Soundness in Negotiations*

Javier Esparza[1], Denis Kuperberg[2], Anca Muscholl[3], and Igor Walukiewicz[4]

1 **Technical University of Munich**
2 **Technical University of Munich**
3 **Technical University of Munich, IAS & CNRS†**
4 **University of Bordeaux, CNRS, LaBRI**

Abstract

Negotiations are a formalism for describing multiparty distributed cooperation. Alternatively, they can be seen as a model of concurrency with synchronized choice as communication primitive. Well-designed negotiations must be sound, meaning that, whatever its current state, the negotiation can still be completed. In a former paper, Esparza and Desel have shown that deciding soundness of a negotiation is PSPACE-complete, and in PTIME if the negotiation is deterministic. They have also provided an algorithm for an intermediate class of acyclic, non-deterministic negotiations, but left the complexity of the soundness problem open.

In the first part of this paper we study two further analysis problems for sound acyclic deterministic negotiations, called the race and the omission problem, and give polynomial algorithms. We use these results to provide the first polynomial algorithm for some analysis problems of workflow nets with data previously studied by Trcka, van der Aalst, and Sidorova.

In the second part we solve the open question of Esparza and Desel's paper. We show that soundness of acyclic, weakly non-deterministic negotiations is in PTIME, and that checking soundness is already NP-complete for slightly more general classes.

1998 ACM Subject Classification D.1.3 Concurrent Programming, D.3.2 Language Classifications, D.2.2 Design Tools and Techniques, F.2.0 Analysis of Algorithms and Problem Commplexity – General, H.4.1 Office Automation

Keywords and phrases Negotiations, workflows, soundness, verification, concurrency.

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2016.12

1 Introduction

A multiparty atomic negotiation is an event in which several processes (agents) synchronize in order to select one out of a number of possible outcomes. In [3] Esparza and Desel introduced *negotiations*, a model of concurrency with multiparty atomic negotiation as interaction primitive. The model describes a workflow of "atomic" negotiations. After an atomic negotiation concludes with the selection of an outcome, the workflow determines the set of atomic negotiations each agent is ready to engage next.

The negotiation model has been studied in [3, 4, 5], and in [6] the results have been applied to the analysis of industrial business processes modeled as workflow Petri nets, a very successful formal backend for graphical notations like BPMN (Business Process Modeling Notation), EPC (Event-driven Process Chain), or UML Activity Diagrams (see e.g. [15, 14]).

* This work was partially supported by the DFG Project "Negotiations: A Model for Tractable Concurrency"

† On leave from the University of Bordeaux.

27th International Conference on Concurrency Theory (CONCUR 2016).
Editors: Josée Desharnais and Radha Jagadeesan; Article No. 12; pp. 12:1–12:13
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

As shown in [1], deterministic negotiations are very closely related to free-choice workflow nets, a class that is expressive enough to model many business processes (for example, 70% of the almost 2000 workflow nets from the suite of industrial models studied in [16, 7, 6] are free-choice).

The most prominent analysis problem for the negotiation model is *soundness*. Loosely speaking, a negotiation is sound if for every reachable configuration there is an execution leading to proper termination of the negotiation. In [3] it is shown that the soundness problem is PSPACE-complete for non-deterministic negotiations and coNP-complete for acyclic non-deterministic negotiations[1]. For this reason, and in search of a tractable class, [3] introduces the class of *deterministic negotiations*. In deterministic negotiations all agents are deterministic, meaning that they are never ready to engage in more than one atomic negotiation per outcome (in the same way that in a deterministic automaton, for each action the automaton is only ready to move to one state). The main results of [3] are a polynomial time *reduction algorithm* for checking soundness of acyclic deterministic negotiations, and an extension of the algorithm to the more expressive class of acyclic, weakly deterministic[2] negotiations. The runtime of this second algorithm was however left open, as well as the more general question of determining the complexity of checking soundness for other classes of acyclic negotiations. In [4] the polynomial result for acyclic deterministic negotiations is extended to the cyclic case.

While unsound negotiations are clearly faulty, sound negotiations are not automatically correct, they must satisfy other properties. In the first contribution of this paper, we study two other analysis problems for *sound* acyclic deterministic negotiations: the *race problem* and the *omission problem*. The race problem is to determine if there is an execution in which two given atomic negotiations are concurrently enabled. The omission problem asks for given sets of atomic negotiations P and B if there exists a run that visits all elements of P and omits all of B. We show that for sound negotiations the race problem is polynomial, as well as the omission problem for P of bounded size. We then apply these polynomial algorithms to analysis problems for negotiations with global data studied in [13, 11] in the context of workflow Petri nets. In this model atomic negotiations can manipulate global variables, so classical analysis questions are raised, for instance whether every value written into a variable is guaranteed to be read, or whether a variable can be allocated and deallocated by two atomic negotiations taking place in parallel. While the algorithms of [13, 11] are exponential, our solutions for acyclic sound deterministic negotiations take polynomial time.

Our second contribution is the study of the complexity of soundness for classes beyond deterministic negotiations. We propose to analyze this problem through properties of the graph of a negotiation. The first indication of the usefulness of this approach is a short argument giving an NLOGSPACE algorithm for deciding soundness of acyclic deterministic negotiations. Next, we settle the question left open in [3], and prove that the soundness problem can be solved in polynomial time for acyclic, weakly non-deterministic negotiations, a class even more general than the one defined in [3]. We then show that if we leave out one of the two assumptions, acyclicity or weak non-determinism, then the problem becomes coNP-complete[3]. These results set a limit to the class of negotiations with a polynomial soundness problems, but also admit a positive interpretation. Indeed, if all processes are

[1] In [3] the notion of soundness has one more requirement, which makes the soundness problem for acyclic negotiations coNP-hard and in DP.

[2] The class considered [3] was called “weakly deterministic”. In this paper we refer to it as “very weakly non-deterministic”.

[3] We show that coNP-hardness holds even for a very mild relaxation of acyclicity.

allowed to be cyclic and non-deterministic, then the soundness problem is PSPACE-complete, while for the class above it belongs to coNP.

