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 negotations, a concurrency model with multiparty 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. [15, 3]), and studied on its own [14]. However, it has not yet been studied from a concurrency-theoretic point of view. In [7] we have initiated a study of negotiation as communication primitive.

Observationally, a negotiation is an interaction in which several partners come together to agree on one out of a number of possible outcomes (a synchronized nondeterministic choice). In [7] we have introduced *negotiations*, a Petri-net like concurrency model combining multiparty "atomic" negotiations or *atoms* into more complex *distributed negotiations*. Each possible outcome of an atom has associated a state-transformer. Negotiation partners enter the atom in certain initial states, and leave it in the states obtained by applying to the initial 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.

Negotiations are close to a colored version of van der Aalst's *workflow nets* [1]. Like in workflow nets, 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 internal states of all parties as a function of their initial internal states. The *summarization problem* consists of computing such an atomic negotiation, called a *summary*.

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. For this reason we

have studied in [7] two natural classes: *deterministic* and *weakly deterministic* negotiations. 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.

In particular, we have shown in [7] that the soundness and summarization problems for *acyclic* deterministic negotiations can be solved in polynomial time. (Notice that the state space of a dterministic negotiation can be exponentially larger then the negotiation itself). The algorithm takes the graphical representation of a reduction procedure in which the original negotiation is progressively reduced to a simpler one by means of a set 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 [4, 11, 12, 10, 6], and many rules, like for example the linear dependency rule of [6], 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 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 very involved. It is structured in several sections, and some technical proofs have been moved to an appendix. More precisely, the paper is structured as follows. Sections 2 and 3 presents the main definitions of [7] in compact form. Section 4 introduces our set of three reduction rules. Section 5 proves that the rules summarize all sound deterministic negotiations. Section 6 proves that the summarization requires a polynomial number of steps.

2 Negotiations: Syntax and Semantics

We recall the main definitions of [7], and refer to this paper for more details.

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 $\delta_n(r)$.*

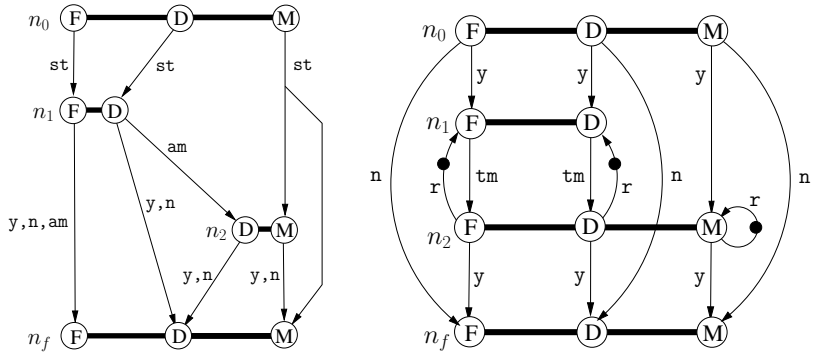

Fig. 1. Acyclic and cyclic negotiations.

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 $(q, q') \in \delta_n(r)$.

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` (`y`) 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

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

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 participant 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 of n 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* $\{(n, n') \in N \times N \mid \exists\, (n, a, r) \in T(N) \colon n' \in \mathcal{X}(n, a, r)\}$. *The negotiation* $\mathcal{N}$ *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 every $a \in P_n$, and $\boldsymbol{x}'(a) = \boldsymbol{x}(a)$ for every $a \in A \setminus P_n$. 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 placing tokens (black dots) on the forking points of the hyperarcs (or in the middle of an arc). 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

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

of small steps 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* from the marking $\boldsymbol{x}_1$, and $\boldsymbol{x}_1$ enables σ. If σ is finite, then we write $\boldsymbol{x}_1 \xrightarrow{\sigma} \boldsymbol{x}_{k+1}$ and say that $\boldsymbol{x}_{k+1}$ is *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*.

Negotiations and Petri nets. A negotiation can be assoicated an equivalent Petri net with the same occurrence sequences (see [7], arXiv version). However, in the worst case the Petri net is exponentially larger.

2.3 Soundness

Following [1, 2], 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 $\mathcal{N}_1$ are equivalent and $\mathcal{N}_2$ and $\mathcal{N}_2$ consists of a single atom, then $\mathcal{N}_2$ is the* summary *of $\mathcal{N}_1$.*

Notice that, according to this definition, all unsound negotiations are equivalent. This amounts to considering soundness essential for a negotiation: if it fails, we do not care about the rest.

3 Deterministic Negotiations

We introduce deterministic negotiations.

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

In the rest of the paper we write $\mathcal{X}(n, a, r) = n'$ instead of $\mathcal{X}(n, a, r) = \{n'\}$.

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 dterministic negotiation" to SDN.

