On Negotiation as Concurrency Primitive II: Deterministic Cyclic Negotiations

Javier Esparza[1] and Jörg Desel[2]

[1] Fakultät für Informatik, Technische Universität München, Germany
[2] Fakultät für Mathematik und Informatik, FernUniversität in Hagen, Germany

Abstract. We continue our study of negotiations, a concurrency model with multi-party negotiation as primitive. In a previous paper [7] we have provided a correct and complete set of reduction rules for sound, acyclic, and (weakly) deterministic negotiations. In this paper we extend this result to all deterministic negotiations, including cyclic ones. We also show that this set of rules allows one to decide soundness and to summarize negotiations in polynomial time.

1 Introduction

Negotiation has long been identified as a paradigm for process interaction [5]. It has been applied to different problems (see e.g. [17,2]), and studied on its own [15]. However, there is only little research on negotiations from a concurrency-theoretic point of view. Some works model the behaviour of a negotiation party using business process languages or Petri net, and model negotiation protocols as the concurrent composition of the parties [4,18,16]. In contrast, in [7] we have introduced a formalism that considers each elementary (multiparty) negotiation a single atom (graphically represented by a node) and model a distributed negotiation as a composition of atoms.

Observationally, an elementary negotiation (an atom) is an interaction in which several partners come together to agree on one out of a number of possible outcomes (a synchronized nondeterministic choice). Each possible outcome has associated a state-transformer. Negotiation partners enter the atom in certain states, and leave it in the states obtained by applying to these states the state-transformer of the outcome agreed upon. Atoms are combined into more complex, distributed negotiations by means of a next-atoms function that determines, for each atom, negotiating agent, and outcome, the set of atoms the agent is ready to engage in next if the atom ends with that outcome.

Like in workflow nets [1], distributed negotiations can be *unsound* because of deadlocks or livelocks. The *soundness* problem consists of deciding if a given negotiation is sound. Moreover, a sound negotiation is equivalent to a single atom whose state transformation function determines the possible final states of all parties as a function of their initial states. The *summarization problem* consists of computing such an atomic negotiation, called a *summary*.

A. Muscholl (Ed.): FOSSACS 2014, LNCS 8412, pp. 258–273, 2014.

Negotiations can simulate 1-safe Petri nets (see the arXiv version of [7]), which proves that the soundness problem and (a decision version of) the summarization problem are, unsurprisingly, PSPACE-complete. We have studied in [7] two subclasses: *deterministic* and *weakly deterministic* negotiations. Both have limited expressive power in comparison to general negotiations, but have natural semantic justifications (see [7]). Only deterministic negotiations are relevant for this paper. Loosely speaking, a negotiation is deterministic if, for each agent and each outcome of an atomic negotiation, the next-atom function yields only one next atom, i.e., each agent can always engage in one atom only.

We have shown in [7] that the soundness and summarization problems for *acyclic* deterministic negotiations can be solved in polynomial time. The algorithm progressively reduces the graphical representation of a negation to a simpler one by means of reduction rules. Each rule preserves soundness and summaries (i.e., the negotiation before the application of the rule is sound iff the negotiation after the application is sound, and both have the same summary). Reduction rules have been extensively applied to Petri nets or workflow nets, but most of this work has been devoted to the liveness or soundness problems [3,12,13,11,6], and many rules do not preserve summaries.

In [7] we conjectured that the addition of a simple rule allowing one to reduce trivial cycles yields a complete set of rules for all sound deterministic negotiations. In this paper we prove this result, and we show that the number of rule applications required to summarize a negotiation is still polynomial. While the new rule is very simple, the proof of our result is involved. It is structured in several sections, and some technical proofs have been moved to an extended version of this paper [8]. Section 2 presents the main definitions of [7] in compact form. Section 3 introduces our reduction rules. Section 4 proves that the rules summarize all sound deterministic negotiations. Section 5 proves that the summarization of a sound negotiation requires a polynomial number of steps.

2 Negotiations: Syntax and Semantics

We fix a finite set A of *agents*. Each agent $a \in A$ has a (possibly infinite) nonempty set Q_a of *internal states*. We denote by Q_A the cartesian product $\prod_{a \in A} Q_a$. A *transformer* is a left-total relation $\tau \subseteq Q_A \times Q_A$. Given $S \subseteq A$, we say that a transformer τ is an *S-transformer* if, for each $a_i \notin S$, $\Big((q_{a_1}, \ldots, q_{a_i}, \ldots, q_{a_{|A|}}), (q'_{a_1}, \ldots, q'_{a_i}, \ldots, q'_{a_{|A|}})\Big) \in \tau$ implies $q_{a_i} = q'_{a_i}$. So an S-transformer only transforms the internal states of agents in S.

Definition 1. *A* negotiation atom*, or just an* atom*, is a triple $n = (P_n, R_n, \delta_n)$, where $P_n \subseteq A$ is a nonempty set of* parties*, R_n is a finite, nonempty set of* outcomes*, and δ_n is a mapping assigning to each outcome r in R_n a P_n-transformer $\delta_n(r)$.*

Intuitively, if the states of the agents before a negotiation n are given by a tuple q and the outcome of the negotiation is r, then the agents change their states to q' for some $(q, q') \in \delta_n(r)$.

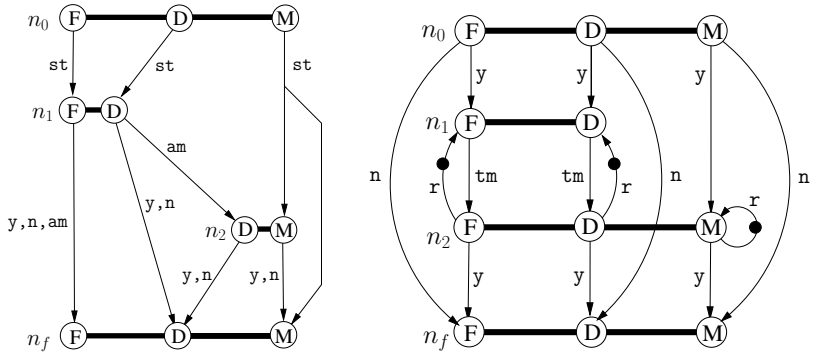

Fig. 1. Acyclic and cyclic negotiations

For a simple example, consider a negotiation atom $n_{\mathtt{FD}}$ with parties `F` (Father) and `D` (teenage Daughter). The goal of the negotiation is to determine whether `D` can go to a party, and the time at which she must return home. The possible outcomes are `yes` and `no`. Both sets $Q_{\mathtt{F}}$ and $Q_{\mathtt{D}}$ contain a state $\perp$ plus a state t for every time $T_1 \leq t \leq T_2$ in a given interval $[T_1, T_2]$. Initially, `F` is in state t_f and `D` in state t_d. The transformer $\delta_{n_{\mathtt{FD}}}$ is given by:

$$\delta_{n_{fd}}(\mathtt{yes}) = \{((t_f, t_d), (t, t)) \mid t_f \leq t \leq t_d \vee t_d \leq t \leq t_f\}$$
$$\delta_{n_{fd}}(\mathtt{no}) = \{((t_f, t_d), (\perp, \perp))\}$$

2.1 Combining Atomic Negotiations

A negotiation is a composition of atoms. We add a *transition function* $\mathcal{X}$ that assigns to every triple (n, a, r) consisting of an atom n, a party a of n, and an outcome r of n a set $\mathcal{X}(n, a, r)$ of atoms. Intuitively, this is the set of atomic negotiations agent a is ready to engage in after the atom n, if the outcome is r.