Related formalisms and related work. The connection between negotiations and Petri nets is studied in detail in [1]. Every negotiation can be transformed into an exponentially larger 1-safe workflow Petri net with an isomorphic reachability graph. Every deterministic negotiation is equivalent to a 1-safe workflow free-choice net with a linear blow-up. Conversely, every sound workflow free-choice net can be transformed into a sound deterministic negotiation with a linear blow-up. Recent papers on free-choice workflow Petri nets are [8, 6]. In [8] soundness is characterized in terms of anti-patterns, which can be used to explain why a given workflow net is unsound. Our work provides an anti-pattern characterization for acyclic weakly non-deterministic negotiations, which goes beyond the free-choice case. In [6] a polynomial reduction algorithm for free-choice workflow Petri nets is presented. Our results show that soundness is also polynomial for workflow Petri nets coming from acyclic weakly deterministic negotiations.

As a process-based concurrent model, negotiations can be compared with another well-studied model for distributed computation, namely Zielonka automata [17, 2, 10]. Such an automaton is a parallel composition of finite transition systems with synchronization on common actions. The important point is that a synchronization involves exchange of information between states of agents: the result of the synchronization depends on the states of all the components taking part in it. Zielonka automata have the same expressive power as arbitrary, possibly nondeterministic negotiations. Deterministic negotiations correspond to a subclass that does not seem to have been studied yet, and for which verification becomes considerably easier. For example, the question whether some local state occurs in some execution is PSPACE-complete for "sound" Zielonka automata, while it can be answered in polynomial time for sound deterministic negotiations.

A somewhat similar graphical formalism are message sequence charts/graphs, used to describe asynchronous communication. Questions like non-emptiness of intersection are in general undecidable for this model, even assuming that communication buffers are bounded. Subclasses of message sequence graphs with decidable model-checking problem were proposed, but the complexity is PSPACE-complete [9].

Overview. Section 2 introduces definitions and notations, then Section 3 reconsiders soundness for acyclic, deterministic negotiations. In Section 4 we provide an NLOGSPACE algorithm for the race problem. Section 5 solves the omitting problem, that is used in Section 6 for analyzing properties of workflows described by acyclic, deterministic negotiations, and later in Section 7 to decide soundness for acyclic weakly non-deterministic negotiations in PTIME. Finally, Section 8 establishes the coNP complexity bounds.

2 Negotiations

A *negotiation* $\mathcal{N}$ is a tuple $\langle Proc, N, dom, R, \delta\rangle$, where *Proc* is a finite set of *processes* (or agents) that can participate in negotiations, and N is a finite set of *nodes* (or *atomic negotiations*) where the processes can synchronize. The function $dom : N \to \mathcal{P}(Proc)$ associates to every atomic negotiation $n \in N$ the (non-empty) set $dom(n)$ of processes participating in it. Nodes are denoted as m or n, and processes as p or q; possibly with indices.

The set of possible outcomes of atomic negotiations is denoted R, and we use $a, b, \dots$ to range over its elements. The control flow in a negotiation is determined by a partial transition function $\delta : N \times R \times Proc \stackrel{\cdot}{\longrightarrow} \mathcal{P}(N)$, telling that after the outcome a of an atomic

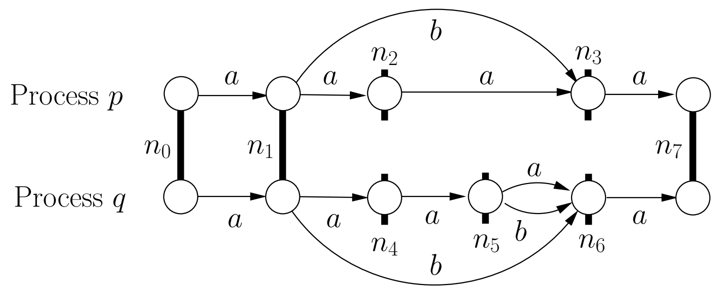

Figure 1 A negotiation. Atomic negotiation n_1 involves processes p, q, and has two possible outcomes a and b. The arrows show next negotiations in which respective processes are willing to engage.

negotiation n, process $p \in dom(n)$ is ready to participate in any of the negotiations from the set $\delta(n, a, p)$. So for every $n' \in \delta(n, a, p)$ we have $p \in dom(n') \cap dom(n)$. Every atomic negotiation $n \in N$ has its set of possible outcomes $out(n)$, and for every n, $a \in out(n)$ and $p \in dom(n)$ the result $\delta(n, a, p)$ has to be defined. So all processes involved in an atomic negotiation should be ready for all its possible outcomes. Observe that atomic negotiations may have one single participant process, and/or have one single outcome.

Negotiations admit a graphical representation. Figure 1 shows a negotiation with $Proc = \{p, q\}$, $N = \{n_0, \dots, n_7\}$ and $R = \{a, b\}$. For example, we have $dom(n_1) = \{p, q\}$, $\delta(n_1, b, p) = \{n_3\}$ and $\delta(n_1, b, q) = \{n_6\}$. More details can be found in [3].

A *configuration* of a negotiation is a function $C : Proc \to \mathcal{P}(N)$ mapping each process p to the set of atomic negotiations in which p is ready to engage. An atomic negotiation n is *enabled* in a configuration C if $n \in C(p)$ for every $p \in dom(n)$, that is, if all processes that participate in n are ready to proceed with it. A configuration is a *deadlock* if no atomic negotiation is enabled in it. If an atomic negotiation n is enabled in C, and a is an outcome of n, then we say that (n, a) *can be executed*, and its execution produces a new configuration C' given by $C'(p) = \delta(n, a, p)$ for $p \in dom(n)$ and $C'(p) = C(p)$ for $p \notin dom(n)$. We denote this by $C \xrightarrow{(n,a)} C'$. For example, in Figure 1 we have $C \xrightarrow{(n_1,a)} C'$ for $C(p) = \{n_1\} = C(q)$ and $C'(p) = \{n_2\}, C(q) = \{n_4\}$.

A *run* of a negotiation $\mathcal{N}$ from a configuration C_1 is a finite or infinite sequence $w = (n_1, a_1)(n_2, a_2)\dots$ such that there are configurations $C_2, C_3, \dots$ with

$$C_1 \xrightarrow{(n_1,a_1)} C_2 \xrightarrow{(n_2,a_2)} C_3 \dots$$

We denote this by $C_1 \xrightarrow{w}$, or $C_1 \xrightarrow{w} C_k$ if the sequence is finite and finishes with C_k. In the latter case we say that C_k is *reachable from* C_1 *on* w. We simply call it *reachable* if w is irrelevant, and write $C_1 \xrightarrow{*} C_k$.

