Bounded context switching for valence systems

Roland Meyer[1], **Sebastian Muskalla**[1], and Georg Zetzsche[2]

September 4, CONCUR 2018, Beijing

1 TU Braunschweig, Germany
`{roland.meyer,s.muskalla}@tu-bs.de`

2 IRIF (Université Paris-Diderot, CNRS), France
`zetzsche@irif.fr`

Result and structure

Theorem

Reachability under bounded context switching

for valence systems over graph monoids

is always in NP *(for all graph monoids).*

Result and structure

Theorem

Reachability under bounded context switching

for valence systems over graph monoids

is always in NP (for all graph monoids).

1. What is bounded context switching (BCS)?

Result and structure

Theorem

Reachability under bounded context switching

for valence systems over graph monoids

is always in NP *(for all graph monoids).*

1. What is bounded context switching (BCS)?
2. What are valence systems over graph monoids?

Result and structure

Theorem

Reachability under bounded context switching

for valence systems over graph monoids

is always in NP *(for all graph monoids).*

1. What is bounded context switching (BCS)?
2. What are valence systems over graph monoids?
3. What is BCS for valence systems?

1. BCS

The problem

Setting:

Concurrent system, each component modeled as automaton

The problem

Setting:
Concurrent system, each component modeled as automaton

Problem:
If components are beyond finite-state,
reachability (safety verification) is *difficult*

The problem

Setting:
Concurrent system, each component modeled as automaton

Problem:
If components are beyond finite-state,
reachability (safety verification) is *difficult*

Solution:
Consider bounded context switching (BCS)

Bounded context switching

Context: Infix of the (sequentialized) computation where a single thread is active

Bounded context switching

Context: Infix of the (sequentialized) computation where a single thread is active

BCS: Number of contexts switches (#contexts -1) bounded by a constant

Bounded context switching

Context: Infix of the (sequentialized) computation where a single thread is active

BCS: Number of contexts switches (#contexts -1) bounded by a constant

<u>**Reachability under bounded context switching**</u> (BCSREACH)

Given: Concurrent system $\mathcal{S}$, number k (in unary)

Decide: Final configuration reachable from initial one in $\mathcal{S}$ by a computation with $\leqslant k$ context switches?

Bounded context switching

Reachability under bounded context switching (BCSREACH)

Given: Concurrent system $\mathcal{S}$, number k (in unary)

Decide: Final configuration reachable from initial one in $\mathcal{S}$ by a computation with $\leqslant k$ context switches?

Reachability under bounded context switching (BCSREACH)

Given: Concurrent system $\mathcal{S}$, number k (in unary)

Decide: Final configuration reachable from initial one in $\mathcal{S}$ by a computation with $\leqslant k$ context switches?

Under-approximation of reachability

Bounded context switching

Reachability under bounded context switching (BCSREACH)

Given: Concurrent system $\mathcal{S}$, number k (in unary)

Decide: Final configuration reachable from initial one in $\mathcal{S}$ by a computation with $\leqslant k$ context switches?

Under-approximation of reachability

Complexity is typically much lower

Bounded context switching

<u>**Reachability under bounded context switching**</u> (BCSREACH)

Given: Concurrent system $\mathcal{S}$, number k (in unary)

Decide: Final configuration reachable from initial one in $\mathcal{S}$ by a computation with $\leqslant k$ context switches?

Under-approximation of reachability

Complexity is typically much lower

Useful as bugs *usually* occur within few context switches [MQ07,LPSZ08]

Example

Example [QR05]:

Concurrent system where each component is a PDS, communicating via finite control

Example

Example [QR05]:

Concurrent system where each component is a PDS, communicating via finite control

↳ essentially a MPDS

Example [QR05]:

Concurrent system where each component is a PDS, communicating via finite control

↳ essentially a MPDS

Reachability is undecidable if #components $\geqslant 2$

Example

Example [QR05]:

Concurrent system where each component is a PDS, communicating via finite control

↳ essentially a MPDS

Reachability is undecidable if #components $\geqslant 2$

Context: Infix in which only one stack is used

Example

Example [QR05]:

Concurrent system where each component is a PDS, communicating via finite control

↳ essentially a MPDS

Reachability is undecidable if #components $\geqslant 2$

Context: Infix in which only one stack is used

Reachability under BCS is NP-complete

Related work

Similar results for

- various types of components,
- various types of communication,
- various BCS-like restrictions.