Definition 2. *Given a finite set of atoms N, let $T(N)$ denote the set of triples (n, a, r) such that $n \in N$, $a \in P_n$, and $r \in R_n$. A* negotiation *is a tuple $\mathcal{N} = (N, n_0, n_f, \mathcal{X})$, where $n_0, n_f \in N$ are the* initial *and* final *atoms, and $\mathcal{X}\colon T(N) \to 2^N$ is the* transition function. *Further, $\mathcal{N}$ satisfies the following properties:*

(1) every agent of A participates in both n_0 and n_f;

(2) for every $(n, a, r) \in T(N)$: $\mathcal{X}(n, a, r) = \emptyset$ iff $n = n_f$.

Negotiations are graphically represented as shown in Figure 1. For each atom $n \in N$ we draw a black bar; for each party a of P_n we draw a white circle on the bar, called a *port*. For each $(n, a, r) \in T(N)$, we draw a hyperarc leading from the port of a in n to all the ports of a in the atoms of $\mathcal{X}(n, a, r)$, and label it by r. Figure 1 shows two Father-Daughter-Mother negotiations. On the left, Daughter and Father negotiate with possible outcomes `yes` (`y`), `no` (`n`), and `ask_mother` (`am`). If the outcome is the latter, then Daughter and Mother negotiate with outcomes `yes`, `no`. In the negotiation on the right, Father, Daughter and Mother

negotiate with outcomes `yes` and `no`. If the outcome is `yes`, then Father and Daughter negotiate a return time (atom n_1) and propose it to Mother (atom n_2). If Mother approves (outcome `yes`), then the negotiation terminates, otherwise (outcome `r`) Daughter and Father renegotiate the return time. For the sake of brevity we do not describe the transformers of the atoms.

Definition 3. *The* graph associated to a negotiation $\mathcal{N} = (N, n_0, n_f, \mathcal{X})$ *is the directed graph with vertices* N *and edges* $\{(n, n') \mid \exists\, (n, a, r) \in T(N)\colon n' \in \mathcal{X}(n, a, r)\}$. $\mathcal{N}$ *is* acyclic *if its graph has no cycles, otherwise it is cyclic.*

The negotiation on the left of Figure 1 is acyclic, the one the right is cyclic.

2.2 Semantics

A *marking* of a negotiation $\mathcal{N} = (N, n_0, n_f, \mathcal{X})$ is a mapping $\boldsymbol{x}\colon A \to 2^N$. Intuitively, $\boldsymbol{x}(a)$ is the set of atoms that agent a is currently ready to engage in next. The *initial* and *final* markings, denoted by $\boldsymbol{x}_0$ and $\boldsymbol{x}_f$ respectively, are given by $\boldsymbol{x}_0(a) = \{n_0\}$ and $\boldsymbol{x}_f(a) = \emptyset$ for every $a \in A$.

A marking $\boldsymbol{x}$ *enables* an atom n if $n \in \boldsymbol{x}(a)$ for every $a \in P_n$, i.e., if every party of n is currently ready to engage in it. If $\boldsymbol{x}$ enables n, then n can take place and its parties agree on an outcome r; we say that (n, r) *occurs*. Abusing language, we will call this pair also an outcome. The occurrence of (n, r) produces a next marking $\boldsymbol{x}'$ given by $\boldsymbol{x}'(a) = \mathcal{X}(n, a, r)$ for $a \in P_n$, and $\boldsymbol{x}'(a) = \boldsymbol{x}(a)$ for $a \in A \setminus P_n$. We write $\boldsymbol{x} \xrightarrow{(n,r)} \boldsymbol{x}'$ to denote this, and call it a *small step*.

By this definition, $\boldsymbol{x}(a)$ is always either $\{n_0\}$ or equals $\mathcal{X}(n, a, r)$ for some atom n and outcome r. The marking $\boldsymbol{x}_f$ can only be reached by the occurrence of (n_f, r) (r being a possible outcome of n_f), and it does not enable any atom. Any other marking that does not enable any atom is considered a *deadlock*.

Reachable markings are graphically represented by tokens (dots) on arcs (on forking points of hyperarcs, respectively). Figure 1 shows on the right a marking in which `F` and `D` are ready to engage in n_1 and `M` is ready to engage in n_2.

We write $\boldsymbol{x}_1 \xrightarrow{\sigma}$ to denote that there is a sequence of small steps

$$\boldsymbol{x}_1 \xrightarrow{(n_1,r_1)} \boldsymbol{x}_2 \xrightarrow{(n_2,r_2)} \cdots \xrightarrow{(n_{k-1},r_{k-1})} \boldsymbol{x}_k \xrightarrow{(n_k,r_k)} \boldsymbol{x}_{k+1} \cdots$$

such that $\sigma = (n_1, r_1) \ldots (n_k, r_k) \ldots$. If $\boldsymbol{x}_1 \xrightarrow{\sigma}$, then σ is an *occurrence sequence* enabled by $\boldsymbol{x}_1$. If σ is finite, then we write $\boldsymbol{x}_1 \xrightarrow{\sigma} \boldsymbol{x}_{k+1}$ and call $\boldsymbol{x}_{k+1}$ *reachable* from $\boldsymbol{x}_1$. If $\boldsymbol{x}_1$ is the initial marking, then we call σ *initial occurrence sequence*. If moreover $\boldsymbol{x}_{k+1}$ is the final marking, then σ is a *large step*.

A negotiation can be associated an equivalent Petri net with the same occurrence sequences (see [7], arXiv version). However, the Petri net can be exponentially larger than the negotiation.

