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

Highlights of Automata
Bruxelles 08/09/2016
Introduction

**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
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 $a$. 
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 \( a \).

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

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

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

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

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

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

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.

**Theorem (DE ’14)**

*Deciding soundness is in PTIME for deterministic negotiations.*
Results on the complexity of the soundness problem

Theorem (EKMW '16)

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

Use of the Omitting Theorem: we can decide in PTIME which parts of a negotiation are essential.
Theorem (EKMW ‘16)

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

Use of the Omitting Theorem: we can decide in PTIME which parts of a negotiation are essential.

Mildly relaxing acyclicity or weak non-determinism: Soundness problem becomes coNP-complete.

Det-acyclicity: only deterministic processes are acyclic. No cycles in runs of weakly ND negotiations.
Applications of sound negotiations
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

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

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 \( \leadsto \) 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].
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.

Good Expressivity/Complexity ratio, well-suited for practical use.