Soundness in negotiations.

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

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

Automata, Logic, and Games
\textit{Communicating, Distributed and Parameterized Systems}
Singapore 22/08/2016
Negotiations [Desel, Esparza ’13]

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

This paper:

- study of different restrictions on the model,
- complexity of deciding soundness, concurrency relationships
- application to workflow analysis for programs
The negotiation model

Negotiations involve a set of processes, which must decide on outcomes according to a fixed structure. The model builds on the notion of atomic negotiation or node.

\[
\begin{array}{ccccc}
p_1 & p_2 & p_3 & p_4 & p_5 \\
n: & & & & \\
\end{array}
\]

This node \( n \) involves 5 processes \( p_1, \ldots, p_5 \). If all five are ready to engage, the node can be fired: the processes agree on an outcome and move on.
The negotiation model

Negotiations involve a set of **processes**, which must decide on **outcomes** according to a fixed structure. The model builds on the notion of **atomic negotiation** or **node**.

\[
\begin{array}{cccccc}
p_1 & p_2 & p_3 & p_4 & p_5 \\
n: & & & & & \\
\end{array}
\]

This node \( n \) involves 5 processes \( p_1, \ldots, p_5 \). If all five are ready to engage, the node can be **fired**: the processes agree on an **outcome** and move on.

A **negotiation** \( \mathcal{N} \) consists of

- a set of processes \( \text{Proc} \),
- a set of nodes \( \mathcal{N} \),
- a domain function \( \text{dom} : \mathcal{N} \rightarrow \mathcal{P}(\text{Proc}) \),
- a set of outcomes \( \mathcal{R} \),
- a transition table \( \delta : \mathcal{N} \times \mathcal{R} \times \text{Proc} \rightarrow \mathcal{P}(\mathcal{N}) \).
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 $a$. 

$p_2$ 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 $a$.

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

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

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

\[
\delta(n_3, a, p_2) = \{n_2, n_{fin}\}
\]
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.
The Soundness problem
Soundness property

**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:**
**INPUT:** A negotiation $\mathcal{N} = (\mathcal{N}, \text{Proc}, R, \delta)$.
**OUTPUT:** Is $\mathcal{N}$ 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:**
**INPUT:** A negotiation \( \mathcal{N} = (N, \text{Proc}, R, \delta) \).
**OUTPUT:** Is \( \mathcal{N} \) sound?

**Problem:**
*Configuration:* \( \text{Proc} \rightarrow \mathcal{P}(N) \)
\rightarrow Number of configurations exponential in \( |\mathcal{N}| \)
\rightarrow Runs can have exponential length.
Subclasses of negotiations

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

Complexity of the soundness problem for classes of negotiations?

Natural Restrictions on negotiations:

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

Soundness problem \textit{PSPACE-complete} in general \cite{DE '13}.

Complexity of the soundness problem for \textit{classes of negotiations}?

Natural \textbf{Restrictions} on negotiations:

- \textbf{Deterministic}: All processes are deterministic.
- \textbf{Weakly non-deterministic}: All nodes involve at least one deterministic process.
- \textbf{Acyclic}: No cycle in the transition graph between nodes.

\textbf{Theorem (DE ‘14)}

\textit{Deciding soundness is in PTIME for deterministic negotiations.}
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 \( \mathcal{N} \) and two sets \( P \subseteq N \times R \) and \( B \subseteq N \), there is a successful run of \( \mathcal{N} \) containing \( P \) and omitting \( B \).*

**Proof**: Via a game argument.

**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.*
Soundness problem for bigger classes

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 (very) weakly non-deterministic negotiations is coNP-complete.*

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

In this context, it is enough to prevent cycles in actual runs.
Applications of sound negotiations
Race Property

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

- standard question for concurrent systems
- used for guaranteeing predictable behaviours
- inherently parallel property, hard to work with linearizations
**Race Property**

**Race Problem:**

**INPUT:** a sound negotiation $\mathcal{N}$, and two nodes $n$, $m$ of $\mathcal{N}$.

**OUTPUT:** can $n$ and $m$ be concurrently enabled?

- standard question for concurrent systems
- used for guaranteeing predictable behaviours
- inherently parallel property, hard to work with linearizations

**Theorem (EKMW ’16)**

*The race problem is*

- $\text{NLOGSPACE}$-complete for deterministic acyclic negotiations,
- in $\text{PTIME}$ for deterministic negotiations.
Application of negotiations: analyze the workflow of programs. We add global variables that can be affected by nodes via operations: $alloc(x)$, $read(x)$, $write(x)$, $dealloc(x)$.

Acyclic deterministic negotiations with variables $\rightsquigarrow$ formalize data-flow problems from the literature [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.
Application of negotiations: analyze the workflow of programs. We add global variables that can be affected by nodes via operations: $alloc(x)$, $read(x)$, $write(x)$, $dealloc(x)$.

Acyclic deterministic negotiations with variables $\rightsquigarrow$ formalize data-flow problems from the literature [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.*

Exponential improvement on [van der Aalst et al, '09]. Proof using the Omitting Theorem.
Conclusion

**Soundness problem** for negotiations:
- PTIME for acyclic weakly non-deterministic
- coNP-complete for mild relaxations

**Race problem** for sound negotiations:
- NLOGSPACE-complete for deterministic acyclic,
- PTIME for deterministic.

**Data-flow analysis:**
- modelisation with deterministic acyclic negotiations,
- PTIME algorithms for standard problems on data-flows.

**Omitting problem** for sound negotiations
- PTIME for deterministic acyclic negotiations
- used for **Soundness problem** and **Data-flow analysis**.