2.3 Soundness

Following [1], we introduce a notion of well-formedness of a negotiation:

Definition 4. *A negotiation is* sound *if* (a) *every atom is enabled at some reachable marking, and* (b) *every occurrence sequence from the initial marking is either a large step or can be extended to a large step.*

The negotiations of Figure 1 are sound. However, if we set in the left negotiation $\mathcal{X}(n_0, \mathtt{M}, \mathtt{st}) = \{n_{\mathtt{DM}}\}$ instead of $\mathcal{X}(n_0, \mathtt{M}, \mathtt{st}) = \{n_{\mathtt{DM}}, n_f\}$, then the occurrence sequence $(n_0, \mathtt{st})(n_{\mathtt{FD}}, \mathtt{yes})$ leads to a deadlock.

Definition 5. *Given a negotiation $\mathcal{N} = (N, n_0, n_f, \mathcal{X})$, we attach to each outcome r of n_f a* summary transformer *$\langle \mathcal{N}, r \rangle$ as follows. Let E_r be the set of large steps of $\mathcal{N}$ that end with (n_f, r). We define $\langle \mathcal{N}, r \rangle = \bigcup_{\sigma \in E_r} \langle \sigma \rangle$, where for $\sigma = (n_1, r_1) \dots (n_k, r_k)$ we define $\langle \sigma \rangle = \delta_{n_1}(r_1) \cdots \delta_{n_k}(r_k)$ (each $\delta_{n_i}(r_i)$ is a relation on Q_A; concatenation is the usual concatenation of relations).*

$\langle \mathcal{N}, r \rangle(q_0)$ is the set of possible final states of the agents after the negotiation concludes with outcome r, if their initial states are given by q_0.

Definition 6. *Two negotiations $\mathcal{N}_1$ and $\mathcal{N}_2$ over the same set of agents are* equivalent *if they are either both unsound, or if they are both sound, have the same final outcomes (outcomes of the final atom), and $\langle \mathcal{N}_1, r \rangle = \langle \mathcal{N}_2, r \rangle$ for every final outcome r. If $\mathcal{N}_1$ and $\mathcal{N}_2$ are equivalent and $\mathcal{N}_2$ consists of a single atom then $\mathcal{N}_2$ is the* summary *of $\mathcal{N}_1$.*

According to this definition, all unsound negotiations are equivalent: if soundness fails, we do not care about the rest. However, an unsound negotiation can have occurrence sequences from the initial to the final marking, and two unsound (and thus equivalent) negotiations may have different such occurrence sequences.

Definition 7. *A negotiation $\mathcal{N}$ is deterministic if for every $(n, a, r) \in T(N)$ there is an atom n' such that $\mathcal{X}(n, a, r) = \{n'\}$*

Graphically, a negotiation is deterministic if there are no proper hyperarcs. The negotiation on the left of Figure 1 is not deterministic (it contains a proper hyperarc for Mother), while the one on the right is deterministic. In the sequel, we often assume that a negotiation is sound and deterministic, and abbreviate "sound and deterministic negotiation" to SDN. For deterministic negotiations we write $\mathcal{X}(n, a, r) = n'$ instead of $\mathcal{X}(n, a, r) = \{n'\}$.

3 Reduction Rules for Deterministic Negotiations

We present three equivalence-preserving reduction rules for negotiations. Two of them were already introduced in [7] (in a slightly more general version), while the iteration rule is new. Here we only consider deterministic negotiations.

A *reduction rule*, or just a rule, is a binary relation on the set of negotiations. Given a rule R, we write $\mathcal{N}_1 \xrightarrow{R} \mathcal{N}_2$ for $(\mathcal{N}_1, \mathcal{N}_2) \in R$. A rule R is *correct* if it preserves equivalence, i.e., if $\mathcal{N}_1 \xrightarrow{R} \mathcal{N}_2$ implies $\mathcal{N}_1 \equiv \mathcal{N}_2$. This implies that $\mathcal{N}_1$ is sound iff $\mathcal{N}_2$ is sound. Given a set of rules $\mathcal{R} = \{R_1, \dots, R_k\}$, we denote

by $\mathcal{R}^*$ the reflexive and transitive closure of $R_1 \cup \ldots \cup R_k$. We call $\mathcal{R}$ *complete with respect to a class of negotiations* if, for every negotiation $\mathcal{N}$ in the class, there is a negotiation $\mathcal{N}'$ consisting of a single atom such that $\mathcal{N} \xrightarrow{\mathcal{R}^*} \mathcal{N}'$. We describe rules as pairs of a *guard* and an *action*; $\mathcal{N}_1 \xrightarrow{R} \mathcal{N}_2$ holds if $\mathcal{N}_1$ satisfies the guard and $\mathcal{N}_2$ is a possible result of applying the action to $\mathcal{N}_1$.

Merge rule. Intuitively, the *merge rule* merges two outcomes with identical next enabled atoms into one single outcome.

Definition 8. *Merge rule*
Guard*: N contains an atom n with two distinct outcomes $r_1, r_2 \in R_n$ such that $\mathcal{X}(n,a,r_1) = \mathcal{X}(n,a,r_2)$ for every $a \in A_n$.*
Action*: (1) $R_n \leftarrow (R_n \setminus \{r_1, r_2\}) \cup \{r_f\}$, where r_f is a fresh name.*
(2) For all $a \in P_n$: $\mathcal{X}(n,a,r_f) \leftarrow \mathcal{X}(n,a,r_1)$.
(3) $\delta(n,r_f) \leftarrow \delta(n,r_1) \cup \delta(n,r_2)$.

Shortcut rule. Intuitively, the shortcut rule merges the outcomes of two atoms that can occur one after the other into one single outcome with the same effect. The examples in Figure 6 illustrate the definition (ignore the big circles for the moment): in both negotiations the outcome (n, r'_f) is a "shortcut" of the outcome (n, r) followed by (n', r').

Definition 9. *Given atoms n, n', we say that (n,r)* unconditionally enables *n' if $P_n \supseteq P_{n'}$ and $\mathcal{X}(n,a,r) = n'$ for every $a \in P_{n'}$.*

Observe that, if (n,r) unconditionally enables n', then, for *every* marking $\boldsymbol{x}$ that enables n, the marking $\boldsymbol{x}'$ given by $\boldsymbol{x} \xrightarrow{(n,r)} \boldsymbol{x}'$ enables n'. Moreover, n' can only be disabled by its own occurrence.

