathsf{Ex}, \mathsf{Re}, \mathsf{In}) \oplus_G e, t'_{ex}, t'_{re}\big)$, where

(i) $t'_{ex}(e') = \begin{cases} 0 & \text{if } e' = e \\ t_{ex}(e') & \text{otherwise} \end{cases}$

(ii) $t'_{re}(e') = \begin{cases} k & \text{if } e \overset{k}{\bullet\rightarrow} e' \\ t_{re}(e') & \text{otherwise} \end{cases}$

We proved the $(\mathsf{Ex}, \mathsf{Re}, \mathsf{In}) \oplus_G e$ part in the (P→G)(B) part of the proof of proposition A.1. We then need to show that ${t_{ex}}'' \implies ({t_{ex}}')_{|\delta}$ and that ${t_{re}}'' \implies ({t_{re}}')_{|\delta}$

${t_{ex}}''(e')$
$= \begin{cases} 0 & \text{if } e' = e \\ t_{ex|\delta}(e') & \text{otherwise} \end{cases}$, according to def 3.3 and considering that if $e' \neq e$, the underlying $t_{ex}$ behaves exactly like $t_{ex|\delta}$
$= \begin{cases} 0 & \text{if } e' = e \\ t_{ex}(e') & \text{if } e' \in \mathsf{Ex}_{|\delta} \end{cases}$, according to def 5.1-iiid

$= \left( \begin{cases} 0 & \text{if } e' = e \\ t_{ex}(e') & \text{otherwise} \end{cases} \right)$ if $e' \in (\mathsf{Ex}_{|\delta}) \cup \{e\}$, since the overall function will continue to behave the same
$= t'_{ex}(e')$ if $e' \in (\mathsf{Ex} \cap \delta_{\mathsf{E}}) \cup \{e\}$, according to def 3.1, and def 5.1-iiia
$= t'_{ex}(e')$ if $e' \in (\mathsf{Ex} \cup \{e\}) \cap (\delta_{\mathsf{E}} \cup \{e\})$
$= t'_{ex}(e')$ if $e' \in \mathsf{Ex}'_{|\delta}$, according to def 3.1, and def 5.1-iiia
$= (t_{ex}{}')_{|\delta}(e')$, according to def 3.3

$t_{re}{}''(e')$
$= \begin{cases} k & \text{if } e \overset{k}{\bullet\!\!\rightarrow}_{|\delta} e' \\ t_{re|\delta}(e') & \text{otherwise} \end{cases}$, according to def 3.3 and considering that if $(e, e') \notin \bullet\!\!\rightarrow_{|\delta}$, the underlying $t_{re}$ behaves exactly like $t_{re|\delta}$
$= \begin{cases} k & \text{if } e \overset{k}{\bullet\!\!\rightarrow}_{|\delta} e' \\ t_{re}(e') & \text{if } e' \in \mathsf{Re}_{|\delta} \end{cases}$, according to def 5.1-iiie
$\left( \begin{cases} k & \text{if } e \overset{k}{\bullet\!\!\rightarrow} e' \\ t_{re}(e') & \text{otherwise} \end{cases} \right)$ if $e' \in (\mathsf{Re}_{|\delta} \setminus \{e\}) \cup e \bullet\!\!\rightarrow_{|\delta}$, according to def 5.1-vi, and since the overall function will continue to behave the same
$= t_{re}{}'(e')$ if $e' \in (\mathsf{Re}_{|\delta} \setminus \{e\}) \cup e \bullet\!\!\rightarrow_{|\delta}$, according to def 3.3
$= t_{re}{}'(e')$ if $e' \in ((\mathsf{Re} \cap (\delta_{\mathsf{E}} \cup \rightarrow\!\diamond\, \delta_{\mathsf{E}}) \setminus \{e\}) \cup e \bullet\!\!\rightarrow_{|\delta})$, according to def 5.1-iiib
$= t_{re}{}'(e')$ if $e' \in ((\mathsf{Re} \setminus \{e\}) \cup e \bullet\!\!\rightarrow_{|\delta}) \cap (\delta_{\mathsf{E}} \cup \rightarrow\!\diamond\, \delta_{\mathsf{E}})$
$= t_{re}{}'(e')$ if $e' \in \mathsf{Re}'_{|\delta}$, according to def 3.1, and def 5.1-vi
$= (t_{re}{}')_{|\delta}(e')$, according to def 5.1-iiie

Since we have proved both parts: ( (P→G)-A and (P→G)-B ), the proposition for $e \in \delta_{\mathsf{E}}$ and $a \in \delta_{\mathsf{L}}$. $\mathsf{M_t}_{|\delta} \vdash_{G_{|\delta}} e \wedge \mathsf{M_t}_{|\delta} \oplus_{G_{|\delta}} e = \mathsf{M_t}{}''$ $\Longrightarrow$ $\mathsf{M_t} \vdash_G e \wedge \mathsf{M_t} \oplus_G e = \mathsf{M_t}{}' \wedge \mathsf{M_t}'_{|\delta} = \mathsf{M_t}{}''$ holds.

Finally, we have proved the proposition in both ways $\big($(G→P) and (P→G)$\big)$, therefore the proposition: for $e \in \delta_{\mathsf{E}}$ and $a \in \delta_{\mathsf{L}}$ it holds that $\mathsf{M_t} \vdash_G e \wedge \mathsf{M_t} \oplus_G e = \mathsf{M_t}{}' \wedge \mathsf{M_t}'_{|\delta} = \mathsf{M_t}{}''$ if and only if $\mathsf{M_t}_{|\delta} \vdash_{G_{|\delta}} e \wedge \mathsf{M_t}_{|\delta} \oplus_{G_{|\delta}} e = \mathsf{M_t}{}''$ holds.

**Proof of Proposition 5.1.2.** Follows trivially from result of proposition A.2.

**Proof of Proposition 5.1.3.** Considering the result of proposition A.3, since the marking of the underlying DCR Graph is the same, this proposition is proved by the result from the part (P→G)-(B) of proposition 1.