4 Reduction Rules for Deterministic Negotiations

We present three equivalence-preserving reduction rules for negotiations. Two of them were already introduced in [7], while the iteration rule is new.

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 $\mathcal{N}_1 \equiv \mathcal{N}_2$. In particular, this implies that $\mathcal{N}_1$ is sound iff $\mathcal{N}_2$ is sound.

Given a set of rules $\mathcal{R} = \{R_1, \ldots, R_k\}$, we denote by $\mathcal{R}^*$ the reflexive and transitive closure of $R_1 \cup \ldots \cup R_k$. We say that $\mathcal{R}$ is *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 $\mathcal{N}_1$.

Slightly more general versions of the following rules have been presented in [7]. Here we only consider deterministic negotiations.

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. Inituitively, the shortcut rule merges the outcomes of two atoms that can occur one after the other into one single outcome with the same effect. Figure 6 illustrates the definition (ignore the big circle for the moment): the outcome (n, r_f), shown in red, is the "shortcut" of the outcome (n, r) followed by the outcome (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 $r'_f \in R_n$: $\delta_n(r'_f) \leftarrow \delta_n(r)^* \, \delta_n(r')$.

It is important to notice that reductions preserve determinism:

Proposition 1. *If a negotiation* $\mathcal{N}$ *is deterministic and the application of the shortcut, merge or iteration rule 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][3]. □

5 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 section 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

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

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, namely the fragment coloured blue in the figure. 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 blue 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 blue fragment (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 blue self-loop can now be eliminated with the help of the iteration rule, and the procedure can be iterated: We identify an "almost acyclic" fragment, coloured red. Its reduction yields the the negotiation shown in (e). The self-loop is 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 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.

5.1 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 occurrence sequence 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.

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 appendix □

5.2 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 every $1 \leq j \leq k$, 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.*

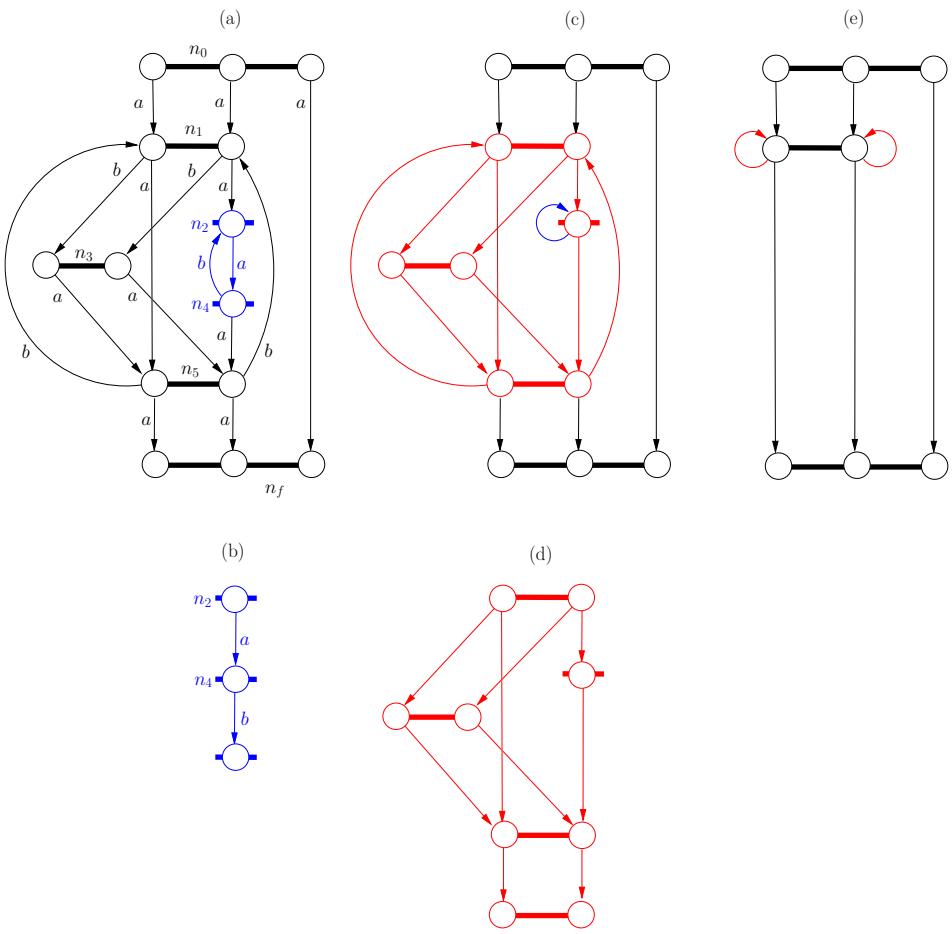

Fig. 2. The reduction procedure

Observe that each loop $\boldsymbol{x} \xrightarrow{(n,r)} \boldsymbol{x}$ is synchronized. In the graph associated to 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 appendix □

Observe that this lemma does not hold for arbitrary (i.e., non-deterministic) sound negotiations. For the negotiation on the right of Figure 3 (all atoms have

only one outcome, whose name is omitted), the sequence $n_1\, n_2$ is a loop without synchronizers.

