Soundness in negotiations.

J. Esparza\textsuperscript{1} D. Kuperberg\textsuperscript{1,2,3} A. Muscholl\textsuperscript{1,2,4} I. Walukiewicz\textsuperscript{1,4}

\textsuperscript{1}TU Munich, \textsuperscript{2}IAS, \textsuperscript{3}ENS Lyon, \textsuperscript{4}LaBRI,CNRS

Séminaire 68NQRT, IRISA, Rennes
12/01/2017
Introduction

Negotiations [Desel, Esparza ’13] CONCUR

- model multiparty distributed cooperation,
- better complexity than alternative models (Petri Nets),
- embeds natural concepts: soundness, race properties,...

This paper [EKMW ’16] CONCUR:

- study of different restrictions on the model,
- complexity of deciding soundness, concurrency relationships
- application to workflow analysis for programs
An atomic negotiation or node involves a set of processes (participants) and has a set of possible outcomes.
Negotiations

- An atomic negotiation or node involves a set of processes (participants) and has a set of possible outcomes.
- If all participants are ready to engage in the node (synchronization), then the node can be fired: the processes agree on one of the outcomes (choice) and move on.
An atomic negotiation or node involves a set of processes (participants) and has a set of possible outcomes.

If all participants are ready to engage in the node (synchronization), then the node can be fired: the processes agree on one of the outcomes (choice) and move on.

A negotiation $\mathcal{N}$ consists of
Negotiations

- An atomic negotiation or node involves a set of processes (participants) and has a set of possible outcomes.
- If all participants are ready to engage in the node (synchronization), then the node can be fired: the processes agree on one of the outcomes (choice) and move on.
- A negotiation $\mathcal{N}$ consists of
  - a set of processes $\text{Proc}$,
An atomic negotiation or node involves a set of processes (participants) and has a set of possible outcomes.

If all participants are ready to engage in the node (synchronization), then the node can be fired: the processes agree on one of the outcomes (choice) and move on.

A negotiation $\mathcal{N}$ consists of
- a set of processes $\text{Proc}$,
- a set of outcomes $R$. 


An atomic negotiation or node involves a set of processes (participants) and has a set of possible outcomes. If all participants are ready to engage in the node (synchronization), then the node can be fired: the processes agree on one of the outcomes (choice) and move on.

A negotiation $\mathcal{N}$ consists of
- a set of processes $\text{Proc}$,
- a set of outcomes $R$,
- a set of nodes $N$ with two distinguished initial and final nodes,
An atomic negotiation or node involves a set of processes (participants) and has a set of possible outcomes.

If all participants are ready to engage in the node (synchronization), then the node can be fired: the processes agree on one of the outcomes (choice) and move on.

A negotiation $N$ consists of

- a set of processes $\text{Proc}$,
- a set of outcomes $R$,
- a set of nodes $N$ with two distinguished initial and final nodes,
- a domain function $\text{dom} : N \rightarrow \mathcal{P}(\text{Proc})$ assigning to each node a set of participants,
Negotiations

- An atomic negotiation or node involves a set of processes (participants) and has a set of possible outcomes.
- If all participants are ready to engage in the node (synchronization), then the node can be fired: the processes agree on one of the outcomes (choice) and move on.
- A negotiation \( \mathcal{N} \) consists of
  - a set of processes \( \text{Proc} \),
  - a set of outcomes \( R \),
  - a set of nodes \( N \) with two distinguished initial and final nodes,
  - a domain function \( \text{dom}: N \to \mathcal{P}(\text{Proc}) \) assigning to each node a set of participants,
  - a transition table \( \delta: N \times R \times \text{Proc} \to \mathcal{P}(N) \)
    \( \delta(n, a, p) = \{n', n''\} \) means: if the participants of \( n \) choose \( a \), then \( p \) is ready to engage in \( n' \) or \( n'' \).
Run of a negotiation

$n_{\text{init}}$ initial node, $n_{\text{fin}}$ final node.
Here: 3 processes $p_1, p_2, p_3$ and only one action $a$. 

