Negotiations and Petri Nets

Jörg Desel[1] and Javier Esparza[2]

[1] Fakultät für Mathematik und Informatik, FernUniversität in Hagen, Germany
joerg.desel@fernuni-hagen.de
[2] Fakultät für Informatik, Technische Universität München, Germany
esparza@in.tum.de

Abstract. Negotiations have recently been introduced as a model of concurrency with multi-party negotiation atoms as primitive. This paper studies the relation between negotiations and Petri nets. In particular, we show that each negotiation can be translated into a 1-safe labelled Petri net with equivalent behaviour. In the general case, this Petri net is exponentially larger than the negotiation. For deterministic negotiations however, the corresponding Petri has linear size compared to the negotiation, and it enjoys the free-choice property. We show that for this class the negotiation is sound if and only if the corresponding Petri net is sound. Finally, we have a look at the converse direction; given a Petri net; can we find a corresponding negotiation?

Keywords: Negotiations, Petri nets, soundness, free-choice nets

1 Introduction

Distributed negotiations have been identified as a paradigm for process interaction since some decades, in particular in the context of multi-agent systems. A distributed negotiation is based on a set of agents that communicate with each other to eventually reach a common decision. It can be viewed as a protocol with atomic negotiations as smallest elements. Multiparty negotiations can employ more than two agents, both in the entire negotiation and in its atoms. A natural way to formally model distributed negotiations is to model the behaviour of the agents separately and then to model the communication between agents by composition of these agent models. Petri nets and related process languages have been used with this aim, see e.g. [2, 8, 7].

In [4, 5] we have introduced a novel approach to formally model negotiations. We argue that this model is sometimes more intuitive than Petri nets for negotiations, but it can also be applied to other application areas which are based on the same communication principles. Like Petri nets, our formalism has a graphical representation. *Atomic negotiations* are represented as nodes, with a specific representation of the participating agents. Roughly speaking, the semantics of a negotiation atom is that these agents, called participants of the atom, come together (and are thus not distributed and do not need any communication means during the atomic negotiation) to agree on one of some possible

outcomes. Given an outcome, the model specifies, for each participating agent, the next possible atomic negotiations in which it can participate. Agents have local states which are only changed when an agent participates in a negotiation. Atomic negotiations are combined into *distributed negotiations*. The state of a distributed negotiation is determined by the atomic negotiations which the agents can participate in next and by all local states. As in Petri nets, these two aspects are carefully distinguished; the current next possible atomic negotiations are represented as *markings* of negotiations.

Our previous contributions [4, 5] concentrate on the analysis of negotiations. In particular, we studied the efficient analysis of well-behavedness of negotiations by means of structural reduction rules. Our work was inspired by known reduction rules of Petri nets but leads to significantly better results when a translation to Petri nets is avoided, at least for the general case. The present paper makes the relation to Petri nets explicit, providing a translation rule from distributed negotiations to Petri nets. It turns out that, for restricted classes of negotiations, the corresponding Petri nets enjoy nice properties, and in this case the converse direction is possible, too.

The paper is organised as follows. Section 2 repeats the syntax and semantics of negotiations. Section 3 provides the translation to Petri nets with the same behaviour. Section 4 discusses properties of these Petri nets. In Section 5 we show that Petri nets enjoying these properties can be translated back to negotiations, this way characterizing a class of Petri nets representable by negotiations.

2 Negotiations: Syntax and Semantics

We recall the main definitions of [4, 5] for syntax and semantics of negotiations. Let A be a finite set (of *agents*), representing potential parties of a negotiation. Each agent $a \in A$ has a (possibly infinite) nonempty set Q_a of *internal states* with a distinguished subset $Q_{0a} \subseteq Q_a$ of *initial states*. We denote by Q_A the cartesian product $\prod_{a \in A} Q_a$. So a state is represented by a tuple $(q_{a_1}, \ldots, q_{a_{|A|}}) \in Q_A$. A *transformer* is a left-total relation $\tau \subseteq Q_A \times Q_A$, representing a nondeterministic state transforming function. Given $S \subseteq A$, we say that a transformer τ is an *S-transformer* if, for each $a_i \notin S$, $((q_{a_1}, \ldots, q_{a_i}, \ldots, q_{a_{|A|}}), (q'_{a_1}, \ldots, q'_{a_i}, \ldots, q'_{a_{|A|}})) \in \tau$ implies $q_{a_i} = q'_{a_i}$. So an S-transformer only transforms internal states of agents in S or in a subset of S.

Internal states of agents and their transformers won't play an important role in this contribution. As will become clear later, states do not influence behaviour in negotiations, i.e., we can consider the control flow and data aspects separately. For the Petri net translation to be defined, local states and their transformers can be modelled by means of token colours and transition modes, respectively, i.e. by means of Coloured Petri nets. These Coloured Petri nets are without guards, because guards restrict transition occurrences by regarding data values.

2.1 Atomic Negotiations

Definition 1. *An* atomic negotiation*, or just an* atom*, over a set of agents A is a triple $n = (P, R, \delta)$, where $P \subseteq A$ is a nonempty set of* parties *or* participants *of n, R is a finite, nonempty set (*results*), and δ is a mapping assigning to each result $r \in R$ a P-transformer $\delta(r)$.*

In the sequel, P_n, R_n and δ_n will denote the components of an atom n. For each result $r \in R_n$, the pair (n, r) is called an *outcome*. The difference between results and outcomes is that the same result can belong to different atoms whereas the sets of outcomes are pairwise disjoint. If we choose disjoint sets for the respective sets of results then we do not have to distinguish results and outcomes.

If the states of the agents before an atomic negotiation n are given by a tuple q and the result of the negotiation is r, then the agents change their states to q' for some $(q, q') \in \delta_n(r)$. Only the parties of n can change their internal states. However, it is not required that a P_n-transformer $\delta_n(r)$ actually changes the states of all agents in P_n. Each result $r \in R_n$ is possible, independent of the previous internal states of the parties of n.