Related work

Similar results for

various types of components,

various types of communication,

various BCS-like restrictions.

For example:

Queues as storages [LMP08]

Pushdowns with dynamic thread creation [ABQ09]

Pushdowns communicating via queues [HLMS12]

...

Related work II

Our goal: General BCS result

Related work II

Our goal: General BCS result

Other people's work:

Related work II

Our goal: General BCS result

Other people's work:

Using graph-theoretic measures (tree width, ...) [MP11, A14]

- Can handle queues
- Cannot handle counters
- Applies to settings where the complexity is beyond NP

Related work II

Our goal: General BCS result

Other people's work:

Using graph-theoretic measures (tree width, ...) [MP11, A14]

- Can handle queues
- Cannot handle counters
- Applies to settings where the complexity is beyond NP

Reductions to ∃PA-satisfiability [HL12,EGT14]

- Can handle reversal-bounded counters
- Does not allow nested combination of counters and stack

Related work II

Our goal: General BCS result

Other people's work:

Using graph-theoretic measures (tree width, ...) [MP11, A14]

- Can handle queues
- Cannot handle counters
- Applies to settings where the complexity is beyond NP

Reductions to ∃PA-satisfiability [HL12,EGT14]

- Can handle reversal-bounded counters
- Does not allow nested combination of counters and stack

Results incomparable to ours

Related work II

Our goal: General BCS result

Other people's work:

Using graph-theoretic measures (tree width, ...) [MP11, A14]

- Can handle queues
- Cannot handle counters
- Applies to settings where the complexity is beyond NP

Reductions to ∃PA-satisfiability [HL12,EGT14]

- Can handle reversal-bounded counters
- Does not allow nested combination of counters and stack

Results incomparable to ours

Our technique provides an algebraic view

2. Valence systems

Valence systems

Need a single model that can represent various types of memory

Valence systems

Need a single model that can represent various types of memory

Introducing valence systems over some monoid $\mathbb{M}$

Valence systems

Need a single model that can represent various types of memory

Introducing valence systems over some monoid $\mathbb{M}$

Monoid $\mathbb{M}$ represents the storage of the system

Valence systems

Need a single model that can represent various types of memory

Introducing valence systems over some monoid $\mathbb{M}$

Monoid $\mathbb{M}$ represents the storage of the system

Syntax:

- Finite control
- Transitions labeled by generators of $\mathbb{M}$

Valence systems

Need a single model that can represent various types of memory

Introducing valence systems over some monoid $\mathbb{M}$

Monoid $\mathbb{M}$ represents the storage of the system

Syntax:

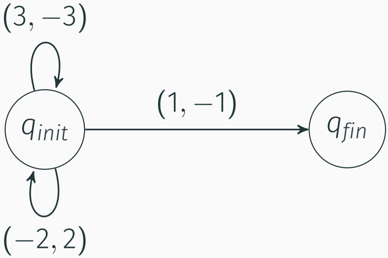
- Finite control
- Transitions labeled by generators of $\mathbb{M}$

Semantics:

- Configurations (q, m) with q control state, $m \in \mathbb{M}$
- Transition $q \xrightarrow{m'} q'$ leads to $(q', m \cdot m')$

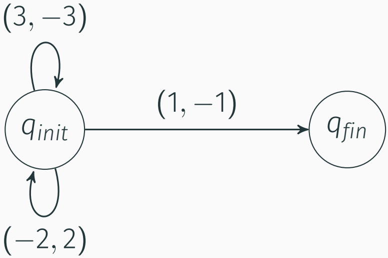
Example

Valence system over $\mathbb{Z} \times \mathbb{Z}$ (with component-wise addition)



Example

Valence system over $\mathbb{Z} \times \mathbb{Z}$ (with component-wise addition)



(essentially an integer 2-VASS)

Want results of the following shape:

Theorem

If monoid $\mathbb{M}$ satisfies condition c, then checking property P for all valence systems over $\mathbb{M}$ is in complexity class $\mathcal{C}$.