Negotiations come equipped with two distinguished initial and final atomic negotiations n_{init} and n_{fin} in which *all* processes in $Proc$ participate. The *initial and final configurations* C_{init}, C_{fin} are given by $C_{init}(p) = \{n_{init}\}$ and $C_{fin}(p) = \{n_{fin}\}$ for all $p \in Proc$. A run is *successful* if it starts in C_{init} and ends in C_{fin}. We assume that every atomic negotiation (except possibly for n_{fin}) has at least one outcome. In Figure 1, $n_{init} = n_0$ and $n_{fin} = n_7$.

2.1 Main definitions

A negotiation $\mathcal{N}$ is *sound* if every partial run starting at C_{init} can be completed to a successful run. If a negotiation has no infinite runs, then it is sound iff it has no reachable deadlock configuration.

Process p is *deterministic* in a negotiation $\mathcal{N}$ if for every $n \in N$, and $a \in R$, the set of possible next negotiations, $\delta(n,a,p)$, is a singleton or the empty set. A negotiation is *deterministic* if every process $p \in \mathit{Proc}$ is deterministic. The negotiation of Figure 1 is deterministic.

A negotiation is *weakly non-deterministic* if for every $n \in N$ at least one of the processes in $\mathit{dom}(n)$ is deterministic. A negotiation is *very weakly non-deterministic*[4] if for every $n \in N$, $a \in R$, and $p \in \mathit{Proc}$, there is a deterministic process q such that $q \in \mathit{dom}(n')$ for all $n' \in \delta(n,a,p)$.

Examples of weakly non-deterministic negotiations can be found in [3]. In particular, weakly non-deterministic negotiations allow to model deterministic negotiations with global resources (see Section 6). The resource (say, a piece of data) can be modeled as an additional process, which participates in the atomic negotiations that use the resource. The outcome of a negotiation can change the state of the resource (say, from "confidential" to "public"), and at each state the resource may be ready to engage in a different set of atomic negotiations.

The *graph of a negotiation* has atomic negotiations, N, as set of nodes; the edges are $n \xrightarrow{p,a} n'$ if $n' \in \delta(n,a,p)$. Observe that $p \in \mathit{dom}(n) \cap \mathit{dom}(n')$.

A negotiation is *acyclic* if its graph is so. Acyclic negotiations cannot have infinite runs, so as mentioned above, soundness is equivalent to deadlock-freedom. For an acyclic negotiation $\mathcal{N}$ we fix a linear order $\preccurlyeq_{\mathcal{N}}$ on its nodes that is a topological order on the graph of $\mathcal{N}$. This means that if there is an edge from m to n in the graph of $\mathcal{N}$ then $m \preccurlyeq_{\mathcal{N}} n$.

The *restriction* of a negotiation $\mathcal{N}$ to a subset of its processes Proc' is the negotiation $\langle \mathit{Proc}', N', \mathit{dom}', R, \delta' \rangle$ where N' is the set of those $n \in N$ for which $\mathit{dom}(n) \cap \mathit{Proc}' \neq \emptyset$, $\mathit{dom}'(n) = \mathit{dom}(n) \cap \mathit{Proc}'$, and $\delta'(n,r,p) = \delta(n,r,p) \cap N'$. The restriction of $\mathcal{N}$ to deterministic processes is denoted as $\mathcal{N}_D$ throughout the paper.

A negotiation $\mathcal{N}$ is *det-acyclic* if $\mathcal{N}_D$ is acyclic. It follows easily from the definitions that a weakly non-deterministic, det-acyclic negotiation does not have any infinite run.

3 Soundness of acyclic deterministic negotiations

The main objective of this section is to provide some tools that we will use later. We show how some properties of negotiations can be determined by patterns in their graphs. As an example of an application of our techniques we revisit the soundness problem for acyclic, deterministic negotiations. We provide an alternative polynomial-time algorithm that is actually in NLOGSPACE, in contrast with the algorithm of [3] that is based on rewriting.

Fix a negotiation $\mathcal{N}$. A *local path* is a path $n_0 \xrightarrow{p_0,a_0} n_1 \xrightarrow{p_1,a_1} \dots \xrightarrow{p_{k-1},a_{k-1}} n_k$ in the graph of $\mathcal{N}$. The path is *realizable* from some configuration C, if there is a run $C \xrightarrow{w}$ with w of the form $(n_0,a_0)w_1(n_1,a_1)\cdots w_{k-1}(n_{k-1},a_{k-1})$, such that $p_i \notin \mathit{dom}(w_{i+1})$, for all i. Here we use $\mathit{dom}(v)$ to denote the set of all processes involved in some atomic negotiation appearing in sequence v: $\mathit{dom}(v) = \bigcup\{\mathit{dom}(n) : \text{for some } a,\ (n,a) \text{ appears in } v\}$.

For what follows Lemma 1 is particularly useful as it gives a simple criterion when an atomic negotiation is a part of some successful run.

▶ **Lemma 1.** *Let $n_0 \xrightarrow{p_0,a_0} n_1 \xrightarrow{p_1,a_1} \dots \xrightarrow{p_{k-1},a_{k-1}} n_k$ be a local path in the graph of a sound deterministic negotiation $\mathcal{N}$. If C is a reachable configuration of $\mathcal{N}$ and n_0 is enabled in C then the path is realizable from C.*

[4] This class was called *weakly deterministic* in [3].

Proof. Let C be such that $C(p) = n_0$ for every $p \in dom(n_0)$. By induction on i we show that there is a run $C \xrightarrow{*} C_i$ realizing $n_0 \xrightarrow{p_0,a_0} n_1 \xrightarrow{p_1,a_1} \dots \xrightarrow{p_{i-1},a_{i-1}} n_i$ and such that n_i is enabled in C_i.