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

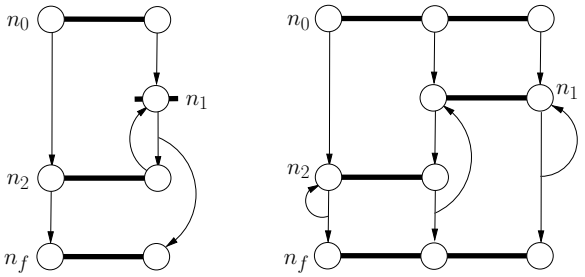

Fig. 3. Two sound and cyclic negotiations

5.3 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.

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 $r \in R_{\mathcal{L}}$.*

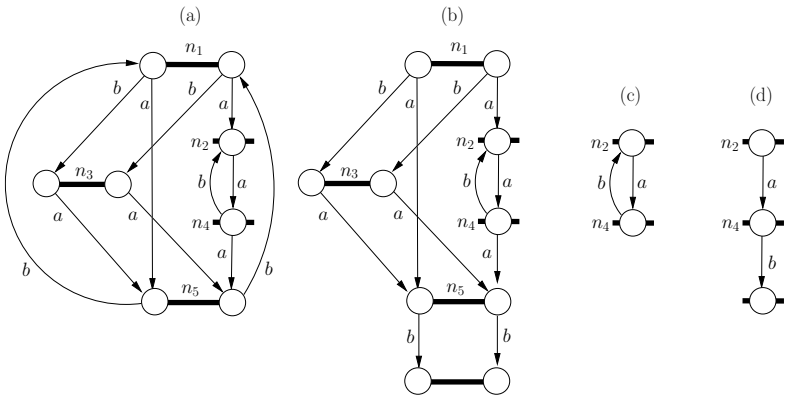

Fig. 4. Fragments of the SDN of Figure 2(a) and their "splittings"

