Towards a Regular Theory of Parameterized Concurrent Systems

Benedikt Bollig

Laboratoire Spécification et Vérification
ENS Cachan & CNRS, France

Reports on joint works with Paul Gastin, Akshay Kumar, and Jana Schubert.

ACTS 2015
Chennai Mathematical Institute
The verification problem for parameterized systems:

«Is a system correct independently of the number of processes / the way they are arranged?»

Talks by Arnaud Sangnier and Pierre Ganty.
The verification problem for parameterized systems:

«Is a system correct independently of the number of processes / the way they are arranged?»

Talks by Arnaud Sangnier and Pierre Ganty.

In this talk, we study language-theoretic questions / expressiveness:

- Complementation
- Equivalent characterization in terms of MSO logic
- Nonemptiness

We are looking for «robust» models of parameterized systems.
The verification problem for parameterized systems:

«Is a system correct independently of the number of processes / the way they are arranged?»

Talks by Arnaud Sangnier and Pierre Ganty.

In this talk, we study language-theoretic questions / expressiveness:

- Complementation
- Equivalent characterization in terms of MSO logic
- Nonemptiness

We are looking for «robust» models of parameterized systems.

There have been robust models for fixed process architectures:

Finite Automata

finite automaton

A finite automaton is a mathematical model of computation that consists of a set of states, input symbols, and transitions between states. The diagram above illustrates a finite automaton with states $s_0, s_1, s_2, s_3, s_4, s_5, s_6$, where the transitions are labeled with input symbols $a$ and $b$. The automaton accepts strings based on the path through the states as per the input sequence.
Finite Automata

finite automaton

determinization
Theorem [Büchi-Elgot-Trakhtenbrot 1960s]:
Finite Automata = MSO

∀x(a(x) → ∃y(succ(x, y) ∧ b(y)))
**Finite Automata**

finite automaton

determinization

complementation

**Theorem [Büchi-Elgot-Trakhtenbrot 1960s]:**
Finite Automata = MSO

∀x(a(x) → ∃y(succ(x, y) ∧ b(y)))

**Proof:**
- free variables
  → extended alphabet
Finite Automata

finite automaton

determinization

complementation

**Theorem** [Büchi-Elgot-Trakhtenbrot 1960s]:
Finite Automata = MSO

∀x(a(x) → ∃y(succ(x, y) ∧ b(y)))

**Proof:**
- free variables ➔ extended alphabet
- existential quantification ➔ projection
Finite Automata

finite automaton

\[
\begin{align*}
&\text{determinization} \\
&\text{complementation}
\end{align*}
\]

**Theorem** [Büchi-Elgot-Trakhtenbrot 1960s]:
Finite Automata = MSO

\[
\forall x (a(x) \rightarrow \exists y (\text{succ}(x, y) \land b(y)))
\]

**Proof:**
- free variables → extended alphabet
- existential quantification → projection
- negation → complementation
Theorem [Büchi-Elgot-Trakhtenbrot 1960s]:
Finite Automata = MSO

∀x(a(x) → ∃y(succ(x, y) ∧ b(y)))

Proof:
- free variables
- extended alphabet
- existential quantification
- projection
- negation
- complementation
Parameterized Communicating Automata (PCA)
Parameterized Communicating Automata (PCA)
Parameterized Communicating Automata (PCA)
Parameterized Communicating Automata (PCA)
Parameterized Communicating Automata (PCA)

non-fixed & unbounded
Parameterized Communicating Automata (PCA) over Rings

non-fixed & unbounded
A PCA is given by:

- finite automaton over \( \{l, r\} \times \{!, ?, \} \times \text{Msg} \) \hspace{2cm} (here: \( \text{Msg} = \{0, 1\} \))
- acceptance condition
Parameterized Communicating Automata (PCA) over Rings

A PCA is given by:

- finite automaton over \( \{l, r\} \times \{!, ?\} \times \text{Msg} \) (here: \( \text{Msg} = \{0, 1\} \))
- acceptance condition
A PCA is given by:

- finite automaton over \( \{l, r\} \times \{!, ?\} \times \text{Msg} \)  
  (here: \( \text{Msg} = \{0, 1\} \))
- acceptance condition
A PCA is given by:

- finite automaton over \( \{l, r\} \times \{!, ?\} \times \text{Msg} \) (here: \( \text{Msg} = \{0, 1\} \))
- acceptance condition
Parameterized Communicating Automata (PCA) over Rings

A PCA is given by:

- finite automaton over \( \{l, r\} \times \{!, ?\} \times Msg \)  
  \( \text{(here: } Msg = \{0, 1\}) \)
- acceptance condition
Parameterized Communicating Automata (PCA) over Rings

rendez-vous
Parameterized Communicating Automata (PCA) over Rings

Remark:
Behavior abstracts away message contents from $Msg = \{0, 1\}$ (like states, or stack symbols in pushdown automata).
Parameterized Communicating Automata (PCA) over Rings
Parameterized Communicating Automata (PCA) over Rings
Parameterized Communicating Automata (PCA) over Rings

- States: $s_0, s_1, s_2, s_3, s_4, s_5, s_6$
- Transitions:
  - $s_0 \xrightarrow{l} s_1$
  - $s_0 \xrightarrow{l} s_2$
  - $s_0 \xrightarrow{r} s_3$
  - $s_1 \xrightarrow{l} s_4$
  - $s_1 \xrightarrow{r} s_5$
  - $s_2 \xrightarrow{l} s_4$
  - $s_2 \xrightarrow{r} s_5$
  - $s_3 \xrightarrow{l} s_6$
  - $s_4 \xrightarrow{l} s_5$
  - $s_4 \xrightarrow{r} s_6$
  - $s_5 \xrightarrow{l} s_6$
  - $s_5 \xrightarrow{r} s_6$
  - $s_6 \xrightarrow{l} s_6$
  - $s_6 \xrightarrow{r} s_6$

Transitions marked with $l$ are left transitions, and those marked with $r$ are right transitions.
Parameterized Communicating Automata (PCA) over Rings
Parameterized Communicating Automata (PCA) over Rings
Parameterized Communicating Automata (PCA) over Rings
Parameterized Communicating Automata (PCA) over Rings
Parameterized Communicating Automata (PCA) over Rings
Parameterized Communicating Automata (PCA) over Rings
Parameterized Communicating Automata (PCA) over Rings
Parameterized Communicating Automata (PCA) over Rings
Parameterized Communicating Automata (PCA) over Rings

Acceptance condition:
MSO formula over rings whose nodes are labeled with states.
Signature: \( s(x) \xrightarrow{r,l} y \)

Thus, there are no constant processes (e.g., no «first» or «last» process).
Parameterized Communicating Automata (PCA) over Rings

\[ \exists x (s_4(x) \land \forall y (y \neq x \rightarrow s_5(y) \lor s_6(y))) \]
Parameterized Communicating Automata (PCA) over Rings

\[ \exists x(s_4(x) \land \forall y(y \neq x \rightarrow s_5(y) \lor s_6(y))) \]
Parameterized Communicating Automata (PCA) over Rings

\[ \exists x (s_4(x) \land \forall y (y \neq x \rightarrow s_5(y) \lor s_6(y))) \]

Token-Ring Protocol
Parameterized Communicating Automata (PCA) over Rings

\[ \exists x(s_4(x) \land \forall y(y \neq x \rightarrow s_5(y) \lor s_6(y))) \]
Parameterized Communicating Automata (PCA) over Rings

\[ \exists x(s_4(x) \land \forall y(y \neq x \rightarrow s_5(y) \lor s_6(y))) \]
Parameterized Communicating Automata (PCA) over Rings

\[ \exists x(s_4(x) \land \forall y(y \neq x \rightarrow s_5(y) \lor s_6(y))) \]
Parameterized Communicating Automata (PCA) over Rings

\[
L = \{ \exists x(s_4(x) \land \forall y(y \neq x \rightarrow s_5(y) \lor s_6(y))) \}
\]
Parameterized Communicating Automata (PCA) over Rings