For $i = 0$, we simply take $C_i = C$. For the induction step we assume the existence of C_i in which n_i is enabled. Let C'_{i+1} be the result of executing (n_i, a_i) from C_i. Observe that $C'_{i+1}(p_i) = n_{i+1}$ (recall that $\mathcal{N}$ is deterministic). Since $\mathcal{N}$ is sound, and C'_{i+1} is reachable, there is a run from C'_{i+1} to C_{fin}. We set then C_{i+1} to be the first configuration on this run when n_{i+1} is enabled. ◀

Lemma 1 says that there is a run containing the atomic negotiation m iff there is a local path from n_{init} to m. If $dom(m) \cap dom(n) \neq \emptyset$ then the lemma also provides an easy test for knowing whether there is a run containing both m, n: it suffices to check the existence of a local path $n_{init} \xrightarrow{*} m \xrightarrow{*} n$, or with m, n interchanged. The next lemma takes care of the opposite situation.

▶ **Lemma 2.** *Let m, n be two atomic negotiations in a sound deterministic negotiation $\mathcal{N}$, and assume that $dom(m) \cap dom(n) = \emptyset$.*

There exists some run of $\mathcal{N}$ containing both m, n iff there is an atomic negotiation m' such that

- *there is a local path from n_{init} to m',*
- *$\delta(m', p, a) = m_0$, $\delta(m', q, a) = n_0$ for some $p, q \in dom(m')$, $a \in out(m')$,*
- *there are two* disjoint *local paths in $\mathcal{N}$, one from m_0 to m, the other from n_0 to n.*

Soundness can be characterized by excluding a special variant of the pattern from the above lemma. Consider two processes $p \neq q$ of an acyclic negotiation $\mathcal{N}$. A *(p, q)-pair* is a pair of *disjoint* local paths of $\mathcal{N}$:

$$m_0 \xrightarrow{p,a_0} \dots \xrightarrow{p,a_{k-1}} m_k \qquad \text{and} \qquad n_0 \xrightarrow{q,b_0} \dots \xrightarrow{q,b_{l-1}} n_l$$

such that $m_k \preccurlyeq_{\mathcal{N}} n_l$ and $q \in dom(m_k)$.

▶ **Lemma 3.** *Let $\mathcal{N}$ be an acyclic deterministic negotiation. Then $\mathcal{N}$ is not sound if and only if there exist an atomic negotiation m' and two processes p, q such that:*

- *there is a local path from n_{init} to m',*
- *$\delta(m', p, a) = m_0$, $\delta(m', q, a) = n_0$ for some $a \in out(m')$,*
- *there is a (p, q)-pair as above.*

▶ **Theorem 4.** *Soundness of acyclic deterministic negotiations is* NLOGSPACE*-complete.*

Proof. Clearly the problem is NLOGSPACE-hard since graph reachability is a special instance of it. The NLOGSPACE algorithm for deciding soundness establishes the existence of the pattern from the previous lemma. Note that the topological order $\preccurlyeq_{\mathcal{N}}$ we use is arbitrary, so we can simply replace the condition $m_k \preccurlyeq_{\mathcal{N}} n_l$ by asking that there is no path from n_l to m_k.

◀

4 Races

For a given pair of atomic negotiations $m, n \in N$ of a deterministic negotiation $\mathcal{N} = \langle Proc, N, dom, R, \delta \rangle$, we want to determine if there is a reachable configuration at which m, n are concurrently enabled. In other words, we are asking whether a race between m and

n is possible. This is a standard question for concurrent systems, that is difficult to answer when working with linearizations. In this section we show a simple linear time algorithm answering the above question for acyclic, sound negotiations. Our algorithm reduces it to graph reachability questions, and can be implemented in logarithmic space. In the long version of our paper we also give a polynomial-time algorithm for possibly cyclic (and sound) negotiations.

We will write $m \parallel n$ when there is a reachable configuration C of $\mathcal{N}$ where both m and n are enabled. Our goal is to decide if $m \parallel n$ holds for given m, n.

We say below that a run $w \in (N \times R)^*$ can be reordered into another run w' if w' can be obtained from w by repeatedly exchanging adjacent $(m,a)(n,b)$ into $(n,b)(m,a)$ whenever $dom(m) \cap dom(n) = \emptyset$.

▶ **Lemma 5.** *Let $\mathcal{N}$ be an acyclic, deterministic, sound negotiation, and let m,n be two atomic negotiations in $\mathcal{N}$. Then $m \parallel n$ iff* every *run w from n_{init} containing both m and n can be reordered into a run w' such that $w' = C_{init} \xrightarrow{*} C \xrightarrow{*} C'$ for some configuration C where both m and n are enabled.*

Proof. It suffices to show the implication from left to right. So assume that there exists some reachable configuration C where both m and n are enabled. In particular, $dom(m) \cap dom(n) = \emptyset$. By way of contradiction, let us suppose that there exists some run containing both m and n, but this run cannot be reordered as claimed. We claim that there must be some local path from m to n in $\mathcal{N}$. To see this, assume the contrary and consider a run of the form $w = w_1(m,a)w_2(n,b)w_3$. The run w defines a partial order (actually a Mazurkiewicz trace) $tr(w)$ with nodes corresponding to positions in w, and edges from (m',c) to (n',d) if $dom(m') \cap dom(n') \neq \emptyset$ and (m',c) precedes (n',d) in w. Since there is no path from m to n in $\mathcal{N}$, nodes (m,a) and (n,b) are unordered in $tr(w)$. So we can choose a topological order w' of $tr(w)$ of the form $w' = w'_1(m,a)(n,b)w'_2$. This shows the claim.

So let π be a path in $\mathcal{N}$ from $m, n_1, \ldots, n_k, n$. Let p be some process such that $n_k \xrightarrow{p,a'} n$ for some outcome a'.

Let us go back to C. Since both m and n are enabled in C, we have a transition $C \xrightarrow{n,b} C_1$, for some $b \in out(n)$. Note that m is still enabled in C_1, since $dom(m) \cap dom(n) = \emptyset$. So we can apply Lemma 1 to C_1 and π (because $\mathcal{N}$ is sound), obtaining a configuration C_2 where $C_2(p) = n$. But since n was executed before C_1, this violates the acyclicity of $\mathcal{N}$. ◀

The next step is to convert the condition from Lemma 5 to a condition on the graph of a negotiation.