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 appendix □

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

5.4 The reduction procedure

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

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

```
1  N ← result of exhaustively applying the merge rule to N_0;
2  while N is cyclic do
3      select s ∈ N such that N_s is acyclic;
4      apply to N the sequence of rules used to summarize 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 3 and 4 $\mathcal{N}$ is acyclic, and the result follows from Theorem 2. If $k > 0$, then by Lemma 5 $\mathcal{N}$ 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, butu required for the complexity result of the next section).

6 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 oop. The next lemma collects 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.

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

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.

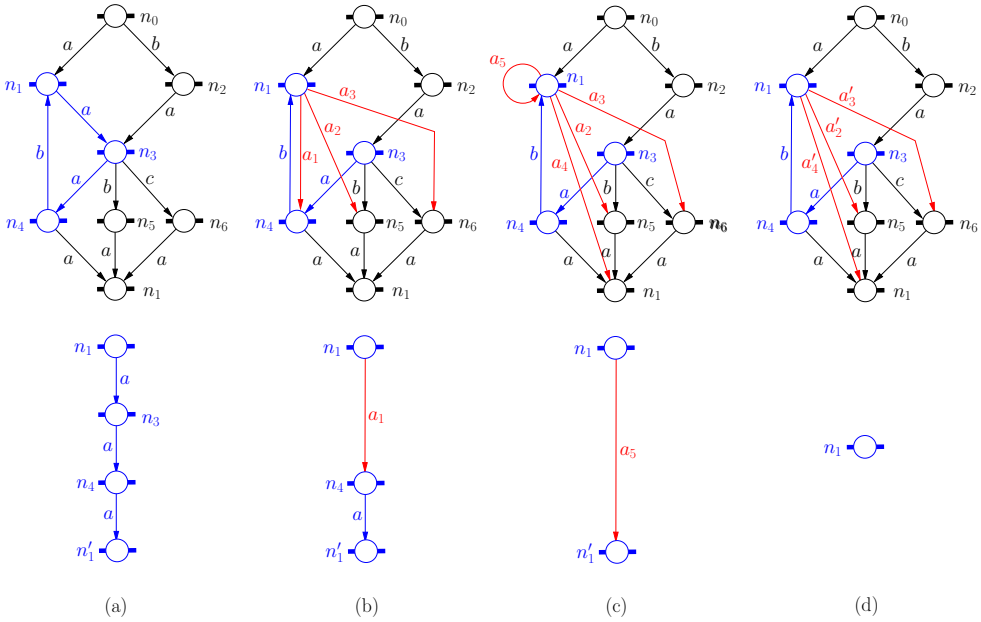

Fig. 5. Reducing an SND with one agent

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 fragment $\mathcal{F}_{n_1}$ is shown in blue, and $\mathcal{N}_{n_1}$ is shown below $\mathcal{N}$. The negotiation $\mathcal{N}_{n_1}$ can be summarized by means of three aplications 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 $\mathcal{N}$.

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 $\mathcal{N}$, shown in red. 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 *targets* and *exits*.

6.1 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 which 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 shorcut 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.

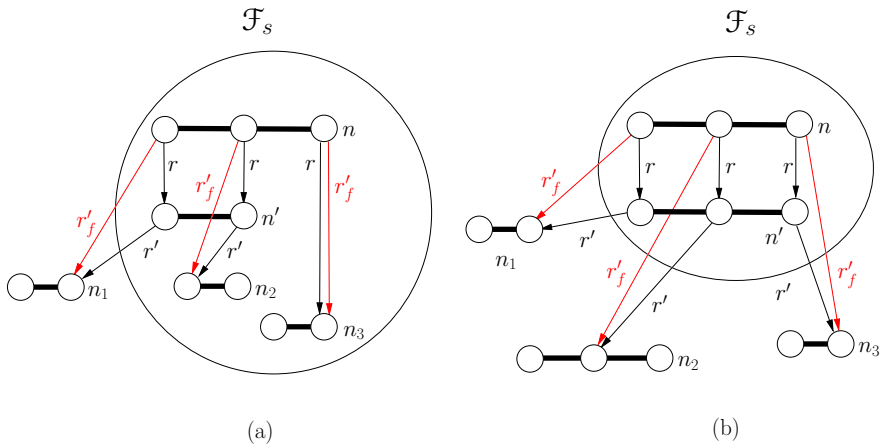

Fig. 6. Exits of SNDs

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.*

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 6 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 appendix. □

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 7, $|Out(N_i)| \leq |N_0| \cdot |Out(N_0)|$.

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.

7 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 teh 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 [13]. 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. [8, 9].

References

1. W. M. P. van der Aalst. The application of Petri nets to workflow management. *J. Circuits, Syst. and Comput.*, 08(01):21–66, 1998.
2. W. M. P. van der Aalst, K. M. van Hee, A. H. M. ter Hofstede, N. Sidorova, H. M. W. Verbeek, M. Voorhoeve, and M. T. Wynn. Soundness of workflow nets: classification, decidability, and analysis. *Formal Asp. Comput.*, 23(3):333–363, 2011.
3. T. Atdelzater, E. M. Atkins, and K. G. Shin. Qos negotiation in real-time systems and its application to automated flight control. *Computers, IEEE Transactions on*, 49(11):1170–1183, 2000.
4. G. Berthelot. Transformations and decompositions of nets. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Advances in Petri Nets*, volume 254 of *LNCS*, pages 359–376. Springer, 1986.
5. R. Davis and R. G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial intelligence*, 20(1):63–109, 1983.
6. J. Desel and J. Esparza. *Free choice Petri nets*. Cambridge University Press, New York, NY, USA, 1995.
7. J. Esparza and J. Desel. On negotiation as concurrency primitive. In P. R. D'Argenio and H. C. Melgratti, editors, *CONCUR*, volume 8052 of *Lecture Notes in Computer Science*, pages 440–454. Springer, 2013. ISBN 978-3-642-40183-1. Extended version in arXiv:1307.2145.
8. P. Gastin, A. Petit, and W. Zielonka. A kleene theorem for infinite trace languages. In J. L. Albert, B. Monien, and M. Rodríguez-Artalejo, editors, *ICALP*, volume 510 of *Lecture Notes in Computer Science*, pages 254–266. Springer, 1991. ISBN 3-540-54233-7.
9. B. Genest, A. Muscholl, and D. Kuske. A kleene theorem for a class of communicating automata with effective algorithms. In C. Calude, E. Calude, and M. J. Dinneen, editors, *Developments in Language Theory*, volume 3340 of *Lecture Notes in Computer Science*, pages 30–48. Springer, 2004. ISBN 3-540-24014-4.
10. H. J. Genrich and P. S. Thiagarajan. A theory of bipolar synchronization schemes. *Theor. Comput. Sci.*, 30:241–318, 1984.
11. S. Haddad. A reduction theory for coloured nets. In G. Rozenberg, editor, *Advances in Petri Nets*, volume 424 of *LNCS*, pages 209–235. Springer, 1988.
12. S. Haddad and J.-F. Pradat-Peyre. New efficient Petri nets reductions for parallel programs verification. *Parallel Processing Letters*, 16(1):101–116, 2006.
13. J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006.