Definition 10. *Shortcut rule for deterministic negotiations*
Guard*: N contains an atom n with an outcome r and an atom n', $n' \neq n$, such that (n,r) unconditionally enables n'.*
Action*: (1) $R_n \leftarrow (R_n \setminus \{r\}) \cup \{r'_f \mid r' \in R_{n'}\}$, where r'_f are fresh names.*
(2) For all $a \in P_{n'}$, $r' \in R_{n'}$: $\mathcal{X}(n,a,r'_f) \leftarrow \mathcal{X}(n',a,r')$.
For all $a \in P \setminus P_{n'}$, $r' \in R_{n'}$: $\mathcal{X}(n,a,r'_f) \leftarrow \mathcal{X}(n,a,r)$.
(3) For all $r' \in R_{n'}$: $\delta_n(r'_f) \leftarrow \delta_n(r)\delta_{n'}(r')$.
(4) If $\mathcal{X}^{-1}(n') = \emptyset$ after (1)-(3), then remove n' from N, where $\mathcal{X}^{-1}(n') = \{(\tilde{n}, \tilde{a}, \tilde{r}) \in T(N) \mid n' \in \mathcal{X}(\tilde{n}, \tilde{a}, \tilde{r})\}$.

Iteration rule. Loosely speaking, the iteration rule replaces the iteration of a negotiation by one single atom with the same effect.

Definition 11. *Iteration rule*
Guard: *N contains an atom n with an outcome r such that $\mathcal{X}(n,a,r) = n$ for every party a of n.*
Action: *(1) $R_n \leftarrow \{r'_f \mid r' \in R_n \setminus \{r\}\}$.*
(2) For every $r'_f \in R_n$: $\delta_n(r'_f) \leftarrow \delta_n(r)^* \, \delta_n(r')$.

Proposition 1. *If the application of the shortcut, merge or iteration rule to a deterministic negotiation $\mathcal{N}$ yields negotiation $\mathcal{N}'$ then $\mathcal{N}'$ is deterministic, too.*

Theorem 1. *The merge, shortcut, and iteration rules are correct.*

Proof. Correctness of the merge and iteration rules is obvious. The correctness of a more general version of the shortcut rule is proved in [7][1]. □

4 Completeness

In [7] we show that every sound and weakly deterministic acyclic negotiation can be summarized to a single atom, and that in the deterministic case the number of rule applications is polynomial (actually, [7] provides a sharper bound than the one in this theorem):

Theorem 2 ([7]). *Every sound deterministic acyclic negotiation $\mathcal{N}$ can be reduced to a single atom by means of $|N|^2 + |Out(\mathcal{N})|$ applications of the merge and shortcut rules, where N is the set of atoms of $\mathcal{N}$, and $Out(\mathcal{N})$ is the set of all outcomes of all atoms of N.*

In the rest of the paper we prove that, surprisingly, the addition of the very simple iteration rule suffices to extend this result to cyclic deterministic negotiations, although with a higher exponent. The argument is complex and requires a detailed analysis of the structure of SDNs.

In this section we present the completeness proof, while the complexity result is presented in the next. We illustrate the reduction algorithm by means of an example. Figure 2 (a) shows a cyclic SDN similar to the Father-Daughter-Mother negotiation on the right of Figure 1. We identify an "almost acyclic" fragment, given by atom n_2 with outcome a and atom n_4 with outcome b. Intuitively, "almost acyclic" means that the fragment can be obtained by "merging" the initial and final atoms of an acyclic SDN; in our example, this is the acyclic SDN shown in Figure 2 (b). This acyclic SDN can be summarized using the shortcut and merge rules. If we apply the same sequence of rules to the fragment mentioned before (with the exception of the last rule, which reduces a negotiation with two different atoms and one single outcome to an atomic negotiation) we obtain the negotiation shown in (c). The self-loop can now be eliminated with the help of the iteration rule, and the procedure can be iterated: we again identify an "almost acyclic" fragment, (d) shows the corresponding acyclic SDN. Its reduction yields the the negotiation shown in (e). The self-loops are eliminated by the iteration rule, yielding an acyclic negotiation, which can be summarized.

In order to prove completeness we must show that every cyclic SDN contains at least one almost acyclic fragment, which is non-trivial. The proof has three parts: We first show that every cyclic SDN has a *loop*: an occurrence sequence from some reachable marking $\boldsymbol{x}$ back to $\boldsymbol{x}$. Then we show that each minimal

[1] The rule of [7] has an additional condition in the guard which is always true for deterministic negotiations.

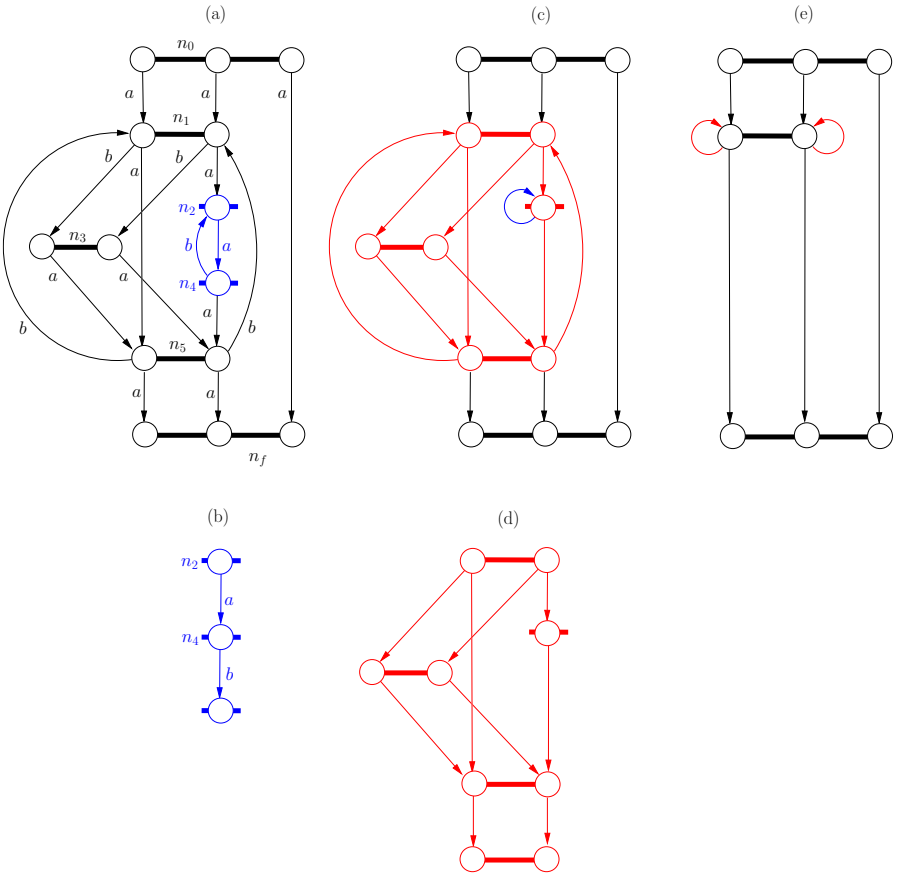

Fig. 2. The reduction procedure

loop has a *synchronizer*: an atom involving each agent that is party of any atom of the loop. Finally we show how to use synchronizers to identify a nonempty and almost acyclic fragment.