As a simple example, consider an atomic negotiation $n_{\mathtt{FD}}$ 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 (a number between 8 and 12) at which she must return home. The possible results are $\{\mathtt{yes}, \mathtt{no}, \mathtt{ask\_mother}\}$. Both sets $Q_{\mathtt{F}}$ and $Q_{\mathtt{D}}$ contain a state *angry* plus a state t for every time $T_1 \leq t \leq T_2$ in a given interval $[T_1, T_2]$. The transformer $\delta_{n_{\mathtt{FD}}}$ includes

$$\begin{aligned}
\delta_{n_{\mathtt{FD}}}(\mathtt{yes}) &= \{\ ((t_f, t_d), (t, t)) \ \mid\ t_f \leq t \leq t_d \vee t_d \leq t \leq t_f\ \} \\
\delta_{n_{\mathtt{FD}}}(\mathtt{no}) &= \{\ ((t_f, t_d), (\mathit{angry}, \mathit{angry}))\ \} \\
\delta_{n_{\mathtt{FD}}}(\mathtt{ask_mother}) &= \{\ ((t_f, t_d), (t_f, t_d))\ \}
\end{aligned}$$

where t_f and t_d are variables used to denote that `F` is in state $t_f \neq \mathit{angry}$ and `D` in state $t_d \neq \mathit{angry}$ before engaging in the negotiation atom $n_{\mathtt{FD}}$. Moreover, if one of the local states before the negotiation atom was *angry*, then $\delta_{n_{\mathtt{FD}}}$ specifies that both agents will be *angry* after executing the atom.

If both parties are not *angry* and the result is `yes`, then `F` and `D` agree on a time t which is not earlier and not later than both suggested times. If it is `no`, then there is a quarrel and both parties get angry. If it is `ask_mother`, then the parties keep their previous times.

2.2 Combining Atomic Negotiations

If the result of the atomic negotiation above is `ask_mother`, then $n_{\mathtt{FD}}$ is followed by a second atomic negotiation $n_{\mathtt{DM}}$ between `D` and `M` (Mother). The combined negotiation is the composition of $n_{\mathtt{FD}}$ and $n_{\mathtt{DM}}$, where the possible internal states of `M` are the same as those of `F` and `D`, and $n_{\mathtt{DM}}$ is a "copy" of $n_{\mathtt{FD}}$, but without the `ask_mother` result. In order to compose atomic negotiations, 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 a result r of n a set $\mathcal{X}(n, a, r)$ of atoms. Intuitively, this

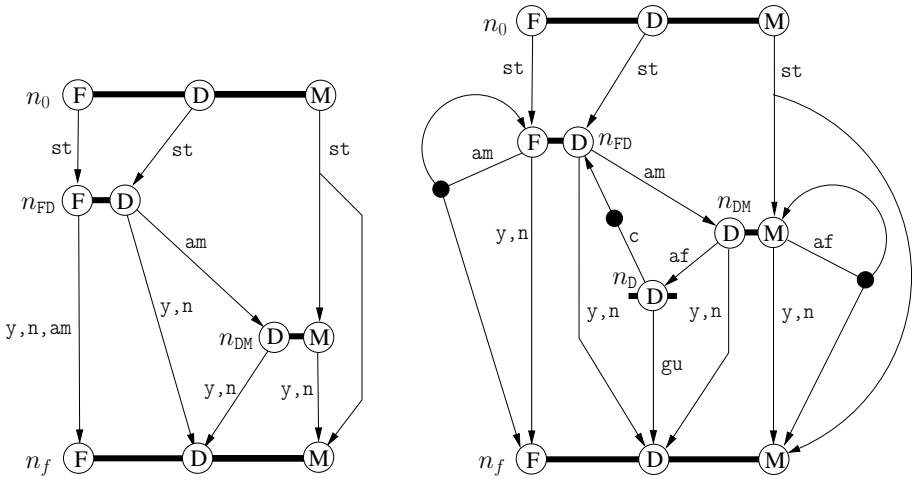

Fig. 1. An acyclic negotiation and the ping-pong negotiation.

is the set of atomic negotiations agent a is ready to engage in after the atom n, if the result of n is r.

Definition 2. *Given a finite set of agents A and a finite set of atoms N over A, 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* (distributed) 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 $n = n_f$.

The graph associated with $\mathcal{N}$ *has 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 initial and final atoms mark the beginning and the end of the negotiation (and sometimes this is their only role). We may have $n_0 = n_f$. In this case, due to (2), $N = \{n_0\}$, i.e, the negotiation has only one single atom. Notice that n_f has, as all other atoms, at least one result $\mathtt{end} \in R_{n_f}$.

2.3 Graphical Representation of Negotiations

Negotiations are graphically represented as shown in Figure 1. For each atom $n \in N$ we draw a bar; for each participant a of P_n we draw a circle on the bar, called a *port*. For each $(n, a, r) \in T(N)$ with $n \neq n_f$, a hyperarc leads from the port of a in n to all the ports of a in the atoms of $\mathcal{X}(n, a, r)$, labelled by the result r. Figure 1 shows on the left the graphical representation of a negotiation where Father (`F`), Daughter (`D`) and Mother (`M`) are the involved agents. After the initial atom n_0, which has only one possible result `st` (`start`), the negotiation atoms

sketched above take place. Notice that possibly Father and Daughter come to an agreement without involving Mother. So the agents of a negotiation can be viewed as potential participants, which necessarily participate only in the initial and the final atom. Instead of multiple (hyper)arcs connecting the same input port to the same output ports we draw a single (hyper)arc with multiple labels. In the figure, we write `y` for `yes`, `n` for `no`, and `am` for `ask mother`. Since n_f has no outgoing arc, the results of n_f do not appear in the graphical representation.