▶ **Proposition 6.** *Let $\mathcal{N}$ be an acyclic, deterministic, sound negotiation, and let m,n be two atomic negotiations in $\mathcal{N}$. Then $m \parallel n$ iff there exists a run containing both m,n, and there is neither a local path from m to n nor a local path from n to m.*

Observe that $dom(m) \cap dom(n) = \emptyset$ is a necessary condition for $m \parallel n$. Thus, from Proposition 6 and Lemma 2 we immediately obtain:

▶ **Theorem 7.** *For any acyclic, deterministic, sound negotiation $\mathcal{N}$ we can decide in linear time whether two atomic negotiations m,n of $\mathcal{N}$ satisfy $m \parallel n$. The above problem is* NLOGSPACE-*complete.*

5 Omitting problem

In this section we will be interested in determining the existence of some special successful runs of a deterministic negotiation $\mathcal{N}$. Let $B \subseteq N$ be a set of nodes of a negotiation $\mathcal{N}$. We say that a run $(n_1, a_1)(n_2, a_2)\dots$ *omits* B if it does not contain any nodes from B, that is, $n_i \notin B$ for all i. Let $P \subseteq N \times R$ be a set of positive requirements. We say that a run as above *includes* P *and omits* B if it omits B and contains all the pairs from P.

We are interested in deciding if for a given $\mathcal{N}$ together with P and B there is a successful run of $\mathcal{N}$ including P and omitting B. We will consider only $\mathcal{N}$ that are sound, acyclic, and deterministic.

As a first step we define a *game* $G(\mathcal{N}, B)$ that is intended to produce runs that omit B (see e.g. [12] for an introduction to games):

- the positions of Eve are $N \setminus B$,
- the positions of Adam are $N \times R$,
- from n, Eve can go to any (n, a) with $a \in out(n)$,
- from (n, a), Adam can choose any process $p \in Proc$ and go to $n' = \delta(n, a, p)$,
- the initial position is n_{init},
- Adam wins if the play reaches a node in B, Eve wins if the play reaches n_{fin}.

Observe that since $\mathcal{N}$ is acyclic, the winning condition for Eve is actually a safety condition: every maximal play avoiding B is winning for Eve. So if Eve can win then she wins with a positional strategy. A *deterministic positional strategy for Eve* is a function $\sigma : N \to R$, it indicates that at position n Eve should go to position $(n, \sigma(n))$. Since $G(\mathcal{N}, B)$ is a safety game for Eve, there is a *biggest non-deterministic winning strategy* for Eve, i.e., a strategy of type $\sigma_{max} : N \to \mathcal{P}(R)$. The strategy σ_{max} is obtained by computing the set W_E of all winning positions for Eve in $G(\mathcal{N}, B)$, and then setting for every $n \in N$:

$$\sigma_{max}(n) = \{a \in out(n) : \forall p \in dom(n).\ \delta(n, a, p) \in W_E\}$$

► **Lemma 8.** *If $\mathcal{N}$ has a run omitting B then Eve has a winning strategy in $G(\mathcal{N}, B)$.*

► **Lemma 9.** *Suppose $\mathcal{N}$ is sound. Let $\sigma : N \to R$ be a winning strategy for Eve in $G(\mathcal{N}, B)$. Consider the set S of nodes that are reachable on a play from n_{init} respecting σ. There is a successful run of $\mathcal{N}$ containing precisely the nodes S.*

Proof. Consider an enumeration $n_1, n_2, ..., n_k$ of the nodes in $S \subseteq (N \setminus B)$ according to the topological order $\preccurlyeq_{\mathcal{N}}$. Let $w_i = (n_1, \sigma(n_1)) \dots (n_i, \sigma(n_i))$. By induction on $i \in \{1, \dots, k\}$ we prove that there is a configuration C_i such that $C_{init} \xrightarrow{w_i} C_i$ is a run of $\mathcal{N}$. This will show that w_k is a successful run containing precisely the nodes of S.

For $i = 1$, $n_1 = n_{init}$, in C_{init} all processes are ready to do n_1, so C_1 is the result of performing $(n_1, \sigma(n_1))$.

For the inductive step, we assume that we have a run $C_{init} \xrightarrow{w_i} C_i$, and we want to extend it by $C_i \xrightarrow{(n_{i+1}, \sigma(n_{i+1}))} C_{i+1}$. Consider a play respecting σ and reaching n_{i+1}. The last step in this play is $(n_j, \sigma(n_j)) \to n_{i+1}$, for some $j \leq i$ and n_j in S. This means that $\delta(n_j, \sigma(n_j), p) = n_{i+1}$ for some process p. Since $j \leq i$ and $(n_j, \sigma(n_j))$ occurred in w_i (but not n_{i+1}), we have $C_i(p) = n_{i+1}$. If we show that $C_i(q) = \{n_{i+1}\}$ for all $q \in dom(n_{i+1})$ then we obtain that n_{i+1} is enabled in C_i and we get the required C_{i+1}. Suppose by contradiction that $C_i(q) = \{n_l\}$ for some $l \neq i + 1$. We must have $l > i + 1$, since otherwise n_l already occurred in w_i. By definition of our indexing $n_{i+1} \prec_{\mathcal{N}} n_l$. But then no execution from C_i can bring process q to a state where it is ready to participate in negotiation n_{i+1}, and p will stay forever in n_{i+1}. This contradicts the fact that the negotiation is sound. ◄

► **Corollary 10.** *For a sound negotiation $\mathcal{N}$: Eve wins in $G(\mathcal{N}, B)$ iff $\mathcal{N}$ has a successful run omitting B.*

▶ **Theorem 11.** *Let K be a constant. It can be decided in* PTIME *if for a given deterministic, acyclic, and sound negotiation $\mathcal{N}$ and two sets $B \subseteq N$, and $P \subseteq N \times R$, with the size of P at most K, there is a successful run of $\mathcal{N}$ containing P and omitting B.*

Proof. If for some atomic negotiation m, we have $(m, a) \in P$ and $(m, b) \in P$ for $a \neq b$ then the answer is negative as $\mathcal{N}$ is acyclic. So let us suppose that it is not the case. By Lemmas 8 and 9 our problem is equivalent to determining the existence of a deterministic strategy σ for Eve in the game $G(\mathcal{N}, B)$ such that $\sigma(m) = a$ for all $(m, a) \in P$, and all these (m, a) are reachable on a play respecting σ.