\[ L = \{ \text{agraph} \} \]
Parameterized Communicating Automata (PCA) over Rings

\[ L = \exists x(s_4(x) \land \forall y(y \neq x \rightarrow s_5(y) \lor s_6(y))) \]
Parameterized Communicating Automata (PCA) over Rings

\[
L = \exists x (s_4(x) \land \forall y (y \neq x \rightarrow s_5(y) \lor s_6(y)))
\]
Complementation

\[ L = \exists x (s_4(x) \land \forall y (y \neq x \rightarrow s_5(y) \lor s_6(y))) \]
Complementation

\[
L = \left\{ \exists x(s_4(x) \land \forall y(y \neq x \rightarrow s_5(y) \lor s_6(y))) \right\}
\]
Complementation

\[ L = \{ \text{Diagram} \} \]
Complementation

\[ L = \{ \text{Diagram 1}, \text{Diagram 2}, \text{Diagram 3} \} \]
Complementation

\[
L = \{ \text{\[diagram\]} } = \{ \text{\[diagram\]} } = \{ \text{\[diagram\]} } = \{ \text{\[diagram\]} }
\]
Complementation

\[
L = \{ \}
\]
Negative Results

**Theorem** [B.-Gastin-Kumar; FSTTCS 2014]:
PCAs over rings are not complementable.
Theorem [B.-Gastin-Kumar; FSTTCS 2014]:
PCAs over rings are not complementable.

Proof:
Negative Results

**Theorem [B.-Gastin-Kumar; FSTTCS 2014]:**
PCAs over rings are not complementable.

**Proof:**

Behaviors encode grids.
Theorem [B.-Gastin-Kumar; FSTTCS 2014]:
PCAs over rings are not complementable.

Proof:
- Behaviors encode grids.
- Grid automata are not closed under complementation [Matz-Schweikardt-Thomas ’02].
Negative Results

Theorem [B.-Gastin-Kumar; FSTTCS 2014]:
PCAs over rings are not complementable.

Proof:

Behaviors encode grids.
Grid automata are not closed under complementation
[Matz-Schweikardt-Thomas ’02].

Theorem [Emerson-Namjoshi 2003]:
Emptiness is undecidable for PCAs over rings
(even token-passing systems, unless $|Msg| = 1$).
Negative Results

**Theorem** [B.-Gastin-Kumar; FSTTCS 2014]:
PCAs over rings are not complementable.

**Proof:**

Behaviors encode grids.
Grid automata are not closed under complementation [Matz-Schweikardt-Thomas '02].
Context-Bounded PCAs
Context-Bounded PCAs

**Idea:** Every process is constrained to a bounded number of contexts.
Context-Bounded PCAs

Idea: Every process is contrained to a bounded number of contexts. There are several possible definitions of a context that lead to positive results.
Context-Bounded PCAs

**Idea:** Every process is constrained to a bounded number of contexts. There are several possible definitions of a context that lead to positive results.

**Here:** Process only sends XOR only receives from one fixed neighbor.
Context-Bounded PCAs

**Idea:** Every process is constrained to a bounded number of contexts. There are several possible definitions of a context that lead to positive results.

**Here:** Process only sends XOR only receives from one fixed neighbor.
Context-Bounded PCAs

Idea: Every process is constrained to a bounded number of contexts. There are several possible definitions of a context that lead to positive results.

Here: Process only sends XOR only receives from one fixed neighbor.
Context-Bounded PCAs

Idea: Every process is constrained to a bounded number of contexts. There are several possible definitions of a context that lead to positive results.

Here: Process only sends XOR only receives from one fixed neighbor.
Context-Bounded PCAs

**Idea:** Every process is constrained to a bounded number of contexts. There are several possible definitions of a context that lead to positive results.

**Here:** Process only sends XOR only receives from one fixed neighbor.
Idea: Every process is constrained to a bounded number of contexts. There are several possible definitions of a context that lead to positive results.

Here: Process only sends XOR only receives from one fixed neighbor.
Idea: Every process is constrained to a bounded number of contexts. There are several possible definitions of a context that lead to positive results.