14. N. R. Jennings, P. Faratin, A. R. Lomuscio, S. Parsons, M. J. Wooldridge, and C. Sierra. Automated negotiation: prospects, methods and challenges. *Group Decision and Negotiation*, 10(2):199–215, 2001.
15. W. H. Winsborough, K. E. Seamons, and V. E. Jones. Automated trust negotiation. In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings*, volume 1, pages 88–102. IEEE, 2000.

Appendix

7.1 Proofs of Section 5.1

Lemma 3. *(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. (1) Let π be a cycle of the graph of the negotiation $\mathcal{N}$. Let n_1 be an arbitrary atom occurring in π, and let n_2 be its successor in π. $n_1 \neq n_f$ because n_f has no successor, and hence no cycle contains n_f.
By soundness some reachable marking $\boldsymbol{x}_1$ enables n_1. For at least one agent a and one result r, $\mathcal{X}(n_1, a, r)$ contains n_2, and by determinism it contains only n_2. Let $\boldsymbol{x}_1 \xrightarrow{(n_1,r)} \boldsymbol{x}'_1$. Again by soundness, there is an occurrence sequence from $\boldsymbol{x}'_1$ that leads to the final marking. This sequence has to contain an occurrence of n_2 because this is the only atom agent a is ready to engage in. In particular, some prefix of this sequence leads to a marking $\boldsymbol{x}_2$ that enables n_2.
Repeating this argument arbitrarily for the nodes n_1, n_2, n_3, …, $n_k = n_1$ of the cycle π, we conclude that there is an infinite occurrence sequence, containing infinitely many occurrences of atoms of the cycle π. Since the set of reachable markings is finite, this sequence contains a loop.

(2) For each agent involved in any atom of the loop, consider the sequence of atoms it is involved in. By definition of the graph of the negotiation, this sequence is a path. It is moreover a (not necessarily simple) cycle, because a loop starts and ends with the same marking. So the generated subgraph is covered by cycles. It is moreover strongly connected because, for each connected component, the projection of the outcomes of the loop to those with atoms in this component is a smaller loop, against minimality of the loop. □

7.2 Proofs of Section 5.2

Lemma 4. *Every minimal loop of a SDN is synchronized.*

Proof. Let σ be a minimal loop, enabled at a reachable marking $\boldsymbol{x}$. Define N_σ as the set of atoms that occur in σ and A_σ as the set of agents involved in atoms of N_σ. Since $\mathcal{N}$ is sound, there is an occurrence sequence σ_f enabled by $\boldsymbol{x}$ that ends with the final atom n_f.

Now choose an agent $\hat{a}$ of A_σ such that $\boldsymbol{x}(\hat{a}) \in N_\sigma$. In σ_f, eventually $\hat{a}$ is involved in n_f, and it is first involved in an atom of N_σ.

Using σ_f, we construct a path π of the graph of $\mathcal{N}$ as follows: We begin this path with the last atom $n \in N_\sigma$ that appears in σ_f and involves agent $\hat{a}$. We call this atom n_π. Then we repeatedly choose the last atom in σ_f that involves $\hat{a}$ and moreover is a successor of the last vertex of the path constructed so far. By construction, this path has no cycles (i.e., all vertices are distinct), starts with an atom of N_σ and has not further atoms of N_σ, ends with n_f, and only contains atoms involving $\hat{a}$.

Since $\boldsymbol{x}$ enables the loop σ and since $n_\pi \in N_\sigma$, after some prefix of σ a marking $\boldsymbol{x}_\pi$ is reached which enables n_π. The loop σ continues with some outcome (n_π, r_1), where r_1 is one possible result of n_π.

By construction of the path π, there is an alternative result r_2 of n_π such that $\mathfrak{X}(n_\pi, \hat{a}, r_2)$ is the second atom of the path π, and this atom does not belong to N_σ. Let $\boldsymbol{x}'_\pi$ be the marking reached after the occurrence of (n_π, r_2) at $\boldsymbol{x}_\pi$.

From $\boldsymbol{x}'_\pi$, we iteratively construct an occurrence sequence as follows:

(1) if an atom n of N_σ is enabled and thus some (n, r) occurs in σ, we continue with (n, r),
(2) otherwise, if an atom n of the path π is enabled, we let this atom occur with an outcome r such that $\mathfrak{X}(n, \hat{a}, r)$ is the successor atom w.r.t. the path π,
(3) otherwise we add a minimal occurrence sequence that either leads to the final marking or enables an atom of σ or an atom of π, so that after this sequence one of the previous rules can be applied. Such an occurrence sequence exists because $\mathcal{N}$ is sound and hence the final marking can be reached.

First observe that, in the constructed sequence, agent $\hat{a}$ will always be ready to engage only in an atom of the path π. So its token is moved along π. Conversely, all atoms of π involve $\hat{a}$. Therefore only finitely many atoms of π occur in the sequence. This limits the total number of occurrences of type (2).