Graph monoids

Want results of the following shape:

Theorem

If monoid $\mathbb{M}$ satisfies condition c, then checking property P for all valence systems over $\mathbb{M}$ is in complexity class $\mathcal{C}$.

Best case: Complete for classification for property P

Want results of the following shape:

Theorem

If monoid $\mathbb{M}$ satisfies condition c, then checking property P for all valence systems over $\mathbb{M}$ is in complexity class $\mathcal{C}$.

Best case: Complete for classification for property P

For example, want classification of $P =$ reachability

Reachability for valence systems

Given: Valence system $\mathcal{A}$ over monoid $\mathbb{M}$

Decide: $(q_{init}, 1_{\mathbb{M}}) \to^* (q_{final}, 1_{\mathbb{M}})$?

Problem: Monoids are too diverse

Graph monoids

Problem: Monoids are too diverse

Focus on an interesting subclass of monoids

Graph monoids

Problem: Monoids are too diverse

Focus on an interesting subclass of monoids

Finitely generated monoids: Too diverse

Graph monoids

Problem: Monoids are too diverse

Focus on an interesting subclass of monoids

- Finitely generated monoids: Too diverse
- Finite monoids: Not expressive

Problem: Monoids are too diverse

Focus on an interesting subclass of monoids

- **Finitely generated monoids**: Too diverse
- **Finite monoids**: Not expressive
- Graph monoids

Example: Graph monoid

Consider the following undirected graph:

Example: Graph monoid

Consider the following undirected graph:

Nodes a, b are counters / stack symbols

Example: Graph monoid

Consider the following undirected graph:

Nodes a, b are counters / stack symbols

Operations: a^+, b^+ (“push a / b”, “increment a / b”)

and a^-, b^- (“pop a / b”, “decrement a / b”)

Example: Graph monoid

Consider the following undirected graph:

Nodes a, b are counters / stack symbols

Operations: a^+, b^+ ("push a / b", "increment a / b")

and a^-, b^- ("pop a / b", "decrement a / b")

Monoid elements: Sequences of operations

Example: Graph monoid

Consider the following undirected graph:

Nodes a, b are counters / stack symbols

Operations: a^+, b^+ (“push a / b”, “increment a / b”)

and a^-, b^- (“pop a / b”, “decrement a / b”)