4.1 Lassos and Loops

Definition 12. *A* lasso *of a negotiation is a pair* (ρ, σ) *of occurrence sequences such that* σ *is not the empty sequence and* $\boldsymbol{x}_0 \xrightarrow{\rho} \boldsymbol{x} \xrightarrow{\sigma} \boldsymbol{x}$ *for some marking* $\boldsymbol{x}$*. A* loop *is an occurrence sequence* σ *such that* (ρ, σ) *is a lasso for some* ρ*. A* minimal loop *is a loop* σ *satisfying the property that there is no other loop* σ' *such that the set of atoms in* σ' *is a proper subset of the set of atoms in* σ*.*

Observe that lassos and loops are behavioural notions, i.e., structures of the reachability graph of a negotiation. The following result establishes relations between loops and cycles, where cycles are defined on the graph of a negotiation.

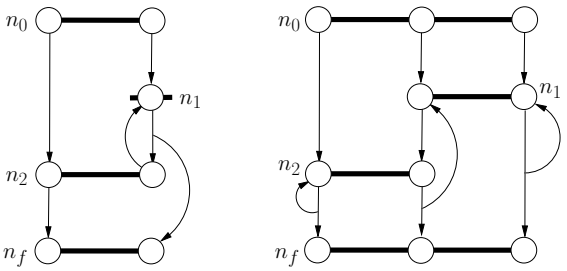

Fig. 3. Two sound and cyclic negotiations

Lemma 1. *(1) Every cyclic SDN has a loop.*
(2) The set of atoms of a minimal loop generates a strongly connected subgraph of the graph of the considered negotiation.

Proof. See [8]. □

4.2 Synchronizers

Definition 13. *A loop* $\sigma = (n_1, r_1) \dots (n_k, r_k)$ *is* synchronized *if there is an atom* n_i *in* σ *such that* $P_j \subseteq P_i$ *for* $1 \leq j \leq k$*, i.e., every party of every atom in the loop is also a party of* n_i*. We call* n_i *a* synchronizer *of the loop. An atom is a synchronizer of a negotiation if it is a synchronizer of at least one of its loops.*

Each loop $\boldsymbol{x} \xrightarrow{(n,r)} \boldsymbol{x}$ is synchronized. In the graph of a negotiation, such a loop appears as a self-loop, i.e., as an edge from atom n to atom n.

Some of the loops of the SDN shown in Figure 2 (a) are $(n_1, a)\,(n_2, a)\,(n_4, a)\,(n_5, b)$, $(n_1, b)\,(n_3, a)\,(n_5, b)$, and $(n_2, a)\,(n_4, b)$. The first loop is synchronized by (n_1, a) and by (n_5, b), the two others are synchronized by all their outcomes.

The main result of this paper is strongly based on the following lemma.

Lemma 2. *Every minimal loop of a SDN is synchronized.*

Proof. See [8] □

The negotiation on the left in Figure 3 shows that Lemma 1(1) holds only in the deterministic case. It is sound and cyclic, but has no loops, because the only big step is $n_0\, n_1\, n_2\, n_1\, n_f$ (all atoms have only one outcome, whose name is omitted).

Lemma 2 does not hold for arbitrary (i.e., non-deterministic) sound negotiations. For the negotiation on the right of Figure 3 (the name of the outcome is again omitted), the sequence $n_1\, n_2$ is a loop without synchronizers.

4.3 Fragments

We assign to each atom n of an SDN a "fragment" $\mathcal{F}_n$ as follows: we take all the loops synchronized by n, and (informally) define $\mathcal{F}_n$ as the atoms and outcomes that appear in these loops. Figure 4 (a) and (c) show $\mathcal{F}_{n_1}$ and $\mathcal{F}_{n_2}$ for the SDN of Figure 2. Since a cyclic SDN has at least one loop and hence also a minimal one, and since every loop has a synchronizer, at least one of the fragments of a cyclic SDN is nonempty.

Given a fragment $\mathcal{F}_n$, let $\mathcal{N}_n$ denote the negotiation obtained by, intuitively, "splitting" the atom n into an initial and a final atom. Figure 4 (b) and (d) show the "splittings" $\mathcal{N}_{n_1}$ and $\mathcal{N}_{n_2}$ of $\mathcal{F}_{n_1}$ and $\mathcal{F}_{n_2}$. Not all fragments are almost acyclic. For instance, $\mathcal{N}_{n_1}$ is not acyclic, and so $\mathcal{F}_{n_1}$ is not almost acyclic. However, we prove that if a fragment is not almost acyclic, then it contains a smaller fragment (for instance, $\mathcal{F}_{n_1}$ contains $\mathcal{F}_{n_2}$). This shows that every minimal fragment is almost acyclic.

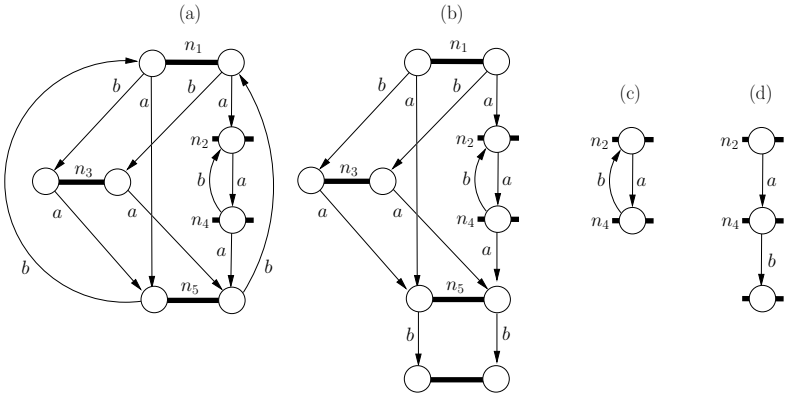

Fig. 4. Fragments of the SDN of Figure 2(a) and their "splittings"

Definition 14. *Let $\mathcal{L}$ be a set of loops of $\mathcal{N}$. Abusing language, we write $(n, r) \in \mathcal{L}$ resp. $n \in \mathcal{L}$ to denote that (n, r) resp. n appears in some loop of $\mathcal{L}$. The* projection *of an atom $n = (P_n, R_n, \delta_n) \in \mathcal{L}$ onto $\mathcal{L}$ is the atom $n_{\mathcal{L}} = (P_{\mathcal{L}}, R_{\mathcal{L}}, \delta_{\mathcal{L}})$, where $P_{\mathcal{L}} = P_n$, $R_{\mathcal{L}} = \{r \mid (n, r) \in \mathcal{L}\}$, and $\delta_{\mathcal{L}}((n_{\mathcal{L}}, r)) = \delta((n, r))$ for every $(n, r) \in \mathcal{L}$.*