To decide this we calculate σ_{max}, the biggest non-deterministic winning strategy for Eve in $G(\mathcal{N}, B)$. This can be done in PTIME as the size of $G(\mathcal{N}, B)$ is proportional to the size of the negotiation. Strategy σ_{max} defines a graph $G(\sigma_{max})$ whose nodes are atomic negotiations, and edges are (m, a, m') if $(m, a) \in \sigma_{max}$ and $m' = \delta(m, a, p)$ for some process p. The size of this graph is proportional to the size of the negotiation. In this graph we look for a subgraph H such that:

- for every node m in H there is at most one a such that (m, a, m') is an edge of H for some m';
- for every $(m, a) \in P$ there is an edge (m, a, m') in H for some m', and moreover m is reachable from n_{init} in H.

We show that such a graph H exists iff there is a strategy σ with the required properties.

Suppose there is a deterministic winning strategy σ such that $\sigma(m) = a$ for all $(m, a) \in P$, and all these (m, a) are reachable on a play respecting σ. We now define H by putting an edge (m, a, m') in H if $\sigma(m) = a$ and and $m' = \delta(m, a, p)$ for some process p. As σ is deterministic and winning, this definition guarantees that H satisfies the first item above. The second item is guaranteed by the reachability property that σ satisfies.

For the other direction, given such a graph H we define a deterministic strategy σ_H. We put $\sigma_H(m) = a$ if (m, a, m') is an edge of H. If m is not a node in H, or has no outgoing edges in H then we put $\sigma_H(m) = b$ for some arbitrary $b \in \sigma_{max}(m)$. It should be clear that σ_H is winning since every play respecting σ_H stays in winning nodes for Eve. By definition $\sigma_H(m) = a$ for all $(m, a) \in P$, and all these (m, a) are reachable on a play respecting σ_H.

So we have reduced the problem stated in the theorem to finding a subgraph H of $G(\sigma_{max})$ as described above. If there is such a subgraph H then there is one in form of a tree, where the edges leading to leaves are of the form (m, a, m') with $(m, a) \in P$. Moreover, there is such a tree with at most $|P|$ nodes with more than one child. So finding such a tree can be done by guessing the $|P|$ branching nodes and solving $|P| + 1$ reachability problems in $G(\sigma_{max})$. This can be done in PTIME since the size of P is bounded by K. ◀

6 Workflows and deterministic negotiations with data

We show how our algorithms from the previous sections can be used to check functional properties of deterministic negotiations, like those studied for workflow systems with data [15]. We take some of the functional properties of [15], and give polynomial algorithms for verifying them over deterministic, acyclic, sound negotiations.

In this section we consider *acyclic*, deterministic negotiations with shared variables over a finite domain, that can be updated by the outcomes of the negotiation. More precisely, each outcome $(n, a) \in N \times R$ comes with a set Σ of operations on these shared variables. In our examples this set Σ is composed of $alloc(x)$, $read(x)$, $write(x)$, and $dealloc(x)$.

Formally, a *negotiation with data* is a negotiation with one additional component: $\mathcal{N} = \langle Proc, N, dom, R, \delta, \ell \rangle$ where $\ell \colon (N \times R) \to \mathcal{P}(\Sigma \times X)$ maps every outcome to a (possibly

Table 1 Data for the negotiation of Figure 1 (adapted from [13]).

atom. neg.	n_0	n_1		n_2	n_3	n_4	n_5		n_6	n_7
outcome	a	a	b	a	a	a	a	b	a	a
alloc	$1, \ldots, 10$									
read		1	1		$1, 8$		5		$2, 7, 9$	$6, 8, 9$
write		$3, 5, 6$	3	$1, 4, 8$	$9, 10$	$2, 7, 10$	7	9	$6, 8$	
dealloc				4		2			$5, 6, 7$	

empty) set of data operations on variables from X. We assume that for each $(n, a) \in N \times R$ and for each variable $x \in X$ the label $\ell(n, a)$ contains at most one operation on x, that is, at most one element of $\Sigma \times \{x\}$.

As an example, we enrich the negotiation of Figure 1 with data, as shown in Table 1. (This example is taken from [13]). The variables are $X = \{x_1, \ldots, x_{10}\}$. The table gives for each outcome and for each operation the set of (indices of the) variables to which the outcome applies this operation.

In [13] some examples of data specifications for workflows are considered. They are presented in the form of anti-patterns, that is, patterns that the negotiation should avoid.

1. *Inconsistent data*: an atomic negotiation reads or writes a variable x while another atomic negotiation is writing, allocating, or deallocating it in parallel.
   In our example there is an execution in which (n_2, a) and (n_6, a) write to x_8 in parallel.
2. *Weakly redundant data*: there is an execution in which a variable is written and never read before it is deallocated or the execution ends.
   In the example, there is an execution in which x_{10} is written by (n_4, a), and never read again.
3. *Never destroyed*: there is an execution in which a variable is allocated and then never deallocated before the execution ends.
   In the example, the execution taking (n_5, b) never deallocates x_2.

It is easy to give algorithms for these properties that are polynomial *in the size of the reachability graph*. We give the first algorithms that check these properties in polynomial time *in the size of the negotiation*, which can be exponentially smaller than its reachability graph.

For the first property we can directly use the algorithm of the previous section: It suffices to check if the negotiation has two outcomes $(m, a), (n, b)$ such that m and n can be concurrently enabled, and there is variable x such that $\ell(a) \cap \{\mathit{read}(x), \mathit{write}(x)\} \neq \emptyset$ and $\ell(b) \cap \{\mathit{write}(x), \mathit{alloc}(x), \mathit{dealloc}(x)\} \neq \emptyset$.