The negotiation on the right (ignore the black dots on the arcs for the moment) is the ping-pong negotiation, well-known in every family. The $n_{\texttt{DM}}$ atom has now an extra result `ask_father` (`af`), and Daughter can be sent back and forth between Mother and Father. After each round, `D` "negotiates with herself" (atom $n_{\texttt{D}}$) with possible outcomes `continue` (`c`) and `give up` (`gu`).

2.4 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 $\boldsymbol{x}_f$, are given by $\boldsymbol{x}_0(a) = \{n_0\}$ and $\boldsymbol{x}_f(a) = \emptyset$ for every $a \in A$. Obviously, the set of markings is finite.

A marking $\boldsymbol{x}$ *enables* an atom n if $n \in \boldsymbol{x}(a)$ for every $a \in P_n$, i.e., if every agent that participates in n is currently ready to engage in n. If $\boldsymbol{x}$ enables n, then n can take place and its participants agree on a result r; we say that the outcome (n, r) *occurs*. 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*.

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 $\boldsymbol{x}_f$, then σ is a *large step*.

As a consequence of this definition, for each agent a, $\boldsymbol{x}(a)$ is always either $\{n_0\}$ or equals $\mathcal{X}(n, a, r)$ for some outcome (n, r). The marking $\boldsymbol{x}_f$ can only be reached by the occurrence of $(n_f, \texttt{end})$ (`end` being a possible result of n_f), and it does not enable any atom.

Reachable markings can be graphically represented by placing tokens (black dots) on the forking points of the hyperarcs (or in the middle of an arc). Thus, both the initial marking and the final marking are represented by no tokens, and all other reachable markings are represented by exactly one token per agent.

Figure 1 shows on the right the marking in which Father is ready to engage in the atomic negotiations $n_{\texttt{FD}}$ and n_f, Daughter is only ready to engage in $n_{\texttt{FD}}$, and Mother is ready to engage in both $n_{\texttt{DM}}$ and n_f.

As mentioned before, the enabledness of an atom does not depend on the internal states of the agents involved; it suffices that all agents are ready to engage in this atom, no matter which internal states they have. Moreover, each result of the atom is possible, independent from the internal states. A given result then determines a state transformer and thus possible next states.

2.5 Reachability Graphs

As known from any family, an occurrence sequence of a negotiation can be arbitrarily long (see the ping-pong negotiation above). Therefore, the set of possible occurrence sequences can be infinite. Since we have markings and steps, an obvious way to describe behaviour with finite means is by reachability graphs:

Definition 3. *The* reachability graph *of a negotiation* $\mathcal{N}$ *has all markings reachable from* $\boldsymbol{x}_0$ *as vertices, and an arc leading from* $\boldsymbol{x}$ *to* $\boldsymbol{x}'$ *and annotated by* (n, r) *whenever* $\boldsymbol{x} \xrightarrow{(n,r)} \boldsymbol{x}'$. *The initial marking* $\boldsymbol{x}_0$ *is the distinguished* initial vertex.

Generally, atoms with disjoint sets of parties can proceed concurrently, whereas atoms sharing a party cannot. Formally, if two outcomes (n_1, r_1) and (n_2, r_2) are enabled by the same reachable marking $\boldsymbol{x}$ and $P_{n_1} \cap P_{n_2} = \emptyset$ then the outcomes can occur concurrently. The condition $P_{n_1} \cap P_{n_2} = \emptyset$ is also necessary for concurrent occurrences of outcomes because, in our model, a single agent cannot be engaged concurrently in two different atoms, and because two state transformers cannot operate concurrently on the local state of an agent. Thus concurrency between outcomes depends only on the involved atoms (and their parties) and not on the results.

Concurrency is formally captured by the *concurrent step reachability graph*, defined next. A *concurrent step* enabled at a reachable marking $\boldsymbol{x}$ is a nonempty set of pairwise concurrent outcomes, each of them enabled by $\boldsymbol{x}$. It is immediate to see that all the outcomes of a concurrent step can be executed subsequently in arbitrary order and that the marking finally reached does not depend on the chosen order. We call this marking *reached by the concurrent step.*

Definition 4. *The* concurrent step reachability graph *of a negotiation* $\mathcal{N}$ *has all markings reachable from* $\boldsymbol{x}_0$ *as vertices. An arc, annotated by a nonempty set of outcomes, leads from* $\boldsymbol{x}$ *to* $\boldsymbol{x}'$ *whenever the outcomes of this set are pairwise concurrent and the concurrent step leads from* $\boldsymbol{x}$ *to* $\boldsymbol{x}'$. *Again,* $\boldsymbol{x}_0$ *is the distinguished* initial vertex.

3 From Negotiations to Petri Nets

We assume that the reader is acquainted with (low-level) initially marked Petri nets, the occurrence rule, reachable markings, liveness, and the graphical representation of nets as directed graphs. For each place, there are directed arcs from all *input transitions* to the place and directed arcs from the place to all *output*