Agent $\hat{a}$ is no more ready to engage in any atom of N_σ during the sequence. So at least n_π cannot occur any more in the sequence because $\hat{a}$ is a party of n_π. By minimality of the loop σ, there is no loop with a set of atoms in $N_\sigma \setminus \{n_\pi\}$. Since the set of reachable markings is finite, there cannot be an infinite sequence of outcomes with atoms of N_σ (type (1)) without occurrences of other atoms.

By determinism, each agent ready to engage in an atom of N_σ can only engage in this atom. So the set of these agents is only changed by occurrences of type (1). By construction, no agent ever leaves the loop after the occurrence of (n_π, r_2), i.e. every agent of this set remains in this set by an occurrence of type (1). Therefore, the set of agents ready to engage in an atom of N_σ never decreases.

For each sequence of type (3) we have three possibilities.

(a) It ends with the final marking.
(b) It ends with a marking that enables an atom of π (type (2)), which then occurs next. However these atoms can occur only finitely often in the constructed sequence, as already mentioned.
(c) It ends with a marking that enables an atom of σ (type (1)) which then occurs next. In that case the last outcome of this sequence necessarily involves an agent of A_σ, which after this occurrence is ready to engage in an atom of N_σ. So it increases the number of agents ready to engage in an atom of N_σ. Since this number never decreases, this option can also happen only finitely often.

Hence, eventually only option (a) is possible, and so the sequence will reach the final marking. Since the final atom involves all agents, no agent was able to remain in the loop. In other words: all agents of A_σ left the loop when (n_π, r_2) has occurred. As a consequence, all these agents are parties of n_π, and n_π therefore is a synchronizer of the loop σ. □

7.3 Proofs of Section 5.3

Lemma 5. *A cyclic SDN $\mathcal{N}$ contains an atom n such that $\mathcal{N}_n$ is an acyclic SDN.*

Proof. Let $\mathcal{N}$ be a cyclic SDN. By Lemma 3(1) $\mathcal{N}$ has a loop and hence also a minimal loop. By Lemma 4 this loop has at least one synchronizer. We choose a synchronizer s such that $\mathcal{N}_s$ (which is nonempty for each synchronizer) is minimal.

First we argue that $\mathcal{N}_s$ is sound. Otherwise either some atom can never become enabled, which is impossible because $\mathcal{N}_s$ is built from loops synchronized by s, or a deadlock marking is reached after some occurrence sequence σ. At this marking, we necessarily have that for each atom $n \in N_s$ some party of n is ready to engage in some different $n' \in N_s$, and by determinism only there (remember that all agents of $\mathcal{N}_s$ are parties of s). In $\mathcal{N}$, starting with some reachable marking that enables s, the sequence σ can occur, too. It leads to a marking with the same property: for each atom $n \in N_s$ some party of n is ready to engage in a different atom of $\mathcal{N}_s$. But them, no matter which atoms outside $\mathcal{N}_s$ occur, the atoms of $\mathcal{N}_s$ can never become enabled again. In particular, since all agents of $\mathcal{N}_s$ are parties of n_f, we have that after σ the final atom n_f can never occur, contradicting soundness of $\mathcal{N}$.

Next we claim that $\mathcal{N}_s$ is acyclic. Assume the contrary. Since $\mathcal{N}_n$ is a SDN, we can argue as above and find a loop and an atom n' such that $\mathcal{N}_{nn'}$ is nonempty. Clearly, $\mathcal{N}_{nn'}$ contains fewer atoms than $\mathcal{N}_n$. Now each loop of $\mathcal{N}_s$ is also a loop of $\mathcal{N}$, and $\mathcal{N}_{nn'}$ is equal to $\mathcal{N}_{n'}$. This contradicts the minimality of $\mathcal{N}_n$. □

7.4 Proofs of Section 6.1

We prove Lemma 6. Recall that, intuitively, the lemma states that the occurrence of an exit (e, r_e) 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 (e, r_e) does not lead any agent back to an atom of $\mathcal{F}_s$.

Lemma 6. *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 $P_e = P_s$ (i.e., e has the same parties as s), and $\mathfrak{X}(e, a, r_e) \notin F_s$ for every $a \in P_e$.*

Proof. We proceed indirectly and assume that either $P_e \subset P_s$ ($P_e \subseteq P_s$ by the definition of fragment) or $\mathfrak{X}(e, a, r_e) \in F_s$ for some $a \in P_e$. Then at least one agent $h \in P_s$ satisfies either $h \notin P_e$ or $\mathfrak{X}(e, h, r_e) \in F_s$. We call h a *home agent* (intuitively, an agent that does not leave "home", i.e., $\mathcal{F}_s$, by the occurrence of the exit). We show that the existence of h leads to a contradiction.