In the rest of the section we present a polynomial algorithm for the following abstract problem, which has the problems (2) and (3) above as special instances. Given sets $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O} \subseteq N \times R$ such that $\mathcal{O}_1 \cup \mathcal{O}_2 \subseteq \mathcal{O}$, we say that the negotiation $\mathcal{N}$ *violates the specification* $(\mathcal{O}_1, \mathcal{O}_2, \mathcal{O})$ if there is a run $w = (n_1, a_1) \cdots (n_k, a_k)$ with indices $0 \leq i < j \leq k$ such that $(n_i, a_i) \in \mathcal{O}_1$, $(n_j, a_j) \in \mathcal{O}_2$, and $(n_l, a_l) \notin \mathcal{O}$ for all $i < l < j$. In this case we also say that $(\mathcal{O}_1, \mathcal{O}_2, \mathcal{O})$ is violated at $(n_i, a_i), (n_j, a_j)$. Otherwise $\mathcal{N}$ *complies with* $(\mathcal{O}_1, \mathcal{O}_2, \mathcal{O})$.

▶ **Example 12.** Observe that variable x is weakly redundant (specification of type (2)) iff $\mathcal{N}$ violates $(\mathcal{O}_1, \mathcal{O}_2, \mathcal{O})$, where $\mathcal{O}_1 = \{(n, a) \in N \times R : \mathit{write}(x) \in \ell(n, a)\}$, $\mathcal{O}_2 = \{(n, b) \in N \times R : n = n_{\mathit{fin}} \vee \mathit{dealloc}(x) \in \ell(n, b)\}$ and $\mathcal{O} = \{(n, c) : \ell(n, c) \cap (\Sigma \times \{x\}) \neq \emptyset\}$.

Variable x is never destroyed (specification of type (3)) iff $\mathcal{N}$ violates $(\mathcal{O}_1, \mathcal{O}_2, \mathcal{O})$, where $\mathcal{O}_1 = \{(n, a) \in N \times R : \mathit{alloc}(x) \in \ell(n, a)\}$, $\mathcal{O}_2 = \{n_{\mathit{fin}}\}$, $\mathcal{O} = \{(n, c) : n = n_{\mathit{fin}} \vee \ell(n, c) \cap \{\mathit{alloc}(x), \mathit{dealloc}(x)\} \neq \emptyset\}$.

For the next proposition it is convenient to use the notation $m \xrightarrow{+} n$, whenever there is a (non-empty) local path in $\mathcal{N}$ from the atomic negotiation m to the atomic negotiation n.

▶ **Proposition 13.** *Let $\mathcal{N}$ be an acyclic, deterministic, sound negotiation with data, and $(\mathcal{O}_1, \mathcal{O}_2, \mathcal{O})$ a specification. Let $(m,a) \in \mathcal{O}_1$, $(n,b) \in \mathcal{O}_2$. Then $\mathcal{N}$ violates $(\mathcal{O}_1, \mathcal{O}_2, \mathcal{O})$ at $(m,a),(n,b)$ iff either $m \parallel n$, or $m \xrightarrow{+} n$ and $\mathcal{N}$ has a run from n_{init} containing $P = \{(m,a),(n,b)\}$, and omitting the set $B = \{m' \in \mathcal{O} : m \xrightarrow{+} m' \xrightarrow{+} n\}$.*

Putting together Proposition 13 and Theorem 11 we obtain:

▶ **Corollary 14.** *Given an acyclic, deterministic, sound negotiation with data $\mathcal{N}$, and a specification $(\mathcal{O}_1, \mathcal{O}_2, \mathcal{O})$, it can be checked in polynomial time whether $\mathcal{N}$ complies with $(\mathcal{O}_1, \mathcal{O}_2, \mathcal{O})$.*

7 Soundness of acyclic weakly non-deterministic negotiations is in PTIME

In previous sections we have presented algorithms for analysis of sound negotiations. Here we show that our techniques also allow to find a bigger class of negotiations for which we can decide soundness in PTIME. The class we consider is that of acyclic, weakly non-deterministic negotiations, c.f. page 5. That is, we allow some processes to be non-deterministic, but every atomic negotiation should involve at least one deterministic process.

Recall that $\mathcal{N}_D$ is the restriction of $\mathcal{N}$ to deterministic processes. Since $\mathcal{N}$ is weakly non-deterministic, every atomic negotiation involves a deterministic process, so $N_D = N$. Recall also that for an acyclic negotiation $\mathcal{N}$ we fixed some linear order $\preccurlyeq_{\mathcal{N}}$ that is a topological order of the graph of $\mathcal{N}$.

The first lemma gives a necessary condition for the soundness of $\mathcal{N}$ that is easy to check. It is proved by showing that $\mathcal{N}_D$ cannot have much more behaviours than $\mathcal{N}$.

▶ **Lemma 15.** *If $\mathcal{N}$ is a sound, acyclic, weakly non-deterministic negotiation then $\mathcal{N}_D$ is sound.*

We then first consider the case of a negotiation with only one non-deterministic process. The next lemma reduces (un)soundness of $\mathcal{N}$ to some pattern in $\mathcal{N}_D$.

▶ **Lemma 16.** *Let $\mathcal{N}$ be an acyclic, weakly non-deterministic negotiation with only one non-deterministic process p. Then $\mathcal{N}$ is not sound, if and only if, either:*
- *$\mathcal{N}_D$ is not sound, or*
- *$\mathcal{N}_D$ is sound, and it has two nodes $m \preccurlyeq_{\mathcal{N}} n$ with outcomes $a \in out(m)$, $b \in out(n)$ such that:*
  - *$p \in dom(m) \cap dom(n)$, $n \notin S_p = \delta(m,a,p)$, and*
  - *there is a successful run of $\mathcal{N}_D$ containing $P = \{(m,a),(n,b)\}$ and omitting $B = \{n' \in S_p : m \prec_{\mathcal{N}} n' \prec_{\mathcal{N}} n\}$.*

The next lemma deals with the case when there is more than one non-deterministic process.

▶ **Lemma 17.** *An acyclic weak non-deterministic negotiation $\mathcal{N}$ is not sound if and only if:*
1. *either its restriction $\mathcal{N}_D$ to deterministic processes is not sound,*
2. *or, for some non-deterministic process p, its restriction $\mathcal{N}^p$ to p and the deterministic processes is not sound.*

▶ **Theorem 18.** *Soundness can be decided in* Ptime *for acyclic, weakly non-deterministic negotiations.*

Proof. By Lemma 17 we can restrict to negotiations $\mathcal{N}$ with one non-deterministic process. For every $m \preccurlyeq_{\mathcal{N}} n$, a and b we check the conditions described in Lemma 16. The existence of a run of $\mathcal{N}_D$ can be checked in Ptime thanks to Theorem 11 and the fact that the size of P is constant. ◀

8 Beyond acyclic weakly non-deterministic negotiations

In this section we show that the polynomial-time result for acyclic, weakly non-deterministic negotiations from Section 7 requires both acyclicity and weak non-determinism. We prove that if we remove one of the two assumptions then the problem becomes coNP-complete. Indeed, even a very mild extension of acyclicity makes the soundness problem coNP-complete.