Definition 15. *Let s be an atom of a negotiation $\mathcal{N}$, and let $\mathcal{L}$ be the set of loops synchronized by s. The* s-fragment *of $\mathcal{N}$ is the pair $\mathcal{F}_s = (F_s, \mathcal{X}_s)$, where $F_s = \{n_{\mathcal{L}} \mid n \in \mathcal{L}\}$ and $\mathcal{X}_s(n_{\mathcal{L}}, a, r) = \mathcal{X}(n, a, r)$ for every $a \in P_{\mathcal{L}}$ and $r \in R_{\mathcal{L}}$.*

The s-negotiation *of $\mathcal{N}$ is the negotiation $\mathcal{N}_s = (N_s, n_{s0}, n_{sf}, \mathcal{X}'_s)$, where N_s contains the atoms of F_s plus a fresh atom n_{sf}; $n_{s0} = s_{\mathcal{L}}$; and*

$$\mathcal{X}'_s(n_{\mathcal{L}}, a, r) = \begin{cases} \mathcal{X}(n, a, r) & \text{if } \mathcal{X}(n, a, r) \neq s \\ n_{sf} & \text{otherwise} \end{cases}$$

Lemma 3. *A cyclic SDN contains an atom n such that $\mathcal{N}_n$ is an acyclic SDN.*

Proof. See [8] □

The example on the left of Figure 3 shows that this result does not hold for the non-deterministic case.

4.4 The Reduction Procedure

We can now formulate a reduction procedure to summarize an arbitrary SDN.

Input: a deterministic negotiation $\mathcal{N}_0$;

1 $\mathcal{N} \leftarrow$ result of exhaustively applying the merge rule to $\mathcal{N}_0$;
2 **while** $\mathcal{N}$ is cyclic **do**
3 select $s \in N$ such that $\mathcal{N}_s$ is acyclic;
4 apply to $\mathcal{N}$ the sequence of rules used to summarize $\mathcal{N}_s$ (but the last);
5 apply the iteration rule to s;
6 exhaustively apply the merge rule
7 apply the reduction sequence of Theorem 2

Theorem 3. *The reduction procedure returns a summary of $\mathcal{N}_0$ iff $\mathcal{N}_0$ is sound.*

Proof. By induction on the number k of atoms of $\mathcal{N}$ that synchonize at least one loop. If $k = 0$, then by Lemma 1 and 2 $\mathcal{N}$ is acyclic, and the result follows from Theorem 2. If $k > 0$, then by Lemma 3 $\mathcal{N}$ contains an almost acylic fragment $\mathcal{F}_s$, and so $\mathcal{N}_s$ is acyclic. Since the sequence of rules of line 4 summarizes $\mathcal{N}_s$, its application to $\mathcal{N}$ ends with a negotiation having a unique self-loop-outcome on s. After removing this outcome with the iteration rule in line 5, we obtain a SDN with $k-1$ synchronizers, which can be summarized by induction hypothesis (line 6 is not necessary for completeness, but required for the complexity result of the next section). □

5 Complexity

We analyze the number of rule applications required by the reduction procedure. Let $\mathcal{N}_i = (\mathcal{N}_i, n_{0i}, n_{fi}, \mathcal{X}_i)$ be the negotiation before the i-th execution of the while-loop. We next collect some basic properties of the sequence $\mathcal{N}_1, \mathcal{N}_2, \ldots$.

Lemma 4. *For every $i \geq 1$: (a) $N_{i+1} \subseteq N_i$; (b) the merge rule cannot be applied to $\mathcal{N}_i$; and (c) $\mathcal{N}_{i+1}$ has fewer synchronizers than $\mathcal{N}_i$.*
In particular, by (c) the while loop is executed at most $|N_1| = |N_0|$ times.

Proof. (a) and (b) follow immediately from the definitions of the rules and the reduction algorithm. For (c), we observe that every synchronizer of $\mathcal{N}_{i+1}$ is a synchronizer of $\mathcal{N}_i$, but the atom s selected at the i-th loop execution is not a synchronizer of $\mathcal{N}_{i+1}$, because all loops synchronized by s are collapsed to self-loops on s during the i-th iteration of the loop, and then removed by the iteration rule. □

By Theorem 2, during the i-th iteration of the while-loop line 4 requires at most $|N_i|^2 + |Out(\mathcal{N}_i)|$ rule applications. Line 5 only requires one application. Now, let $\mathcal{N}'_i$ be the negotiation obtained after the execution of line 5. The number of rule applications of line 6 is clearly bounded by the number of outcomes of $Out(\mathcal{N}'_i)$. For the total number of rule applications $Appl(\mathcal{N}_0)$ we then obtain

$$
\begin{aligned}
Appl(\mathcal{N}_0) &\leq \sum_{i=1}^{|N_0|} (|N_i|^2 + |Out(\mathcal{N}_i)| + 1 + |Out(\mathcal{N}'_i)|) && \text{Lemma 4(c) and Theorem 2} \\
&\leq \sum_{i=1}^{|N_0|} (|N_0|^2 + 1 + |Out(\mathcal{N}_i)| + |Out(\mathcal{N}'_i)|) && \text{Lemma 4(a)} \\
&\in \mathcal{O}(\ |N_0|^3 + |N_0| \textstyle\sum_{i=1}^{|N_0|} |Out(\mathcal{N}_i)| + |Out(\mathcal{N}'_i)|\ ) && (*)
\end{aligned}
$$

However, we cannot yet bound $Appl(\mathcal{N}_0)$ by a polynomial in $|N_0|$ and $|Out(N_0)|$, because, in principle, the number of outcomes of $\mathcal{N}_i$ or $\mathcal{N}'_i$ might grow exponentially with i. Indeed, the shortcut rule can increase the number of outcomes. Consider the degenerate negotiation $\mathcal{N}$ with only one agent shown in Figure 5(a). $\mathcal{N}$ has one single loop, namely $(n_1, a)\,(n_3, a)\,(n_4, b)$. The corresponding fragment $\mathcal{F}_{n_1}$ consists of the atoms and outcomes of this loop, and $\mathcal{N}_{n_1}$ is shown below $\mathcal{N}$. The negotiation $\mathcal{N}_{n_1}$ can be summarized by three applications of the shortcut rule, shown in the lower row of the figure. The upper row shows the result of application of the same rules to $\mathcal{N}$.