Here: Process only sends XOR only receives from one fixed neighbor.
Context-Bounded PCAs

**Idea:** Every process is constrained to a bounded number of contexts. There are several possible definitions of a context that lead to positive results.

**Here:** Process only sends XOR only receives from one fixed neighbor.
**Context-Bounded PCAs**

**Idea:** Every process is constrained to a bounded number of contexts. There are several possible definitions of a context that lead to positive results.

**Here:** Process only sends XOR only receives from one fixed neighbor.
Context-Bounded PCAs

Idea: Every process is constrained to a bounded number of contexts. There are several possible definitions of a context that lead to positive results.

Here: Process only sends XOR only receives from one fixed neighbor.

3-bounded
Definition: A PCA is $k$-bounded if the finite automaton restricts to $k$ contexts.
**Definition:** A PCA is $k$-bounded if the finite automaton restricts to $k$ contexts.
Definition: A PCA is $k$-bounded if the finite automaton restricts to $k$ contexts.

Theorem [B.-Gastin-Kumar; FSTTCS 2014]:
For every bounded PCA $A$, there is a PCA $B$ such that $L(B) = \overline{L(A)}$. 
Proof Outline

nondeterminism

\[ \exists x(s_1(x) \land \forall y(y \neq x \rightarrow s_5(y) \lor s_6(y))) \]

\( k \)-bounded

disambiguation

every behavior has a unique run

complementation
Proof Outline

nondeterminism

$\exists x (s_4(x) \wedge \forall y (y \neq x \rightarrow s_5(y) \lor s_6(y)))$

$\forall l \in \{0, 1\}$

$\forall r \in \{0, 1\}$

$A$

$k$-bounded

disambiguation

every behavior has a unique run

complementation
Proof Outline

nondeterminism

\[ \exists x (s_4(x) \land \forall y (y \neq x \rightarrow s_5(y) \lor s_6(y))) \]

\[ s_0 \]

\[ s_1 \]

\[ s_2 \]

\[ s_3 \]

\[ s_4 \]

\[ s_5 \]

\[ s_6 \]

\[ r!1 \]

\[ l?1 \]

\[ l?0 \]

\[ r!0 \]

\[ r!0 \]

\[ k\text{-bounded} \]

disambiguation
every behavior has a unique run

\[ A \]

\[ \forall \phi \]

complementation

\[ \neg \phi \]
Proof Outline

nondeterminism

\[
\begin{align*}
\exists x(s_4(x) \land \forall y(y \neq x \rightarrow s_5(y) \lor s_6(y)))
\end{align*}
\]

\(s_0 \xrightarrow{r!1} s_1 \xrightarrow{l?0} s_3 \xrightarrow{l?1} s_2 \xrightarrow{r!0} s_4 \xrightarrow{r!0} s_5 \xrightarrow{r!0} s_6\)

\(k\)-bounded

complementation

Powerset construction not applicable due to message contents.
Proof Outline

nondeterminism

Disambiguation through summaries:
Disambiguation of context-bounded PCAs
Disambiguation of context-bounded PCAs
Disambiguation of context-bounded PCAs
Disambiguation of context-bounded PCAs
Disambiguation of context-bounded PCAs
Every process traverses a bounded number of zones.
Disambiguation of context-bounded PCAs

- Every process traverses a bounded number of zones.
- Zone numbers can be computed unambiguously by a PCA.
Disambiguation of context-bounded PCAs

- Every process traverses a bounded number of zones.
- Zone numbers can be computed unambiguously by a PCA.
Every process traverses a bounded number of zones.
Zone numbers can be computed unambiguously by a PCA.
Disambiguation of context-bounded PCAs

- Every process traverses a bounded number of zones.
- Zone numbers can be computed unambiguously by a PCA.
Every process traverses a bounded number of zones.