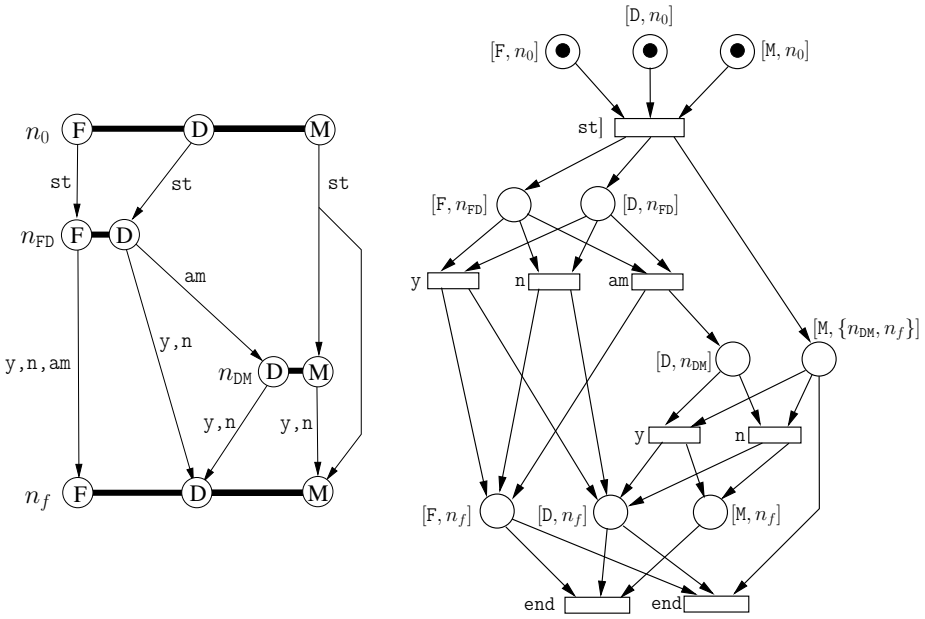

Fig. 2. Petri net semantics of the negotiation of Figure 1

transitions. *Input places* and *output places* of transitions are defined analogously. A *labelled* Petri net is a Petri net with a labelling function λ, mapping transitions to some set of labels. Graphically, the label $\lambda(t)$ of a transition t is depicted as an annotation of t.

3.1 Examples

The semantics of negotiations uses many notions from Petri net theory. In this section, we provide a translation and begin with an example.

Figure 2 shows on the right the net for the negotiation shown on the left (which was also shown in Figure 1). Since the number of places of the net equals the number of ports of the negotiation, one might assume that the relation between ports and places is a simple one-to-one mapping. Moreover, the transitions of the net have an obvious relation to the outcomes, i.e., to the results of the negotiation atoms (if the two `end`-transitions are ignored).

Now we have a look at the two `end`-transitions of the Petri net. The left transition refers to the last result of the negotiation's occurrence sequence $(n_0, \mathtt{st}), (n_{\mathtt{FD}}, \mathtt{am}), (n_{\mathtt{DM}}, \mathtt{y}), (n_f, \mathtt{end})$, where `end` is a result of n_f. The right transition refers to the last result of the occurrence sequence $(n_0, \mathtt{st}), (n_{\mathtt{FD}}, \mathtt{y}), (n_f, \mathtt{end})$. Hence, roughly speaking, the left transition refers to the left branch of the (only) proper hyperarc of the graphical representation of the negotiation, and the right transition refers to the right branch.

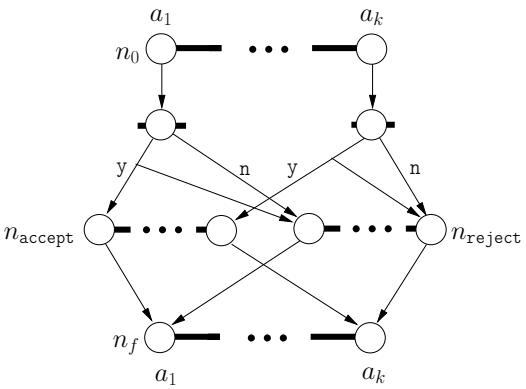

Fig. 3. A (not yet completely correct) negotiation for unanimous vote (all agents participate in all atoms)

For a negotiation with more than one proper hyperarc, each occurrence sequence can involve a particular branching of a hyperarc (moreover, an atom can occur more than once, leading to different branches of the same hyperarc). For k hyperarcs with binary branching, this results in 2^k possible patterns. As can be seen in the following example, this can result in exponentially many transitions of the associated Petri net.

Figure 3 shows a class of negotiations with parameter k, involving agents $a_1, \ldots, a_k$. These negotiations represent a distributed voting process. Each agent votes with possible outcome `yes` or `no` (one-party-negotiations). For each `yes`-outcome there are two possible next atoms, $n_{\texttt{accept}}$ and $n_{\texttt{reject}}$, whereas for each `no`-outcome $n_{\texttt{reject}}$ is the only possibility. So the atom $n_{\texttt{accept}}$ is only enabled if all agents vote `yes`, while the atom $n_{\texttt{reject}}$ is always enabled when all agents have voted.

A Petri net representing this behaviour necessarily has to distinguish the k possible `yes`-outcomes and `no`-outcomes, because final acceptance is only possible if all agents have accepted. So we need $2 \cdot k$ corresponding places, k for acceptance and k for rejection. When all agents came to a result, one of 2^k possible markings is reached. Only for one of these markings (all agents accepted), final acceptance is possible, and this will be represented by one transition. For each of the $2^k - 1$ alternative constellations, we need a separate transition to remove the tokens and come to final rejection. So we end up with 2^k transitions.

3.2 Formal Translation of Negotiations

We associate with a negotiation $\mathcal{N} = (N, n_0, n_f, \mathfrak{X})$ a (labelled) Petri net. The places of this net are, for each atom n except n_f, the pairs $[a, S]$ such that $a \in P_n$, $r \in R_n$, and $\mathfrak{X}(n, a, r) = S$, plus, for each $a \in A$, the pair $[a, \{n_0\}]$. Observe that the number of places is linear in the size of $\mathcal{N}$ (which might exceed $|N|$ significantly, because, for each n in N, for each a in P_n and for each result

$r \in R_n$ we have a set of possible successor negotiations in $\mathfrak{X}$). In the sequel (and in the figures) we write $[a, n]$ instead of $[a, \{n\}]$. The *initial marking* assigns one token to each place $[a, \{n_0\}]$ and no token to all other places.

The net has a set of transitions $T(n, r)$ for each outcome (n, r). An input place of a transition in $T(n, r)$ reflects that a party of negotiation n is actually ready to engage in n (and possibly in other atoms as well). For a single agent, there might be more than one such place, resulting in several transitions. Each transition in $T(n, r)$ has input places referring to all involved parties, which results in a transition for each combination of respective input places.