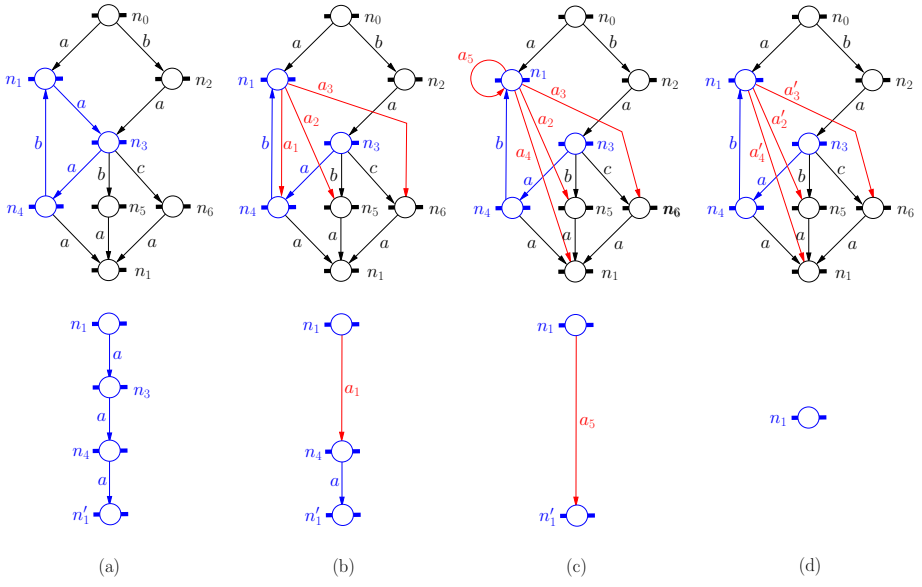

Fig. 5. Reducing an SND with one agent

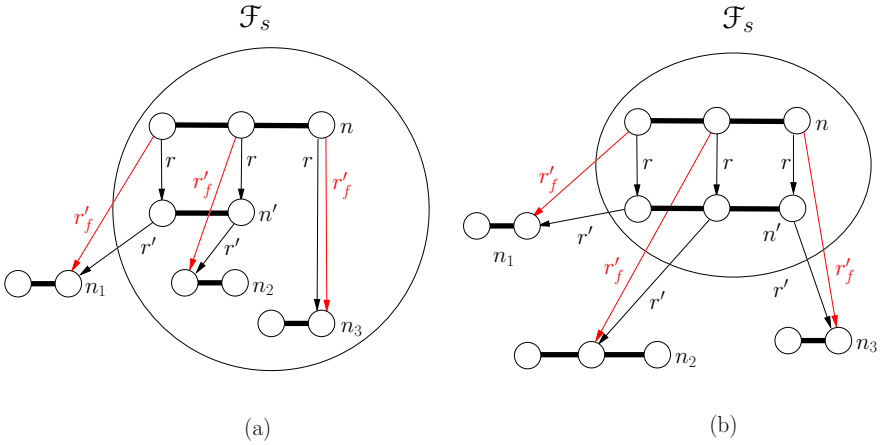

Fig. 6. Exits of SNDs

The first application removes n_3 from $\mathcal{N}_{n_1}$ but not from $\mathcal{N}$, because n_3 has more than one input arc in $\mathcal{N}$ (Figure 5(b)). Moreover, the rule adds three outcomes to $\mathcal{N}$ (outgoing arcs of n_1). The second application removes n_4 from $\mathcal{N}_{n_1}$ but not from $\mathcal{N}$, and adds two new outcomes (n_1, a_4) and (n_1, a_5) (Figure 5(c)). The third application removes n'_1 from $\mathcal{N}_{n_1}$; in $\mathcal{N}$ it is replaced by an application of the iteration rule, yielding the negotiation at the top of Figure 5(d), which has two outcomes more than the initial one.

To solve this problem we introduce *sources*, *targets* and *exits*.

5.1 Sources, Targets, and Exits

Definition 16. *Let $\mathcal{N} = (N, n_0, n_f, \mathcal{X})$ be a negotiation, and let (n, r) be an outcome. The source of (n, r) is n. The* target *of (n, r) is the partial function $A \to N$ that assigns to every party $a \in P_n$ the atom $\mathcal{X}(n, a, r)$, and is undefined for every $a \in A \setminus P_n$. The set of* targets of $\mathcal{N}$*, denoted by $Ta(\mathcal{N})$, contains the targets of all outcomes of $\mathcal{N}$.*

Consider the reduction process from $\mathcal{N}_i$ to $\mathcal{N}_{i+1}$. It proceeds by applying to $\mathcal{N}_i$ the same sequence of rules that summarizes an acyclic negotiation $\mathcal{N}_s$. This sequence progressively reduces the fragment $\mathcal{F}_s$ until it consists of self-loops on the atom s, which can then be reduced by the iteration rule. However, the sequence also produces new outcomes of s that leave $\mathcal{F}_s$, and that become outcomes of $\mathcal{N}_{i+1}$ not present in $\mathcal{N}_i$. Consider for instance Figure 6(a), which sketches an application of the shortcut rule. The outcome (n, r) unconditionally enables n', whose outcome (n', r') makes the left agent leave $\mathcal{F}_s$. The target of (n, r'_f) assigns the agents of the negotiations to atoms n_1, n_2 and n_3, respectively. This target is different from the targets of the other atoms in the figure.

We investigate the sources and targets of outcomes that leave $\mathcal{F}_s$. We call them *exits of $\mathcal{F}_s$*.

Definition 17. *Let $\mathcal{F}_s$ be a fragment of $\mathcal{N}$. An* exit *of $\mathcal{F}_s$ is an outcome $(n, r) \in Out(\mathcal{N})$ such that $n \in F_s$ but $(n, r) \notin Out(\mathcal{F}_s)$.*

The following lemma presents a key property of the exits of fragments of SDNs: the occurrence of an exit (n, r) of $\mathcal{F}_s$ forces all agents of P_s to leave the fragment $\mathcal{F}_s$. In other words: all agents of P_s are parties of n, and the occurrence of (n, r) does not lead any agent back to an atom of $\mathcal{F}_s$.

Lemma 5. *Let $\mathcal{F}_s$ be a fragment of a SDN $\mathcal{N}$, and let (e, r_e) be an exit of $\mathcal{F}_s$. Then e has the same agents as s (i.e., e is also a synchonizer of $\mathcal{F}_s$), and $\mathcal{X}(e, a, r_e) \notin F_s$ for every agent a of e.*

Proof. See [8]. □

In particular, the situation of Figure 6(a) cannot occur, and so in SDNs the correct picture for the application of the shorcut rule to exits is the one of Figure 6(b): the exit n' has the same agents as the synchronizer s. Moreover, the new target of (s, r'_f) equals the already existing target of (n', r'). So Lemma 5 leads to the following bound on the number of targets of $\mathcal{N}_i$:

Lemma 6. *For every $1 \leq i \leq |N_0|$: $Ta(\mathcal{N}_i) \subseteq Ta(N_0)$.*

Proof. See [8]. □

We use this lemma to bound $Out(N'_i)$.