Zone numbers can be computed unambiguously by a PCA.
Disambiguation of context-bounded PCAs

Every process traverses a bounded number of zones.
Zone numbers can be computed unambiguously by a PCA.
Every process traverses a bounded number of zones.
Zone numbers can be computed unambiguously by a PCA.
Disambiguation of context-bounded PCAs

- Every process traverses a bounded number of zones.
- Zone numbers can be computed unambiguously by a PCA.
Disambiguation of context-bounded PCAs

\[ R_i \subseteq S^3 \times S^3 \]
Disambiguation of context-bounded PCAs

Every process traverses a bounded number of zones.
Zone numbers can be computed unambiguously.
Sending processes deterministically compute summaries for zones.
Disambiguation of context-bounded PCAs

Every process traverses a bounded number of zones.
Zone numbers can be computed unambiguously.
Sending processes deterministically compute summaries for zones.
Every process traverses a bounded number of zones.
Zone numbers can be computed unambiguously.
Sending processes deterministically compute summaries for zones.
Acceptance condition checks if summaries correspond to accepting run.
Every process traverses a bounded number of zones.

Zone numbers can be computed unambiguously.

Sending processes deterministically compute summaries for zones.

Acceptance condition checks if summaries correspond to accepting run.
Disambiguation of context-bounded PCAs

$m_1 \subseteq S^3 \times S^3$

$m_i \subseteq S^3 \times S^3$

$m_2 \subseteq S^3 \times S^3$

$m_3 \subseteq S^3 \times S^3$

- Every process traverses a bounded number of zones.
- Zone numbers can be computed unambiguously.
- Sending processes deterministically compute summaries for zones.
- Acceptance condition checks if summaries correspond to accepting run.
Disambiguation of context-bounded PCAs

Every process traverses a bounded number of zones.
Zone numbers can be computed unambiguously.
Sending processes deterministically compute summaries for zones.
Acceptance condition checks if summaries correspond to accepting run.
Disambiguation of context-bounded PCAs

Every process traverses a bounded number of zones.
Zone numbers can be computed unambiguously.
Sending processes deterministically compute summaries for zones.
Acceptance condition checks if summaries correspond to accepting run.
Disambiguation of context-bounded PCAs

- Every process traverses a bounded number of zones.
- Zone numbers can be computed unambiguously.
- Sending processes deterministically compute summaries for zones.
- Acceptance condition checks if summaries correspond to accepting run.
The Logic:
MSO logic over graphs, including process nodes and event nodes.
Logical Characterization of Context-Bounded PCAs

The Logic:
MSO logic over graphs, including process nodes and event nodes.
For every bounded set $L$ of behaviors, the following are equivalent:

- $L$ is recognized by some PCA.
- $L$ is definable in MSO logic.

**The Logic:**
MSO logic over graphs, including process nodes and event nodes.

**Corollary [B.-Gastin-Kumar; FSTTCS 2014]:**
For every bounded set $L$ of behaviors, the following are equivalent:

- $L$ is recognized by some PCA.
- $L$ is definable in MSO logic.
Topologies of Bounded Degree

Complementation and MSO characterization hold wrt. the class of all topologies over a fixed set of ports. With 4 ports, this captures rings, binary trees, and grids.
Complementation and MSO characterization hold wrt. the class of all topologies over a fixed set of ports. With 4 ports, this captures rings, binary trees, and grids.
Complementation and MSO characterization hold wrt. the class of all topologies over a fixed set of ports. With 4 ports, this captures rings, binary trees, and grids.
Theorem [B.-Gastin-Kumar; FSTTCS 2014]:
Context-bounded MSO model checking is decidable over rings.

Input: PCA $A$; $k \in \mathbb{N}$; MSO formula $\varphi$
Question: $M \models \varphi$ for all $k$-bounded $M \in L(A)$?
Context-Bounded Model Checking

**Theorem** [B.-Gastin-Kumar; FSTTCS 2014]:
Context-bounded MSO model checking is decidable over rings.

Input: PCA $\mathcal{A}$; $k \in \mathbb{N}$; MSO formula $\varphi$
Question: $M \models \varphi$ for all $k$-bounded $M \in L(\mathcal{A})$?

**Theorem** [B.-Gastin-Schubert; RP 2014]:
Context-bounded nonemptiness checking over rings is PSPACE-complete when the acceptance condition is presented as a finite automaton.