Formally, let $P_n = \{a_1, \ldots, a_k\}$. $T(n, r)$ contains a transition $[n, r, L]$ for every tuple $L = ([a_1, S_1], \ldots, [a_k, S_k])$ such that $n \in S_1 \cap \ldots \cap S_k$. The set of input places of $[n, r, L]$ is $\{[a_1, S_1], \ldots, [a_k, S_k]\}$, and its set of output places is $\{[a_1, \mathfrak{X}(n, a_1, r)], \ldots, [a_k, \mathfrak{X}(n, a_k, r)]\}$. All transitions of the set $T(n, r)$ are labelled by the outcome (n, r). They all have the same output places. Moreover, they have the same number of input and output places, both of them equal to the number of parties of n.

For the negotiation on the left of Figure 2, we get seven sets of transitions: $T(n_0, \mathtt{st})$, $T(n_{\mathtt{FD}}, \mathtt{y})$, $T(n_{\mathtt{FD}}, \mathtt{n})$, $T(n_{\mathtt{FD}}, \mathtt{am})$, $T(n_{\mathtt{DM}}, \mathtt{y})$, $T(n_{\mathtt{DM}}, \mathtt{n})$, and $T(n_f, \mathtt{end})$. All of them are singletons, with the exception of $T(n_f, \mathtt{end})$, which contains the two transitions shown at the bottom of the figure. In the figure, we annotate transitions only by results r instead of outcomes (n, r). Notice that here we assume a unique result `end` of n_f.

Proposition 1. *For each atom $n \neq n_f$, each transition labelled by (n, r) has exactly one input place $[a, X]$ for each agent $a \in P_n$, and exactly one output place $[a, Y]$ for each agent $a \in P_n$. Transitions labelled by $(n_f, \mathtt{end})$ have no output places.* □

Corollary 1. *For each agent a, the number of tokens on places $[a, X]$ never increases. Since this number is one initially, it is at most one for each reachable marking.* □

Corollary 2. *The net associated with a negotiation is 1-safe, i.e., no reachable marking assigns more than one token to a place.* □

Lemma 1. *The net associated with a negotiation is deterministic, i.e., no reachable marking enables two distinct transitions with the same label.*

Proof. A transition labelled by (n, r) has an input place for each participant of n. Two equally labelled transitions cannot have identical sets of input places by construction. Hence, for at least one agent a there is a place $[a, X]$ which is input place of one of the transitions and a distinct place $[a, Y]$ which is input place of the other transition. Since, by Corollary 1, each reachable marking marks at most one of these two places, each reachable marking enables at most one of the transitions. □

The (sequential) behaviour of a labelled Petri net is represented by its reachability graph:

Definition 5. *The* reachability graph *of a Petri net has all reachable markings m as vertices, an arc annotated by t leading from m to m' when m enables transition t and the occurrence of t leads to m', and a distinguished initial marking m_0. The* label reachability graph *of a labelled Petri net is obtained from its reachability graph by replacing each transition by its label.*

In terms of reachability graphs, a labelled Petri net is deterministic if and only if its label reachability graph has no vertex with two outgoing edges which carry the same label. An occurrence sequence of a deterministic labelled Petri net is fully determined by the sequence of transition labels, as shown in the following proposition, and so is the sequence of markings reached.

For a labelling function λ and an occurrence sequence $\sigma = t_1\ t_2\ t_3 \ldots$, we write $\lambda(\sigma)$ for the sequence of labels $\lambda(t_1)\ \lambda(t_2)\ \lambda(t_3) \ldots$ in the sequel.

Proposition 2. *Let σ_1 and σ_2 be two finite, initially enabled occurrence sequences of a deterministic labelled Petri net with labelling function λ. Let m_1 be the marking reached by σ_1, and let m_2 be the marking reached by σ_2. If $\lambda(\sigma_1) = \lambda(\sigma_2)$ then $m_1 = m_2$.* □

3.3 Behavioural Equivalence between Negotiations and Nets

In this subsection, we will employ the usual notion of isomorphism between reachability graphs:

Definition 6. *Two reachability graphs are isomorphic if there exists a bijective mapping φ between their sets of vertices, mapping the initial vertex of the first graph to the initial vertex of the second graph, such that there is an edge from u to v labelled by some t in the first graph if and only if there is an edge from $\lambda(u)$ to $\lambda(v)$ labelled by t in the second graph.*

Reachability graph isomorphism is a very strong behavioral equivalence notion for sequential behaviour. If moreover the concurrent step reachability graphs of two models are isomorphic, then also the concurrent behaviour of the systems coincide. We will show the existence of both isomorphisms between negotiations and associated Petri nets.

Proposition 3. *The reachability graph of a negotiation and the label reachability graph of the associated labelled Petri net are isomorphic.*

Proof. (Sketch). We interpret a token on a place $[a, \{n_1, \ldots, n_k\}]$ on the negotiation side as "agent a is ready to engage in the atoms of the set $\{n_1, \ldots, n_k\}$". It is immediate to see that this holds initially. By construction of the Petri net, a small step (n, r) of the negotiation is mimicked by an occurrence of a transition of the set $T(n, r)$, and hence by a transition labelled by (n, r). By construction, the marking of the negotiation reached by the occurrence of the outcome corresponds to the marking of the net reached by the occurrence of the transition. □

For comparing the concurrent behaviour of negotiations and associated labelled Petri nets, we have to define concurrent enabledness of transitions. This is easy in our setting, because the considered nets are 1-safe.

Definition 7. *Two transitions t and t' of a 1-safe Petri net are concurrently enabled at a reachable marking m if m enables both t and t' and if moreover t and t' have no common input place.*

Concurrent behaviour is captured by the concurrent step reachability graph and, for labelled Petri nets, by its label version. In the following definition, a set of transitions is said to be *concurrently enabled* if any two distinct transitions in this set are concurrently enabled.