It is not very surprising that deciding soundness for acyclic, non-deterministic negotiations is coNP-complete. The problem is in coNP since all runs are of polynomial size, so it suffices to guess a run and check if the reached configuration is a deadlock. The hardness is by a simple reduction of SAT to the complement of the soundness problem. It strongly relies on non-determinism.

▶ **Proposition 19.** *Soundness of acyclic non-deterministic negotiations is* coNP*-complete.*

In view of the above proposition, the other possibility is to keep weak non-determinism and relax the notion of acyclicity. We consider a very mild relaxation: deterministic processes still need to be acyclic. This condition implies that all the runs are of polynomial size. We show that even for *very* weakly non-deterministic negotiations (c.f. page 5) the problem is already coNP-complete.

▶ **Theorem 20.** *Non-soundness of det-acyclic, very weakly non-deterministic negotiations is* NP*-complete.*

9 Conclusions

Analysis of concurrent systems is very often Pspace-hard because of the state explosion problem. One way to address this problem is to look for restricted classes of concurrent systems which are non-trivial, and yet are algorithmically easier to analyze. We argue in this paper that negotiations are well adapted for this task. Processes in a negotiation are stateless, at every moment their state is the set of negotiations they are willing to engage. When processes are non-deterministic this mechanism can simulate states, so that the interesting cases occur when non-determinism is limited. These limitations are still relevant as show examples from workflow nets. In short, the negotiation model offers a simple way to formulate restrictions that are sufficiently expressive and algorithmically relevant.

We have shown that a number of verification problems for sound deterministic acyclic negotiations can be solved in Ptime or even in Nlogspace. In our application to workflow Petri nets, acyclicity and determinism (equivalent to free-choiceness) are quite common: about 70% of the industrial workflow nets of [16, 7, 6] are free-choice, and about 60% are both acyclic and free-choice (see e.g. the table at the end of [6]).

Open problems. It would be interesting to have a better understanding what verification problems for deterministic, acyclic, sound negotiations can be solved in Ptime. The coNP result for weakly-deterministic negotiations shows that one should proceed carefully here: allowing arbitrary products with finite automata would not work.

References

1 Jörg Desel and Javier Esparza. Negotiations and Petri nets. In *Int. Workshop on Petri Nets and Software Engineering (PNSE'15)*, volume 1372 of *CEUR Workshop Proceedings*, pages 41–57. CEUR-WS.org, 2015.

2 Volker Diekert and Grzegorz Rozenberg, editors. *The Book of Traces*. World Scientific, Singapore, 1995.

3 Javier Esparza and Jörg Desel. On negotiation as concurrency primitive. In *CONCUR*, pages 440–454, 2013. Extended version in CoRR abs/1307.2145.

4 Javier Esparza and Jörg Desel. On negotiation as concurrency primitive II: Deterministic cyclic negotiations. In *FoSSaCS*, pages 258–273, 2014. Extended version in CoRR abs/1403.4958.

5 Javier Esparza and Jörg Desel. Negotiation programs. In *Application and Theory of Petri Nets and Concurrency*, volume 9115 of *LNCS*, pages 157–178. Springer, 2015.

6 Javier Esparza and Philipp Hoffmann. Reduction rules for colored workflow nets. In *FASE*, volume 9633 of *LNCS*, pages 342–358. Springer, 2016.

7 Dirk Fahland, Cédric Favre, Barbara Jobstmann, Jana Koehler, Niels Lohmann, Hagen Völzer, and Karsten Wolf. Instantaneous soundness checking of industrial business process models. In *Business Process Management*, pages 278–293. Springer, 2009.

8 Cédric Favre, Hagen Völzer, and Peter Müller. Diagnostic information for control-flow analysis of workflow graphs (a.k.a. free-choice workflow nets). In *TACAS 2016*, volume 9636 of *LNCS*, pages 463–479. Springer, 2016.

9 Blaise Genest, Dietrich Kuske, and Anca Muscholl. A Kleene theorem and model checking algorithms for existentially bounded communicating automata. *Inf.& Comput.*, 204(6):920–956, 2006.

10 Anca Muscholl. Automated synthesis of distributed controllers. In *ICALP 2015*, volume 9135 of *LNCS*, pages 11–27. Springer, 2015.

11 Natalia Sidorova, Christian Stahl, and Nikola Trcka. Soundness verification for conceptual workflow nets with data: Early detection of errors with the most precision possible. *Inf. Syst.*, 36(7):1026–1043, 2011.

12 Wolfgang Thomas. Church's problem and a tour through automata theory. In *Pillars of Computer Science, Essays Dedicated to Boris (Boaz) Trakhtenbrot on the Occasion of His 85th Birthday*, volume 4800 of *LNCS*, pages 635–655. Springer, 2008.

13 Nikola Trcka, Wil M. P. van der Aalst, and Natalia Sidorova. Data-flow anti-patterns: Discovering data-flow errors in workflows. In *Advanced Information Systems Engineering (CAiSE)*, volume 5565 of *LNCS*, pages 425–439. Springer, 2009.

14 Wil M. P. van der Aalst. Business process management as the "Killer App" for Petri nets. *Software & Systems Modeling*, 14(2):685–691, 2015.

15 Wil M.P. van der Aalst. The application of Petri nets to workflow management. *J. Circuits, Syst. and Comput.*, 08(01):21–66, 1998.

16 B. van Dongen, M. Jansen-Vullers, H.M.W. Verbeek, and Wil M. P. van der Aalst. Verification of the SAP reference models using EPC reduction, state-space analysis, and invariants. *Computers in Industry*, 58(6):578–601, 2007.

17 W. Zielonka. Notes on finite asynchronous automata. *RAIRO–Theoretical Informatics and Applications*, 21:99–135, 1987.