Input: PCA $\mathcal{A}$; $k \in \mathbb{N}$
Question: Does $L(\mathcal{A})$ contain some $k$-bounded behavior?
Context-Bounded Nonemptiness Problem

Theorem [B.-Gastin-Schubert; RP 2014]:
Context-bounded nonemptiness checking over rings is PSPACE-complete.

Input: PCA $A$; $k \in \mathbb{N}$
Question: Does $L(A)$ contain some $k$-bounded behavior?
Theorem [B.-Gastin-Schubert; RP 2014]:
Context-bounded nonemptiness checking over rings is PSPACE-complete.
Theorem [B.-Gastin-Schubert; RP 2014]: Context-bounded nonemptiness checking over rings is PSPACE-complete.
Context-Bounded Nonemptiness Problem

Theorem [B.-Gastin-Schubert; RP 2014]:
Context-bounded nonemptiness checking over rings is PSPACE-complete.

Finite automaton guesses local states & checks membership in summaries.

Input: PCA $A$; $k \in \mathbb{N}$
Question: Does $L(A)$ contain some $k$-bounded behavior?
Theorem [B.-Gastin-Schubert; RP 2014]:
Context-bounded nonemptiness checking over rings is PSPACE-complete.
Theorem [B.-Gastin-Schubert; RP 2014]:
Context-bounded nonemptiness checking over rings is PSPACE-complete.
Theorem [B.-Gastin-Schubert; RP 2014]:
Context-bounded nonemptiness checking over rings is PSPACE-complete.
Context-Bounded Nonemptiness Problem

Input: PCA $A$; $k \in \mathbb{N}$

Question: Does $L(A)$ contain some $k$-bounded behavior?

Finite automaton guesses local states & checks membership in summaries.

Theorem [B.-Gastin-Schubert; RP 2014]:
Context-bounded nonemptiness checking over rings is PSPACE-complete.
Theorem [B.-Gastin-Schubert; RP 2014]:
Context-bounded nonemptiness checking over rings is PSPACE-complete.
Theorem [B.-Gastin-Schubert; RP 2014]:
Context-bounded nonemptiness checking over rings is PSPACE-complete.
Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.
Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.
Context-Bounded Nonemptiness Problem

Finite automaton guesses local states & checks membership in summaries.

However, summaries may match locally, but not give rise to an accepting run!

Check causal dependencies.

Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.
Context-Bounded Nonemptiness Problem

Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.
Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.

Finite automaton guesses local states & checks membership in summaries.
However, summaries may match locally, but not give rise to an accepting run!
Check causal dependencies.

→ = strict precedence
= synchronization
Theorem [B.-Gastin-Schubert 2014]: Context-bounded emptiness checking over rings is PSPACE-complete.
Finite automaton guesses local states & checks membership in summaries.

However, summaries may match locally, but not give rise to an accepting run!

Check causal dependencies.

\[ \rightarrow \] = strict precedence

\[ \underset{\text{light blue}}{\rightarrow} \] = synchronization

**Theorem [B.-Gastin-Schubert 2014]:**

Context-bounded emptiness checking over rings is PSPACE-complete.
Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.

Finite automaton guesses local states & checks membership in summaries.

However, summaries may match locally, but not give rise to an accepting run!

Check causal dependencies.

\[ \rightarrow = \text{strict precedence} \]

\[ \rightarrow = \text{synchronization} \]
Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.
Theorem [B.-Gastin-Schubert 2014]: Context-bounded emptiness checking over rings is PSPACE-complete.
Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.
Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.

strict cycle $\implies$ run is not accepting
Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.
Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.
Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.
Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.
Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.
Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.
Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.
Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.
Theorem:
Context-bounded PCAs are complementable and expressively equivalent to MSO logic.
Summary of Results

Theorem:
Context-bounded PCAs are **complementable** and expressively equivalent to **MSO logic**.

Theorem:
Context-bounded **nonemptiness checking is decidable** over rings and trees.
Summary of Results

Theorem:
Context-bounded PCAs are complementable and expressively equivalent to MSO logic.

Theorem:
Context-bounded nonemptiness checking is decidable over rings and trees.