Lemma 7. *For every $1 \leq i \leq |N_0|$: $|Out(N'_i)| \in \mathcal{O}(|N_0|^2 \cdot |Out(N_0)|)$.*

Proof. We first give an upper bound for $|Out(N_i)|$. Since the merge rule cannot be applied to $\mathcal{N}_i$, no two outcomes of $\mathcal{N}_i$ have the same source and the same target, and so $|Out(N_i)| \leq |N_i| \cdot |Ta(N_i)|$.

By Lemma 6, $|Out(N_i)| \leq |N_0| \cdot |Out(N_0)|$.

Now we consider $|Out(N'_i)|$. Each outcome of $Out(\mathcal{N}'_i) \setminus Out(\mathcal{N}_i)$ has some atom of $\mathcal{F}_s$ as source, and is generated by some exit of $\mathcal{F}_s$. So the number of such outcomes is at most the product of the numbers of nodes of $\mathcal{F}_s$ and the number of exits of $\mathcal{F}_s$. Since these numbers are bounded by $|N_i|$ and $|Out(\mathcal{N}_i)|$ respectively, we get $|Out(N'_i)| \leq |Out(N_i)| + |N_i| \cdot |Out(\mathcal{N}_i)|$. The result now follows from $|Out(N_i)| \leq |N_0| \cdot |Out(N_0)|$ and Lemma 4(a). □

Finally, combining $(*)$ and Lemma 7 we get

Theorem 4. *Let $\mathcal{N}_0$ be an SDN. Then $Appl(\mathcal{N}_0) \in \mathcal{O}(\ |N_0|^4 \cdot Out(\mathcal{N}_0)\ )$.*

We conjecture that a more detailed complexity analysis can improve this bound to at least $\mathcal{O}(|N_0|^3 \cdot Out(\mathcal{N}_0))$, but this is beyond the scope of this paper.

6 Conclusions

We have continued the analysis of negotiations started in [7]. We have provided a set of three reduction rules that can summarize all and only the sound deterministic negotiations. Moreover, the number of rule applications is polynomial in the size of the negotiation.

The completeness and polynomiality proofs turned out to be quite involved. At the same time, we think they provide interesting insights. In particular, the completeness proofs shows how in deterministic negotiations soundness requires to synchronize all agents at least once in every loop. It also shows that, intuitively, loops must be properly nested. Intuitively, sound deterministic negotiations are *necessarily* well structured, in the sense of structured programming.

Our rules generalize the rules used to transform finite automata into regular expressions by eliminating states [14]. Indeed, deterministic negotiations can be seen as a class of communicating deterministic automata, and thus our result becomes a generalization of Kleene's theorem to a concurrency model. In future work we plan to investigate the connection to other concurrent Kleene theorems in the literature like e.g. [9,10].

References

1. van der Aalst, W.M.P.: The application of Petri nets to workflow management. J. Circuits, Syst. and Comput. 08(01), 21–66 (1998)
2. Atdelzater, T., Atkins, E.M., Shin, K.G.: QoS negotiation in real-time systems and its application to automated flight control. IEEE Transactions on Computers 49(11), 1170–1183 (2000)
3. Berthelot, G.: Transformations and decompositions of nets. In: Brauer, W., Reisig, W., Rozenberg, G. (eds.) APN 1986. LNCS, vol. 254, pp. 359–376. Springer, Heidelberg (1987)
4. Chen, Y., Peng, Y., Finin, T., Labrou, Y., Cost, S., Chu, B., Sun, R., Wilhelm, B.: A negotiation-based multi-agent system for supply chain management. In: Proceedings of Agents 1999 Workshop on Agent Based Decision-Support for Managing the Internet-Enabled Supply-Chain, pp. 15–20 (1999)
5. Davis, R., Smith, R.G.: Negotiation as a metaphor for distributed problem solving. Artificial Intelligence 20(1), 63–109 (1983)
6. Desel, J., Esparza, J.: Free choice Petri nets. Cambridge University Press, New York (1995)
7. Esparza, J., Desel, J.: On negotiation as concurrency primitive. In: D'Argenio, P.R., Melgratti, H. (eds.) CONCUR 2013. LNCS, vol. 8052, pp. 440–454. Springer, Heidelberg (2013); (Extended version in arXiv:1307.2145)
8. Esparza, J., Desel, J.: On negotiation as concurrency primitive II: Deterministic cyclic negotiations. Technical report, Technische Universität München, Germany. Available via arxiv.org (2014)
9. Gastin, P., Petit, A., Zielonka, W.: An extension of Kleene's and Ochmanski's theorems to infinite traces. Theor. Comput. Sci. 125(2), 167–204 (1994)
10. Genest, B., Kuske, D., Muscholl, A.: A Kleene theorem and model checking algorithms for existentially bounded communicating automata. Inf. Comput. 204(6), 920–956 (2006)

11. Genrich, H.J., Thiagarajan, P.S.: A theory of bipolar synchronization schemes. Theor. Comput. Sci. 30, 241–318 (1984)
12. Haddad, S.: A reduction theory for coloured nets. In: Rozenberg, G. (ed.) APN 1989. LNCS, vol. 424, pp. 209–235. Springer, Heidelberg (1990)
13. Haddad, S., Pradat-Peyre, J.-F.: New efficient Petri nets reductions for parallel programs verification. Parallel Processing Letters 16(1), 101–116 (2006)
14. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation, 3rd edn. Addison-Wesley Longman Publishing Co., Inc., Boston (2006)
15. Jennings, N.R., Faratin, P., Lomuscio, A.R., Parsons, S., Wooldridge, M.J., Sierra, C.: Automated negotiation: prospects, methods and challenges. Group Decision and Negotiation 10(2), 199–215 (2001)
16. Simon, C.: Negotiation Processes – The Semantic Process Language and Applications. Shaker, Aachen (2008)
17. Winsborough, W.H., Seamons, K.E., Jones, V.E.: Automated trust negotiation. In: Proceedings of the DARPA Information Survivability Conference and Exposition, DISCEX 2000, vol. 1, pp. 88–102. IEEE (2000)
18. Xu, H., Shatz, S.M.: An agent-based Petri net model with application to seller/buyer design in electronic commerce. In: Proceedings of the 5th International Symposium on Autonomous Decentralized Systems, pp. 11–18. IEEE (2001)