We partition the set of A of agents into internal agents, the agents of P_s, and external agents, the agents of $A \setminus P_s$. We also partition the set of atoms: an atom is internal if it has only internal parties, otherwise it is external. Clearly all atoms of F_s are internal, but there can also be internal atoms outside F_s. If P_s contains all agents of the negotiation, then all agents are internal, and so are

all atoms (also the final atom n_f). Otherwise at least n_f has an external party and is hence an external atom.

Next we define a function $p\colon N \to Out(\mathcal{N})$ that assigns to each atom one of its outcomes (the *preferred* outcome). p is defined for internal and external atoms separately, i.e., it is the union of functions p_i assigning outcomes to internal atoms, and p_e assigning outcomes to external atoms.

If there are external atoms, and hence n_f is external, p_e is defined as follows. First we set $p_e(n_f)$ to an arbitrary outcome of n_f. Then we proceed iteratively: If some external atom n has an outcome r and an external agent a such that $p_e(\mathcal{X}(n,a,r))$ is defined, then set $p_e(n) := r$ (if there are several possibilities, we choose one of them arbitrarily). At the end of the procedure p_e is defined for every external atom, because each external atom n has an external agent, say a, and, since a participates in n_f, the graph of $\mathcal{N}$ has a path of atoms, all of them with a as party, leading from n to n_f.

Now we define p_i for internal atoms. For the internal atoms n not in $\mathcal{F}_s$ we define $p_i(n)$ arbitrarily. For the internal atoms $n \in F_s$ such that $\mathcal{X}(n,a,r) = s$ for some agent a we set $p_i(n) = r$. For the rest of the internal atoms of F_s we proceed iteratively. If $n \in F_s$ has an outcome r and an agent a (necessarily internal) such that $p_i(\mathcal{X}(n,a,r))$ is defined, then we set $p_i(n,a) := r$ (if there are several possibilities, we choose one of them). By Lemma 4(2), the graph of $\mathcal{F}_s$ is strongly connected, and so eventually p_i is defined for all atoms of $\mathcal{F}_s$.

Let σ be an arbitrary occurrence sequence leading to a marking $\boldsymbol{x}_s$ that enables s (remember that $\mathcal{N}$ is sound). By the definition of $\mathcal{F}_s$, the marking $\boldsymbol{x}_s$ enables an occurrence sequence σ_e that starts with an occurrence of s, contains only atoms of F_s, and ends with an occurrence of (e, r_e), the considered exit of $\mathcal{F}_s$.

We now define a *maximal* occurrence sequence τ enabled at $\boldsymbol{x}_s$. We start with $\tau := \epsilon$ and while τ enables some atom proceed iteratively as follows:

- If τ enables σ_e, then $\tau := \tau\sigma_e$, i.e., we extend the current sequence with σ_e.
- Otherwise, choose any enabled atom n, and set $\tau := \tau(n, p(n))$, i.e., we extend the current sequence with $(n, p(n)$.

We first show that τ is infinite, i.e., that we never exit the while loop. By soundness, there is always an enabled atom as long as the final marking is not reached, i.e., as long as at least one agent is ready to engage in an atom. So it suffices to show that this is the case. We prove that the home agent h is ready to engage in an atom after the occurrence of an arbitrary finite prefix of τ. This result follows from the following claim.

Claim. If $\boldsymbol{x}_s \xrightarrow{\tau'} \boldsymbol{x}'$ for some prefix τ' of τ then $\boldsymbol{x}'(h) \in F_s$, i.e., the home agent h only participates in atoms of the fragment and is always only ready to participate in atoms of the fragment.

Proof of claim. The proof follows the iterative construction of τ. We start at marking $\boldsymbol{x}_s$, and we have $\boldsymbol{x}_s(h) = s$ because h is a party of s and $\boldsymbol{x}_s$ enables s.

Whenever σ_e or a prefix of σ_e occurs, the property is preserved, because first, $\mathfrak{X}(n,h,r) \in F_s$ holds for all outcomes (n,r) of σ_e except the last one (this holds for all parties of n); and second, for the last outcome, which is (e,r_e), h is either not party of e whence the marking of h does not change, or $\mathfrak{X}(e,h,r_e) \in F_s$ by definition of h.

Whenever an outcome $(n,p(n))$ occurs, either h is not a party of n, and then the marking of h does not change, or h is a party of n, and n is an atom of F_s. By construction of p (actually, of p_i), the property is preserved, which finishes the proof of the claim.