Monoid elements: Sequences of operations

modulo the congruence $o^+.o^- \cong \varepsilon$

Example: PDS

$$\mathbb{M}_G = \left\{a^+, b^+, a^-, b^-\right\}^* / \cong$$

$$o^+.o^- \cong \varepsilon \; \forall o \in \{a, b\}$$

Example: PDS

$$\mathbb{M}_G = \{a^+, b^+, a^-, b^-\}^* / \cong$$

$$o^+.o^- \cong \varepsilon \; \forall o \in \{a, b\}$$

$\bullet$ a $\quad$ $\bullet$ b

$a^+ b^+ b^- a^-$

Example: PDS

$$\mathbb{M}_G = \left\{a^+, b^+, a^-, b^-\right\}^* / \cong$$

$$o^+.o^- \cong \varepsilon \; \forall o \in \{a, b\}$$

$$a^+ b^+ b^- a^- \;\cong\; a^+ a^-$$

$$\mathbb{M}_G = \{a^+, b^+, a^-, b^-\}^* / \cong$$

$$o^+.o^- \cong \varepsilon \; \forall o \in \{a, b\}$$

$$a^+ b^+ b^- a^- \;\cong\; a^+ a^- \;\cong\; \varepsilon \;=\; 1_{\mathbb{M}}$$

Example: PDS

$$\mathbb{M}_G = \left\{a^+, b^+, a^-, b^-\right\}^* / \cong$$

$$o^+.o^- \cong \varepsilon \; \forall o \in \{a, b\}$$

$$a^+ b^+ b^- a^- \;\cong\; a^+ a^- \;\cong\; \varepsilon \;=\; 1_{\mathbb{M}}$$

$a^+ b^+ a^- b^-$ irreducible

Example: PDS

$$\mathbb{M}_G = \{a^+, b^+, a^-, b^-\}^* / \cong$$

$$o^+.o^- \cong \varepsilon \; \forall o \in \{a, b\}$$

$$a^+ b^+ b^- a^- \;\cong\; a^+ a^- \;\cong\; \varepsilon \;=\; 1_{\mathbb{M}}$$

$a^+ b^+ a^- b^-$ irreducible

$a^- a^+$ irreducible

Example: PDS

$$\mathbb{M}_G = \{a^+, b^+, a^-, b^-\}^* / \cong$$

$$o^+.o^- \cong \varepsilon \; \forall o \in \{a, b\}$$

$$a^+ b^+ b^- a^- \;\cong\; a^+ a^- \;\cong\; \varepsilon \;=\; 1_{\mathbb{M}}$$

$a^+ b^+ a^- b^-$ irreducible

$a^- a^+$ irreducible

Valence systems over $\mathbb{M}_G$ are PDS over stack alphabet $\{a, b\}$

Graph monoids

Graph monoid $\mathbb{M}_G$ given by undirected graph $G = (V, \mathcal{I})$

Graph monoids

Graph monoid $\mathbb{M}_G$ given by undirected graph $G = (V, \mathcal{I})$

Nodes of G are counters

Graph monoids

Graph monoid $\mathbb{M}_G$ given by undirected graph $G = (V, \mathcal{I})$

- Nodes of G are counters
- Operations $\mathcal{O}$ consisting of o^+, o^- for each node $o \in V$

Graph monoids

Graph monoid $\mathbb{M}_G$ given by undirected graph $G = (V, \mathcal{I})$

- Nodes of G are counters
- Operations $\mathcal{O}$ consisting of o^+, o^- for each node $o \in V$
- $\mathbb{M}_G = \mathcal{O}^* / \cong$

Graph monoids

Graph monoid $\mathbb{M}_G$ given by undirected graph $G = (V, \mathcal{I})$

- Nodes of G are counters
- Operations $\mathcal{O}$ consisting of o^+, o^- for each node $o \in V$
- $\mathbb{M}_G = \mathcal{O}^* / \cong$
- Monoid elements are represented by sequences of operations

Graph monoids

Graph monoid $\mathbb{M}_G$ given by undirected graph $G = (V, \mathcal{I})$

Nodes of G are counters

Operations $\mathcal{O}$ consisting of o^+, o^- for each node $o \in V$

$\mathbb{M}_G = \mathcal{O}^* / \cong$

Monoid elements are represented by sequences of operations

Monoid operation: Concatenation of representatives

Graph monoid $\mathbb{M}_G$ given by undirected graph $G = (V, \mathcal{I})$

- Nodes of G are counters
- Operations $\mathcal{O}$ consisting of o^+, o^- for each node $o \in V$
- $\mathbb{M}_G = \mathcal{O}^* / \cong$
- Monoid elements are represented by sequences of operations
- Monoid operation: Concatenation of representatives
- Congruence $\cong$ satisfies $o^+ . o^- \cong \varepsilon$ for all o

Graph monoids

Graph monoid $\mathbb{M}_G$ given by undirected graph $G = (V, \mathcal{I})$

Congruence $\cong$ satisfies $o^+.o^- \cong \varepsilon$ for all o

Graph monoids

Graph monoid $\mathbb{M}_G$ given by undirected graph $G = (V, \mathcal{I})$

Congruence $\cong$ satisfies $o^+.o^- \cong \varepsilon$ for all o

Edge relation $\mathcal{I}$ called independence relation

Graph monoid $\mathbb{M}_G$ given by undirected graph $G = (V, \mathcal{I})$

Congruence $\cong$ satisfies $o^+.o^- \cong \varepsilon$ for all o

Edge relation $\mathcal{I}$ called independence relation

Intuition:

Graph monoids

Graph monoid $\mathbb{M}_G$ given by undirected graph $G = (V, \mathcal{I})$

Congruence $\cong$ satisfies $o^+.o^- \cong \varepsilon$ for all o

Edge relation $\mathcal{I}$ called independence relation

Intuition:

If $o \mathrel{\mathcal{I}} u$, then o and u belong to independents part of the storage