$$
\delta(n_{\text{init}}, a, p_2) = \{n_1, n_3\}
$$

$p_2$ is non-deterministic, while $p_1$ and $p_3$ are deterministic.
Run of a negotiation

$n_{init}$ initial node, $n_{fin}$ final node. Here: 3 processes $p_1, p_2, p_3$ and only one action $a$.

$$\delta(n_{init}, a, p_2) = \{n_1, n_3\}$$
Run of a negotiation

$n_{init}$ initial node, $n_{fin}$ final node.
Here: 3 processes $p_1, p_2, p_3$ and only one action $a$.

$$\delta(n_{init}, a, p_2) = \{n_1, n_3\}$$

$$\delta(n_1, a, p_2) = \{n_2, n_{fin}\}$$
Run of a negotiation

\( n_{init} \) initial node, \( n_{fin} \) final node.
Here: 3 processes \( p_1, p_2, p_3 \) and only one action \( a \).

\[
\delta(n_{init}, a, p_2) = \{ n_1, n_3 \}
\]
\[
\delta(n_1, a, p_2) = \{ n_2, n_{fin} \}
\]
\[
\delta(n_2, a, p_2) = \{ n_1, n_3 \}
\]
Run of a negotiation

\( n_{init} \) initial node, \( n_{fin} \) final node.
Here: 3 processes \( p_1, p_2, p_3 \) and only one action \( a \).

\[
\delta(n_{init}, a, p_2) = \{n_1, n_3\}
\]

\[
\delta(n_1, a, p_2) = \{n_2, n_{fin}\}
\]

\[
\delta(n_2, a, p_2) = \{n_1, n_3\}
\]

\[
\delta(n_3, a, p_2) = \{n_2, n_{fin}\}
\]

\( p_2 \) is non-deterministic, while \( p_1 \) and \( p_3 \) are deterministic.
Run of a negotiation

\( n_{\text{init}} \) initial node, \( n_{\text{fin}} \) final node.
Here: 3 processes \( p_1, p_2, p_3 \) and only one action \( a \).

\[
\delta(n_{\text{init}}, a, p_2) = \{n_1, n_3\}
\]
\[
\delta(n_1, a, p_2) = \{n_2, n_{\text{fin}}\}
\]
\[
\delta(n_2, a, p_2) = \{n_1, n_3\}
\]
\[
\delta(n_3, a, p_2) = \{n_2, n_{\text{fin}}\}
\]

\( p_2 \) is non-deterministic, while \( p_1 \) and \( p_3 \) are deterministic.
Deterministic and weakly non-deterministic negotiations

- **Deterministic negotiations**: All processes are deterministic.
Deterministic and weakly non-deterministic negotiations

- **Deterministic negotiations**: All processes are deterministic.
- **Weakly non-deterministic negotiations**: Each node involves at least one deterministic process.
- **Acyclic**: No cycle in the transition graph between nodes.
Deterministic and weakly non-deterministic negotiations

- **Deterministic negotiations**: All processes are deterministic.
- **Weakly non-deterministic negotiations**: Each node involves at least one deterministic process.
- **Acyclic**: No cycle in the transition graph between nodes.

**Intuition** for weakly non-deterministic negotiations:

The negotiation is guided by the deterministic processes. Non-deterministic processes are “told” where to go by the deterministic ones.
Deterministic and weakly non-deterministic negotiations

- **Deterministic negotiations**: All processes are deterministic.
- **Weakly non-deterministic negotiations**: Each node involves at least one deterministic process.
- **Acyclic**: No cycle in the transition graph between nodes.

**Intuition** for weakly non-deterministic negotiations:

The negotiation is *guided* by the deterministic processes. Non-deterministic processes are “told” where to go by the deterministic ones.

**Research program**: investigate the complexity of analysis problems for deterministic and weakly non-deterministic negotiations.
The Soundness problem
**Soundness**:
Every partial run can be completed into an accepting run. Non-blocking property, witnessing good design.