Definition 8. *The* concurrent step reachability graph *of a Petri net has all reachable markings m as vertices, a distinguished initial marking m_0 and an arc labelled by U leading from m to m' when m concurrently enables a nonempty set U of transition and the occurrence of all transitions of U (in any order) leads from m to m'.*

The label concurrent step reachability graph *of a labelled Petri net is obtained from its concurrent step reachability graph by replacing each set of transitions by the multiset of its labels.*

Fortunately, in our setting two equally labelled transitions are never enabled concurrently, so that the labels of concurrent steps will never be proper multisets, but just sets.

Lemma 2. *If two outcomes (n, r) and (n', r') of a negotiation are concurrently enabled at a marking reached by an initial occurrence sequence σ, then there is an initially enabled occurrence sequence μ of the associated labelled Petri net such that $\lambda(\mu) = \sigma$ and the marking reached by μ concurrently enables two transitions labelled by (n, r) and (n', r') respectively.*

Conversely, if a marking of the (λ-)labelled Petri net reached by an occurrence sequence μ concurrently enables two transitions t and t', then the marking of the negotiation reached by $\lambda(\mu)$ concurrently enables the two outcomes $\lambda(t)$ and $\lambda(t')$.

Proof. (Sketch). By construction of the Petri net, a transition t has an input place $[a, X]$ only if $\lambda(t) = (n, r)$ for an agent $a \in P_n$. Assume that two enabled transitions are not concurrent. Then they share an input place $[a, X]$ only if their labels refer to two outcomes (n, r) and (n', r') such that $a \in P_n$ and $a \in P_{n'}$. So $P_n \cap P_{n'} \neq \emptyset$, and thus the two outcomes are not concurrent.

Conversely, if two outcomes (n, r) and (n', r') are enabled but not concurrent, then some agent a belongs to both P_n and $P_{n'}$. In the Petri net, each transition labelled by (n, r) or by (n', r') has an input place $[a, X]$. Since each reachable marking marks only one place $[a, X]$ by Corollary 1, two distinct enabled transitions labelled by (n, r) or by (n', r') share this marked input place, whence they are not concurrent. □

Corollary 3. *The concurrent step reachability graph of a negotiation and the label concurrent step reachability graph of its associated Petri net are isomorphic.* □

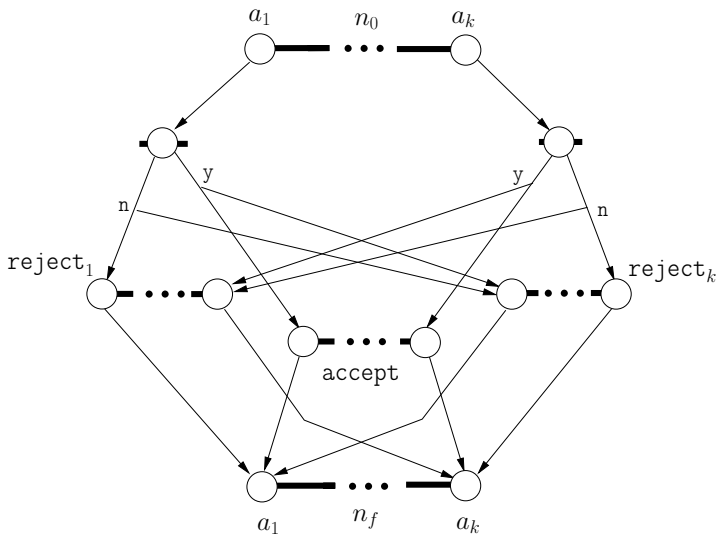

Fig. 4. A corrected negotiation for unanimous vote (all agents participate in all atoms)

3.4 Excursion: On the Voting Example

The reader possibly finds unsatisfactory that the negotiation given in Figure 3 can reject even when all parties vote yes. This results in 2^k respective transitions of the Petri net. If we want to avoid this possibility in the Petri net, we just remove the single transition that removes tokens from all `accept`-places and enables overall rejection. For the negotiation, we found the following work-around: we replace the atom $n_{\texttt{reject}}$ by k rejecting atoms $\texttt{reject}_i$, for $1 \leq i \leq k$. If agent a_i votes yes, then it is ready to engage in `accept` and in all $\texttt{reject}_j$ such that $j \neq i$. Any of the $\texttt{rejecj}_j$-atoms have a single result that leads to final rejection. When all agents vote `yes` then none of the $\texttt{reject}_i$-atoms are enabled, whence only overall acceptance can take place. Notice that this construction is a bit clumsy (see Figure 4), but still does not require exponentially many elements, as the associated Petri net does.

4 Properties of the Net Associated with a Negotiation

4.1 S-components

An *S-component* of a Petri net is a subnet such that, for each place of the subnet, all input- and output-transitions belong to the subnet as well, and such that each transition of the subnet has exactly one input- and exactly one output-place of the subnet. It is immediate to see that the number of tokens in an S-component never changes. A net is covered by S-components if each place and each transition belongs to an S-component. Nets covered by S-components carrying exactly one

token are necessarily 1-safe. For example, every live and 1-safe free-choice net enjoys this nice property [3].

Petri nets associated with negotiations are not covered by S-components, only because the `end`-transitions have no output places. However, if we add an arc from each `end`-transition to each initially marked place, then the resulting net is covered by S-components:

Proposition 4. *The Petri net associated with a negotiation, with additional arcs from each* `end`*-transition to each initially marked place, is covered by S-components.*

Proof. (Sketch). For each agent a, the subnet generated by all places $[a, X]$ and all transitions labelled by (n, r), where $a \in P_n$, is an S-component (being generated implies that the arcs of the subnet are all arcs of the original net connecting nodes of the subnet). An arbitrary place of the net belongs to one such subnet, because it corresponds to an agent. Each transition has a label (n, r), and each atom n has a nonempty set of participants. □

4.2 Soundness

The following notion of sound negotiations was inspired by van der Aalst's soundness of workflow nets [1].