Corollary:
Context-bounded MSO model checking is decidable over rings and trees.
Summary of Results

**Theorem:**
Context-bounded PCAs are **complementable** and expressively equivalent to **MSO logic**.

**Theorem:**
Context-bounded **nonemptiness checking is decidable** over rings and trees.

**Corollary:**
Context-bounded **MSO model checking is decidable** over rings and trees.

Context-bounded PCAs form a robust automata model.
Franklin's leader-election protocol (1982)
Franklin’s leader-election protocol (1982)
Application to Verification of Distributed Algorithms

Franklin's leader-election protocol (1982)
Franklin’s leader-election protocol (1982)
Distributed algorithms often proceed in rounds/contexts.

Franklin's leader-election protocol (1982)
Distributed algorithms often proceed in rounds/contexts.

Franklin’s leader-election protocol (1982)
Distributed algorithms often proceed in rounds/contexts.

Franklin’s leader-election protocol (1982)
Distributed algorithms often proceed in rounds/contexts.

Franklin's leader-election protocol (1982)
Distributed algorithms often proceed in rounds/contexts.
Number of rounds is sometimes logarithmic in the number of processes.
Distributed algorithms often proceed in rounds/contexts.

Number of rounds is sometimes logarithmic in the number of processes.
Distributed algorithms often proceed in rounds/contexts.

Number of rounds is sometimes logarithmic in the number of processes.

MSO can trace back origin of unique process identifiers (pids).
Distributed algorithms often proceed in rounds/contexts.

Number of rounds is sometimes logarithmic in the number of processes.

MSO can trace back origin of unique process identifiers (pids).
Distributed algorithms often proceed in rounds/contexts.
- Number of rounds is sometimes logarithmic in the number of processes.
- MSO can trace back origin of unique process identifiers (pids).
- Underapproximate verification of distributed algorithms that send and compare pids.

Franklin’s leader-election protocol (1982)

```plaintext
rec(r); r < id
```

```plaintext
rec(r); r > id
```

leader
Beyond Context Bounds …

\[ \exists x(s_4(x) \land \forall y(y \neq x \rightarrow s_5(y) \lor s_6(y))) \]

weak PCA
Beyond Context Bounds …

\[ \exists x(s_4(x) \land \forall y(y \neq x \rightarrow s_5(y) \lor s_6(y))) \]

weak PCA

weak logic
Beyond Context Bounds …

\[ \exists x(s_4(x) \land \forall y(y \neq x \rightarrow s_5(y) \lor s_6(y))) \]

weak PCA

weak logic
\[ \exists x (s_4(x) \land \forall y (y \neq x \rightarrow s_5(y) \lor s_6(y))) \]

weak PCA

weak logic
Beyond Context Bounds …

\[ \exists x(s_4(x) \land \forall y(y \neq x \rightarrow s_5(y) \lor s_6(y))) \]

weak PCA

weak logic
Theorem [B.; CSL-LICS 2014]:
Let $T$ be any of the following topology classes: rings, grids, binary trees.
Theorem [B.; CSL-LICS 2014]:

Let $T$ be any of the following topology classes: rings, grids, binary trees.

For every set $L$ of behaviors over a topology from $T$ the following are equivalent:

- $L$ is recognized by some weak PCA.
- $L$ is definable in weak EMSO logic (projection of weak-FO-definable language).
Theorem [B.; CSL-LICS 2014]:

Let $T$ be any of the following topology classes: rings, grids, binary trees.

For every set $L$ of behaviors over a topology from $T$ the following are equivalent:

- $L$ is recognized by some weak PCA.
- $L$ is definable in weak EMSO logic (projection of weak-FO-definable language).

Other Future Work

- Topologies of unbounded degree (unranked trees, stars, …)

- Temporal logics and efficient model checking

- Split-width for parameterized systems

  [Aiswarya-Gastin-Narayan Kumar 2012]
Other Future Work

- Topologies of unbounded degree (unranked trees, stars, …)

- Temporal logics and efficient model checking

- Split-width for parameterized systems
  [Aiswarya-Gastin-Narayan Kumar 2012]

Thank You!