**Example**: Previous negotiation is sound.
Soundness property

**Soundness:**
Every partial run can be completed into an accepting run. Non-blocking property, witnessing good design.

**Example:** Previous negotiation is sound.

**Aim:** Understand the fine-grained complexity of the following problem, depending on restrictions on $\mathcal{N}$:

**INPUT:** A negotiation $\mathcal{N} = (N, Proc, R, \delta)$.

**OUTPUT:** Is $\mathcal{N}$ sound?
Subclasses of negotiations

Soundness problem **PSPACE-complete** in general [DE ’13].

Complexity of the soundness problem for *classes of negotiations*?

**Theorem (DE ’14)**

*Deciding soundness is in PTIME for deterministic negotiations.*

**This paper** [EKMW ’16]: explores the room between the two.
Theorem (EKMW ’16)

Deciding soundness is in PTIME for acyclic weakly non-deterministic negotiations.

Main tool used in the proof: the Omitting Theorem.

Theorem (EKMW ’16)

It can be decided in PTIME if for a given deterministic, acyclic, and sound negotiation \( N \) and two sets \( P \subseteq N \times \mathbb{R} \) and \( B \subseteq N \), there is a successful run of \( N \) containing \( P \) and omitting \( B \).

General interest: characterize the important parts of a negotiation.
What happens if we drop restrictions in the previous results? Dropping weak non-determinism:

Theorem (EKMW ’16)

*The soundness problem for acyclic negotiations is coNP-complete.*
What happens if we drop restrictions in the previous results?

Dropping **weak non-determinism**:

*Theorem (EKMW ’16)*

The soundness problem for *acyclic negotiations* is coNP-complete.

Dropping **acyclicity** for a milder constraint:

*Theorem (EKMW ’16)*

The soundness problem for **det-acyclic weakly non-deterministic negotiations** is coNP-complete.

**Det-acyclicity**: deterministic processes are acyclic.

Enough here to prevent cycles in actual runs.
Det-acyclicity example

**Det-acyclicity** + Weak ND $\implies$ no cycles in runs,
Here not weakly ND.

\[
\begin{align*}
\delta(n_{\text{init}}, a, p_2) &= \{n_1, n_3\} \\
\delta(n_1, a, p_2) &= \{n_2\} \\
\delta(n_2, a, p_2) &= \{n_1, n_3\} \\
\delta(n_3, a, p_2) &= \{n_2\}
\end{align*}
\]
**Det-acyclicity example**

**Det-acyclicity** + Weak ND $\implies$ no cycles in runs,
Here not weakly ND.

$$\delta(n_{init}, a, p_2) = \{n_1, n_3\}$$
Det-acyclicity example

**Det-acyclicity** + Weak ND $\implies$ no cycles in runs,
Here not weakly ND.

\[
\delta(n_{\text{init}}, a, p_2) = \{n_1, n_3\}
\]

\[
\delta(n_1, a, p_2) = \{n_2, n_{\text{fin}}\}
\]
**Det-acyclicity** + Weak ND $\implies$ no cycles in runs,
Here not weakly ND.

\[ \delta(n_{init}, a, p_2) = \{n_1, n_3\} \]
\[ \delta(n_1, a, p_2) = \{n_2, n_{fin}\} \]
\[ \delta(n_2, a, p_2) = \{n_1, n_3\} \]
**Det-acyclicity example**

**Det-acyclicity** $+$ Weak ND $\implies$ no cycles in runs, Here not weakly ND.

\[
\begin{align*}
\delta(n_{init}, a, p_2) &= \{n_1, n_3\} \\
\delta(n_1, a, p_2) &= \{n_2, n_{\text{fin}}\} \\
\delta(n_2, a, p_2) &= \{n_1, n_3\} \\
\delta(n_3, a, p_2) &= \{n_2, n_{\text{fin}}\}
\end{align*}
\]
Det-acyclicity example