Definition 9. *A negotiation is* sound *if each outcome occurs in some initial occurrence sequence and if, moreover, each finite occurrence sequence is a large step or can be extended to a large step.*

All the negotiations shown in the figures of this paper are sound. For an example of an unsound negotiation, consider again the ping-pong negotiation shown in Figure 1 on the right hand side. Imagine that Daughter could choose to start negotiating with Father or with Mother. This could be expressed by replacing the arc from port `D` of n_0 to port `D` of $n_{\texttt{FD}}$ by a hyperarc from port `D` of n_0 to ports `D` of both $n_{\texttt{FD}}$ and $n_{\texttt{DM}}$. If the first negotiation is between Daughter and Mother, and if it is successful, a marking is reached where both Daughter and Mother can only engage in the final atom n_f, whereas father is still only able to participate in $n_{\texttt{FD}}$. So the distributed negotiation has reached a marking which is neither final nor enables any outcome. We call such a marking a *deadlock*. Clearly, sound negotiations have no reachable deadlocks.

Since the Petri nets associated with negotiations are not workflow nets, we cannot immediately compare the soundness notions of workflow nets and of negotiations. Instead, we first provide a translation of nets associated with negotiations to workflow nets. It turns out that a sound negotiation does not necessarily lead to a sound workflow net in the general case. However, for the subclass of deterministic negotiations the two concepts coincide, as will be shown next.

We begin with a very simple equivalence transformation of nets:

Definition 10. *Two Petri nets N and N' are in the relation $\mathcal{R}$ if*

- *either N has two distinct places with identical sets of input transitions, identical sets of output transitions and equal initial markings, and N' is obtained from N by deletion of one of these places (and adjacent arcs),*
- *or N has a place without output transition, and N' is obtained from N by deletion of this place.*

The symmetrical, reflexive and transitive closure of $\mathcal{R}$ is called place equivalence.

Obviously, two place-equivalent nets have identical behaviour, i.e., their reachability graphs are isomorphic and so are their concurrent step reachability graphs. Notice, however, that place-equivalence does not respect 1-safety. If the only place that violates 1-safety has no output-transition, then deletion of this place can make a net 1-safe.

A *workflow net* is a Petri net with two distinguished places p_{in} and p_{out} such that p_{in} has no input transition, p_{out} has no output transition and, for each place or transition x, there are directed paths from p_{in} to x and from x to p_{out}. The initial marking of a workflow net assigns one token to the place p_{in} and no token to all other places. Workflow nets also have a *final marking*, assigning only one token to p_{out}. A workflow net is *sound* if it has no dead transitions (i.e., each transition is in an initially enabled occurrence sequence) and, moreover, each initially enabled occurrence sequence is a prefix of an occurrence sequence leading to the final marking.

Proposition 5. *The net associated with a sound negotiation is place-equivalent to a workflow net.*

Proof. (Sketch). We derive a single input place p_{in} by deleting all but one of the initially marked places. We add a new place p_{out} with all `end`-transitions as input transitions. Both transformations apparently lead to place equivalence nets.

Since, by soundness of the negotiation, every atom (and therefore every outcome) can be enabled, a token can be moved from the initial atom to any other atom, and therefore there is a directed path from the initial atom to any other atom (more precisely, there is a path in the graph of the negotiation). By the construction of the Petri net, there are according paths from the place p_{in} to arbitrary places and transitions of the net.

Again by soundness of the negotiation, every occurrence sequence can be extended to a large step, i.e., the final atom can eventually be enabled and the final marking reached. So every "token" can be led to the final atom, and therefore there are paths in the graph of the negotiation from every atom to the final atom. By construction of the Petri net, there are thus paths from any element to an `end`-transition, and finally to the new place p_{out}. □

Unfortunately, soundness of a negotiation does not necessarily imply soundness of a related workflow net. The reason is that soundness requires that every atom can occur but not that every branch of a hyperarc is actually used. If,

for example, there would be an additional hyperarc in Figure 1 from the port F in n_0 to the ports F in $n_{\mathtt{FD}}$ and n_f instead of the arc from n_0 to $n_{\mathtt{FD}}$, then the resulting negotiation would still be sound (actually, the behaviour does not change at all). In the associated Petri net, however, there would be an additional transition end with new input place $[\mathtt{F}, \{n_{\mathtt{FD}}, n_f\}]$ (and other input places) which never is enabled. This net is therefore not sound.

4.3 Deterministic Negotiations

In [5], we concentrate on *deterministic negotiations* which are negotiations without proper hyperarcs.

Definition 11. *A negotiation is* deterministic *if, for each atom n, agent $a \in P_n$ and result $r \in R_n$, $\mathcal{X}(n, a, r)$ contains at most one atom (and no atom only if $n = n_f$).*

The term deterministic is justified because there is no choice for an agent with respect to the next possible atom.

Since both, the exponential blow-up and the problem of useless arcs (branches of hyperarcs) stem from proper hyperarcs, we can expect that deterministic negotiations allow for better results. Actually, the Petri net associated with a deterministic negotiation is in fact much smaller, because all its places have the form $[a, X]$, where a is an agent and X is a singleton set of atoms. So the set of places is linear in agents and in atom.