Graph monoids

Graph monoid $\mathbb{M}_G$ given by undirected graph $G = (V, \mathcal{I})$

Congruence $\cong$ satisfies $o^+.o^- \cong \varepsilon$ for all o

Edge relation $\mathcal{I}$ called independence relation

Intuition:

- If $o \mathrel{\mathcal{I}} u$, then o and u belong to independents part of the storage
- Congruence should identify computations that order independent operations differently

Graph monoids

Graph monoid $\mathbb{M}_G$ given by undirected graph $G = (V, \mathcal{I})$

Congruence $\cong$ satisfies $o^+.o^- \cong \varepsilon$ for all o

Edge relation $\mathcal{I}$ called independence relation

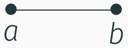
Intuition:

If $o \mathcal{I} u$, then o and u belong to independents part of the storage

Congruence should identify computations that order independent operations differently

If $o \mathcal{I} u$, then $o^\pm$ and $u^\pm$ commute: $o^\pm.u^\pm \cong u^\pm.o^\pm$

Example: VASS

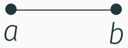


$$\mathbb{M}_G = \left\{a^+, b^+, a^-, b^-\right\}^* / \cong$$

$$o^+.o^- \cong \varepsilon \;\forall o \in \{a, b\}$$

$$o^\pm.u^\pm \cong u^\pm.o^\pm \text{ where } \{u, o\} = \{a, b\}$$

Example: VASS



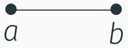
$$\mathbb{M}_G = \{a^+, b^+, a^-, b^-\}^* / \cong$$

$$o^+.o^- \cong \varepsilon \; \forall o \in \{a, b\}$$

$$o^\pm.u^\pm \cong u^\pm.o^\pm \text{ where } \{u, o\} = \{a, b\}$$

$a^+ b^+ b^- a^- \;\cong\; a^+ a^- \;\cong\; \varepsilon \;=\; 1_{\mathbb{M}}$ still valid

Example: VASS



$$\mathbb{M}_G = \left\{a^+, b^+, a^-, b^-\right\}^* / \cong$$

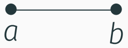
$$o^+.o^- \cong \varepsilon \;\forall o \in \{a, b\}$$

$$o^\pm.u^\pm \cong u^\pm.o^\pm \text{ where } \{u, o\} = \{a, b\}$$

$$a^+b^+b^-a^- \;\cong\; a^+a^- \;\cong\; \varepsilon \;=\; 1_{\mathbb{M}} \qquad \text{still valid}$$

$$a^+b^+a^-b^- \;\cong\; a^+b^+b^-a^- \;\cong\; \varepsilon$$

Example: VASS



$$\mathbb{M}_G = \left\{a^+, b^+, a^-, b^-\right\}^* / \cong$$

$$o^+.o^- \cong \varepsilon \; \forall o \in \{a, b\}$$

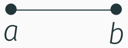
$$o^{\pm}.u^{\pm} \cong u^{\pm}.o^{\pm} \text{ where } \{u, o\} = \{a, b\}$$

$a^+b^+b^-a^- \;\cong\; a^+a^- \;\cong\; \varepsilon \;=\; 1_{\mathbb{M}}$ still valid

$a^+b^+a^-b^- \;\cong\; a^+b^+b^-a^- \;\cong\; \varepsilon$

a^-a^+ still irreducible

Example: VASS



$$\mathbb{M}_G = \{a^+, b^+, a^-, b^-\}^* / \cong$$

$$o^+.o^- \cong \varepsilon \; \forall o \in \{a, b\}$$

$$o^\pm.u^\pm \cong u^\pm.o^\pm \text{ where } \{u, o\} = \{a, b\}$$

$a^+b^+b^-a^- \;\cong\; a^+a^- \;\cong\; \varepsilon \;=\; 1_{\mathbb{M}}$ still valid