**Det-acyclicity** + Weak ND $\implies$ no cycles in runs,
Here not weakly ND.

$$
\delta(n_{\text{init}}, a, p_2) = \{n_1, n_3\}
$$

$$
\delta(n_1, a, p_2) = \{n_2, n_{\text{fin}}\}
$$

$$
\delta(n_2, a, p_2) = \{n_1, n_3\}
$$

$$
\delta(n_3, a, p_2) = \{n_2, n_{\text{fin}}\}
$$
Race Problem:

GIVEN: a negotiation $\mathcal{N}$, and two nodes $n, m$ of $\mathcal{N}$.
DECIDE: can $n$ and $m$ be concurrently enabled?
Race Problem:
**GIVEN**: a negotiation $\mathcal{N}$, and two nodes $n, m$ of $\mathcal{N}$.
**DECIDE**: can $n$ and $m$ be concurrently enabled?

- standard question for concurrent systems
- inherently concurrent property, hard to work with linearizations
The race problem is PSPACE-complete in the general case and NP-complete for acyclic negotiations.

Determinism alone does not help:

**Theorem (EKMW ’16)**

The race problem stays PSPACE-complete and NP-complete for deterministic and acyclic deterministic negotiations.
The race problem is PSPACE-complete in the general case and NP-complete for acyclic negotiations.

Determinism alone does not help:

**Theorem (EKMW '16)**

The race problem stays PSPACE-complete and NP-complete for deterministic and acyclic deterministic negotiations.

But determinism **and** soundness **together** help:

**Theorem (EKMW '16)**

The race problem is

- in \( PTIME \) for sound deterministic negotiations.
- NLOGSPACE-complete for sound deterministic acyclic negotiations,
Application of negotiations: analyze the workflow of programs.  

**Data-flows:** Petri-net-based modeling notation, widely used (Protos) [van der Aalst et al, ’09].

Model the behaviour of programs or protocols acting on global variables via operations: $\text{alloc}(x), \text{read}(x), \text{write}(x), \text{dealloc}(x)$. 
Sound acyclic deterministic negotiations with variables

\[\Rightarrow\text{formalize data-flow problems from}\ [\text{van der Aalst et al, ’09}]:\]

- **Well-defined behaviour**: no concurrent operations on the same variable,
- **No redundancy**: allocated variables are used,
- **Clean memory**: allocated variables are deallocated.
Sound acyclic deterministic negotiations with variables
formalize data-flow problems from [van der Aalst et al, '09]:

- **Well-defined behaviour**: no concurrent operations on the same variable,
- **No redundancy**: allocated variables are used,
- **Clean memory**: allocated variables are deallocated.

**Theorem (EKMW '16)**

*All these properties can be checked in PTIME on data-flows.*

Proof using the Omitting Theorem.

Exponential improvement on [van der Aalst et al, '09].
Data-flow result: Embeds typical specifications into a class of trace properties easy to decide (PTIME).

All these properties can be described by fixed-size automata with simple structure.

Can we generalize this?
Data-flow result: Embeds typical specifications into a class of trace properties easy to decide (PTIME).

All these properties can be described by fixed-size automata with simple structure.

Can we generalize this?

Theorem (Unpublished)

There is a property $P$ specified by a 6-state automaton such that checking $P$ for a sound acyclic deterministic negotiation is NP-complete.

We get a fine-grained description of the difficulty of trace property checking.
Conclusion

- **The negotiation model provides new insights** on what makes communicating finite-state processes hard to analyze.
- **Soundness is a key property** that can decrease the complexity of checking other properties.
- **Detailed picture** of complexities of property checking.
- **Applications** to the static analysis of workflow processes.
Conclusion

- The negotiation model provides new insights on what makes communicating finite-state processes hard to analyze.
- Soundness is a key property that can decrease the complexity of checking other properties.
- Detailed picture of complexities of property checking.
- Applications to the static analysis of workflow processes.

Open problem
Soundness for Weakly non-deterministic negotiations: PSPACE-complete? coNP-complete?.