Let us now investigate the occurrences of external and internal atoms in τ. Let G_E be the graph with the external atoms as nodes and an edge from n to n' if $p_e(n) = n'$. By the definition of p_e, the graph G_E is acyclic with n_f as sink. By the definition of τ, after an external atom n occurs in τ, none of its predecessors in G_E can occur in τ. So τ contains only finitely many occurrences of external atoms.

Since τ is infinite, it therefore has an infinite suffix τ' in which only internal atoms occur. Since s is a synchronizer with a minimal set of parties, every internal agent participates in infinitely many outcomes of τ', in particular the home agent h. By the claim, τ' contains infinitely many occurrences of atoms of $\mathcal{F}_s$.

Now let G_s be the graph with the atoms of F_s as nodes, and an edge from n to n' if $p_i(n) = n'$. By the definition of p_i, every cycle of the graph G_s goes through the synchronizer s. So τ' contains infinitely many occurrences of s. Whenever s is enabled, σ_e is enabled, too, and actually occurs by the definition of τ. Since σ_e ends with the outcome (e,r_e), τ' also contains infinitely many occurrences of (e,r_e). Since negotiations have finitely many reachable markings, τ' contains a loop synchronized by s (by minimality of the synchronizer) and containing (e,r_e). However, by the definition of a fragment this implies that this loop and thus (e,r_e) belongs to $\mathcal{F}_s$ as well, contradicting that (e,r_e) is an exit of $\mathcal{F}_s$. □

Lemma 7. *For every* $1 \leq i \leq |N_0|$*:* $Ta(\mathcal{N}_i) \subseteq Ta(N_0)$.

Proof. It suffices to prove $Ta(\mathcal{N}_{i+1}) \subseteq Ta(\mathcal{N}_i)$ for $i < |N_0|$. Let (n,r) be an arbitrary outcome of $\mathcal{N}_{i+1}$. We show that there exists an outcome (n',r') of N_i such that (n,r) and (n',r') have the same targets.

If (n,r) is also an outcome of N_i, then we are done. So assume this is not the case. Then (n,r) is generated by a particular application of the shortcut rule during the reduction process leading from $\mathcal{N}_i$ to $\mathcal{N}_{i+1}$. Let $\mathcal{N}'$ and $\mathcal{N}''$ be the negotiations right before and after this application of the rule. $\mathcal{N}'$ contains a fragment $\mathcal{F}'_s$ obtained by applying to $\mathcal{F}_s$ the same sequence of rules leading from $\mathcal{N}_i$ to $\mathcal{N}'$. Similarly, $\mathcal{N}''$ contains a fragment $\mathcal{F}''_s$.

By the definition of the shortcut rule, $\mathcal{N}'$ has an outcome (n_1,r_1) such that n_1 is an atom of $\mathcal{F}'_s$ and (n_1,r_1) unconditionally enables another atom n_2 of $\mathcal{F}'_s$. Moreover, (n,r) is the shortcut of (n_1,r_1) and (n_2,r_2), i.e., (n,r) is obtained from clause (2) in Definition 10.

We prove the following three claims:

(1) (n_1, r_1) is an outcome of $\mathcal{F}'_s$, i.e., $\mathfrak{X}(n_1, a, r_1) \in F'_s$ for every party a of n_1.
Assume the contrary. Then, since $n_1 \in F'_s$, (n_1, r_1) is an exit of $\mathcal{F}'_s$, by Lemma 6 we have $\mathfrak{X}(n_1, a, r_1) \notin F'_s$ for every party a of n_1, contradicting that (n_1, r_1) unconditionally enables an atom of $\mathcal{F}'_s$.

(2) (n_2, r_2) is an exit of $\mathcal{F}'_s$.
Assume the contrary, i.e., $(n_2, r_2) \in \mathcal{F}'_s$. By (1), both (n_1, r_1) and (n_2, r_2) are outcomes of $\mathcal{F}'_s$, and so (n, r) is an outcome of $\mathcal{F}''_s$. But then, since $\mathcal{F}''_s$ is completely reduced by the reductions leading from $\mathcal{N}''$ to $\mathcal{N}_{i+1}$, the outcome (n, r) is removed by some rule in the reduction path between $\mathcal{N}''$ and $\mathcal{N}_{i+1}$, contradicting our assumption that (n, r) is an outcome of $\mathcal{N}_{i+1}$.

(3) (n, r) and (n_2, r_2) have the same target.
By (2) and Lemma 6, n_2 has exactly the same parties as the synchronizer s. Since (n_1, r_1) unconditionally enables n_2, the same holds for n_1. So we have $P_{n_1} = P_{n_2} = P_s$ and $\mathfrak{X}(n_1, a, r_1) = n_2$ for every $a \in P_{n_2}$. By the definition of the shortcut rule, $\mathfrak{X}(n, a, r) = \mathfrak{X}(n_2, a, r_2)$ for every $a \in P_{n_2}$, and we are done.

To finally prove that (n, r) has the same target as some outcome of $\mathcal{N}_i$ we proceed by induction on the number k of times the shortcut rule has been applied between $\mathcal{N}_i$ and $\mathcal{N}'$. If $k = 0$, then (n_2, r_2) is an outcome of $\mathcal{N}_i$, and by (3) we are done. If $k > 0$, then either (n_2, r_2) is an outcome of $\mathcal{N}_i$, and by (3) we are done, or it is produced by a former application of the shortcut rule. In this case, by induction hypothesis, (n_2, r_2) has the same target in $\mathcal{N}_i$ and therefore, by (3), so has (n, r). □