Before discussing soundness of deterministic negotiations, we make a structural observation:

Proposition 6. *The net associated with a deterministic negotiation is a free-choice net, i.e., every two places either share no output transitions, or they share all their output transitions.*

Proof. (Sketch). Since, in nets associated with deterministic negotiations, each place has the form $[a, X]$, where X is a singleton set $\{n\}$, all its output transitions are labelled by (n, r), r being a possible result of n. By construction, every other place $[b, \{n\}]$ has exactly the same output transitions as $[a, \{n\}]$ whereas all other places have no common output transition with $[a, \{n\}]$. □

Proposition 7. *The net associated with a deterministic negotiation is sound if and only if it is place equivalent to a sound workflow net.*

Proof. (Sketch). Observe that the translation from the negotiation to the associated Petri net is much easier in this case: for each atom n we add places $[a, n]$ for each a in P_n and transitions (n, r) for each $r \in R_n$. There are no two transitions for any outcome (n, r), and so transition labels are not necessary (formally, we can label each transition by itself). For each such place $[a, n]$ of an atom n and each such transition (n, r) we add an arc from $[a, n]$ to (n, r). Finally we add arcs from transitions (n, r) to places $[b, n']$ whenever $\mathcal{X}(n, a, r) = \{n'\}$ and $b \in P_{n'}$.

It is immediate that to see this net is free-choice and that the behaviour of the negotiation is precisely mimicked by the net. So the negotiation is sound if and only if the net has no dead transitions and moreover can always reach the final (empty) marking.

The result follows since the net can, as above, be translated into a place equivalent workflow net. □

5 From Nets to Negotiations

In this section we study the converse direction: Given a labelled Petri net, is there a negotiation such that the net is associated with the negotiation? Obviously, for a positive answer the net has to enjoy all the properties derived before. In particular, it must have disjoint S-components and initially marked input places. However, in the general case it appears to be difficult to characterise nets that have corresponding negotiations.

We will provide an answer for the case of sound deterministic negotiations and sound free-choice workflow nets.

Proposition 8. *Every sound free-choice workflow net is place equivalent to a net which is associated with a sound deterministic negotiation.*

Proof. (Sketch). A workflow net is sound if and only if the net with an additional feedback transition moving the token from p_{out} back to p_{in} is live and 1-safe [1]. Live and 1-safe workflow nets are covered by S-components [3]. Therefore a sound workflow net is covered by S-components as well. However, these S-components have not necessarily disjoint sets of places. Consequently, we cannot easily find candidates for agents involved in the negotiation to be constructed.

Instead we proceed as follows: We choose a minimal set of S-components that cover the net. Since each S-component of a live net has to carry a token, all these S-components contain the place p_{in}. Each S-component will be an agent of the net to be constructed, and each conflict cluster (i.e., each maximal set of places together with their common output transitions) a negotiation atom.

Each place p of the net is contained in at least one S-component of the cover. Let C_p be the set of all S-components of the derived minimal cover containing p. If C_p contains more than one S-component, we duplicate the place p, getting a new place p' with input and output transitions like p. Now the new net still has a cover by S-components, where one of the S-components containing p now contains p' instead. Repetition of this procedure eventually leads to a net where each place p belongs to exactly one S-component C_p of the cover. Finally we delete the place p_{out}. Both operations, duplication of places and deletion of p_{out}, lead to place-equivalent nets.

The resulting net is associated with the following negotiation: The set of agents is the set of S-components of the minimal cover. The atoms are the conflict clusters of the net. The results of an atom are the transitions of the corresponding conflict cluster. The $\mathcal{X}$-function can be derived from the arcs of the Petri net leading from transitions to places. □

6 Conclusions

This contribution presented the translation from distributed negotiations to Petri nets such that a negotiation and its associated Petri nets are behaviourally equivalent in a strong sense. In the general case, the Petri net is exponentially larger than the negotiation, wheras for deterministic negotiations its size is only linear. Petri nets do not inherit many properties from arbitrary negotiations, but for deterministic negotiations soundness and non-soundness is respected by the transformation to workflow-like Petri nets, whence in this case the reverse translation is possible as well.

Analysis results for negotiations might be transferable to Petri nets and vice versa via the translation. In future work, we will study this question in particular for the respective sound and complete sets of reduction rules for negotiations [4, 5] and for free-choice Petri nets [3].

Based on the reduction results, a very recent work [6] introduces a global specification language for negotiations and characterises the negotiations expressible with this language. Similar results for Petri nets could be derived via the translation procedure of this paper.

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. Chen, Y., Peng, Y., Finin, T., Labrou, Y., Chu, B., Yao, J., Sun, R., Willhelm, B., Cost, S.: A negotiation-based multi-agent system for supply chain management. In: In Proceedings of Agents 99 - Workshop on Agent Based Decision-Support for Managing the Internet-Enabled Supply-Chain. pp. 15–20 (1999)
3. Desel, J., Esparza, J.: Free Choice Petri Nets. Cambridge University Press, New York, NY, USA (1995)
4. Esparza, J., Desel, J.: On negotiation as concurrency primitive. In: D'Argenio, P.R., Melgratti, H.C. (eds.) CONCUR. Lecture Notes in Computer Science, vol. 8052, pp. 440–454. Springer (2013), extended version in arXiv:1307.2145, http://arxiv.org/abs/1403.4958
5. Esparza, J., Desel, J.: On negotiation as concurrency primitive II: Deterministic cyclic negotiations. In: Muscholl, A. (ed.) FoSSaCS. Lecture Notes in Computer Science, vol. 8412, pp. 258–273. Springer (2014), extended version in CoRR abs/1403.4958, http://arxiv.org/abs/1403.4958
6. Esparza, J., Desel, J.: Negotiation programs. In: Devillers, R., Valmari, A. (eds.) Petri Nets. Lecture Notes in Computer Science, vol. 9115, pp. 157–178. Springer (2015)
7. Simon, C.: Negotiation Processes – The Semantic Process Language and Applications. Shaker, Aachen, Germany (2008)
8. Xu, H., Shatz, S.M.: An agent-based Petri net model with application to seller/buyer design in electronic commerce. In: Fifth International Symposium on Autonomous Decentralized Systems, ISADS 2001, Dallas, Texas, USA, March 26-28, 2001. pp. 11–18. IEEE Computer Society (2001)