$a^+b^+a^-b^- \;\cong\; a^+b^+b^-a^- \;\cong\; \varepsilon$

a^-a^+ still irreducible

Valence systems over $\mathbb{M}_G$ are 2-VASS

$$\mathbb{M}_G = \{a^+, b^+, a^-, b^-\}^* / \cong$$

$$o^+.o^- \cong \varepsilon \ \forall o \in \{a, b\}$$

$$o^\pm.u^\pm \cong u^\pm.o^\pm \ \forall u, o \in \{a, b\}$$

Example: integer VASS

$$\mathbb{M}_G = \{a^+, b^+, a^-, b^-\}^* / \cong$$

$$o^+.o^- \cong \varepsilon \ \forall o \in \{a, b\}$$

$$o^\pm.u^\pm \cong u^\pm.o^\pm \ \forall u, o \in \{a, b\}$$

$a^+b^+b^-a^- \cong a^+a^- \cong \varepsilon = 1_{\mathbb{M}}$ still valid

Example: integer VASS

$$\mathbb{M}_G = \{a^+, b^+, a^-, b^-\}^* / \cong$$

$$o^+.o^- \cong \varepsilon \ \forall o \in \{a, b\}$$

$$o^\pm.u^\pm \cong u^\pm.o^\pm \ \forall u, o \in \{a, b\}$$

$a^+b^+b^-a^- \ \cong \ a^+a^- \ \cong \ \varepsilon \ = \ 1_{\mathbb{M}}$ still valid

$a^+b^+a^-b^- \ \cong \ a^+b^+b^-a^- \ \cong \ \varepsilon$

Example: integer VASS

$$\mathbb{M}_G = \{a^+, b^+, a^-, b^-\}^* / \cong$$

$$o^+.o^- \cong \varepsilon \ \forall o \in \{a, b\}$$

$$o^\pm.u^\pm \cong u^\pm.o^\pm \ \forall u, o \in \{a, b\}$$

$a^+b^+b^-a^- \ \cong \ a^+a^- \ \cong \ \varepsilon \ = \ 1_{\mathbb{M}}$ still valid

$a^+b^+a^-b^- \ \cong \ a^+b^+b^-a^- \ \cong \ \varepsilon$

$a^-a^+ \ \cong \ a^+a^- \ \cong \ \varepsilon$

Example: integer VASS

$$\mathbb{M}_G = \{a^+, b^+, a^-, b^-\}^* / \cong$$

$$o^+.o^- \cong \varepsilon \;\forall o \in \{a, b\}$$

$$o^\pm.u^\pm \cong u^\pm.o^\pm \;\forall u, o \in \{a, b\}$$

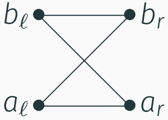
$a^+b^+b^-a^- \;\cong\; a^+a^- \;\cong\; \varepsilon \;=\; 1_{\mathbb{M}}$ still valid

$a^+b^+a^-b^- \;\cong\; a^+b^+b^-a^- \;\cong\; \varepsilon$

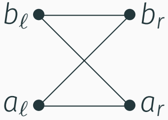
$a^-a^+ \;\cong\; a^+a^- \;\cong\; \varepsilon$

Valence systems over $\mathbb{M}_G$ are integer 2-VASS

Example: MPDS



Example: MPDS

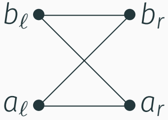


Any $m \in \{{a_\ell}^+, {a_\ell}^-, \ldots\}^*$ can be written as

$$m \cong m_{\upharpoonright_\ell} . m_{\upharpoonright_r}$$

such that $m \cong \varepsilon$ iff $m_{\upharpoonright_\ell} \cong \varepsilon$ and $m_{\upharpoonright_r} \cong \varepsilon$

Example: MPDS



Any $m \in \{{a_\ell}^+, {a_\ell}^-, \ldots\}^*$ can be written as

$$m \cong m_{\restriction_\ell}.m_{\restriction_r}$$

such that $m \cong \varepsilon$ iff $m_{\restriction_\ell} \cong \varepsilon$ and $m_{\restriction_r} \cong \varepsilon$

Valence systems over $\mathbb{M}_G$ are 2-PDS (with a binary stack alphabet for each stack)

Graph monoids can model:

Graph monoids can model:

- Natural (partially blind) counters
- Integer (blind) counters

Graph monoids can model:

- Natural (partially blind) counters
- Integer (blind) counters
- Combinations of these

Graph monoids

Graph monoids can model:

- Natural (partially blind) counters
- Integer (blind) counters
- Combinations of these
- Stacks of these

Graph monoids can model:

- Natural (partially blind) counters
- Integer (blind) counters
- Combinations of these
- Stacks of these

Graph monoids cannot model:

Graph monoids

Graph monoids can model:

- Natural (partially blind) counters
- Integer (blind) counters
- Combinations of these
- Stacks of these

Graph monoids cannot model:

- Queues

Graph monoids

Graph monoids can model:

- Natural (partially blind) counters
- Integer (blind) counters
- Combinations of these
- Stacks of these

Graph monoids cannot model:

- Queues
- Higher-order stacks

Results

Characterization results for valence systems/automata:

- reachability [Z15]
- regularity [Z11]
- context-freeness [BZ13]
- semilinearity of the Parikh image [BZ13]
- ...

3. BCS for valence systems

BCS for valence systems

How to define BCS for valence systems over graph monoids?

How to define BCS for valence systems over graph monoids?

Concurrent system as valence system

Assume:

How to define BCS for valence systems over graph monoids?

Concurrent system as valence system

Assume:

The system is modeled as a single valence system

How to define BCS for valence systems over graph monoids?

Concurrent system as valence system

Assume:

- The system is modeled as a single valence system
- The monoid models the total storage of all components

How to define BCS for valence systems over graph monoids?

Concurrent system as valence system

Assume:

- The system is modeled as a single valence system
- The monoid models the total storage of all components
- The components share a control state
  (communication between components)

BCS for valence systems

How to define BCS for valence systems over graph monoids?

How to define BCS for valence systems over graph monoids?

A slight modification

How to define BCS for valence systems over graph monoids?

A slight modification

Consider configurations of the shape (q, m) where m is a sequence of operations

How to define BCS for valence systems over graph monoids?

A slight modification

Consider configurations of the shape (q, m) where m is a sequence of operations

We do not store the monoid element, but its syntactic representation

How to define BCS for valence systems over graph monoids?

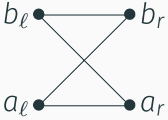
A slight modification

Consider configurations of the shape (q, m) where m is a sequence of operations

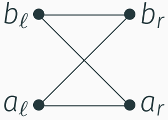
We do not store the monoid element, but its syntactic representation

Crucial as our notion of context is not invariant under congruence

Contexts

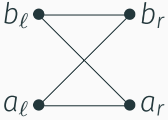


Contexts



Nodes belonging to independent parts of the storage are connected by an edge

Contexts



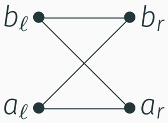
Nodes belonging to independent parts of the storage are connected by an edge

Intuitively:

$$m \;=\; \ldots o^{\pm}.u^{\pm} \ldots$$

with $o \, \mathcal{I} \, u$, then this constitutes a context switch

Contexts



Nodes belonging to independent parts of the storage are connected by an edge

Intuitively:

$$m \; = \; \dots o^{\pm}.u^{\pm} \dots$$

with $o \; \mathcal{I} \; u$, then this constitutes a context switch

In general, we need a more restrictive definition

Dependent computations

Definition

A sequence of operations m is called dependent

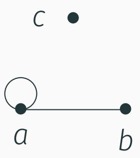
if for all $o^{\pm}, u^{\pm}$ in m with $o \neq u$, $o \mathcal{I} u$ **does not hold**.

Dependent computations

Definition

A sequence of operations m is called dependent

if for all $o^{\pm}, u^{\pm}$ in m with $o \neq u$, $o \mathcal{I} u$ **does not hold**.

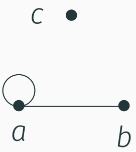


Definition

A sequence of operations m is called dependent

if for all $o^{\pm}, u^{\pm}$ in m with $o \neq u$, $o \,\mathcal{I}\, u$ **does not hold**.

$a^{+}c^{+}$ dependent



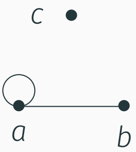
Dependent computations

Definition

A sequence of operations m is called dependent

if for all $o^{\pm}, u^{\pm}$ in m with $o \neq u$, $o \mathcal{I} u$ **does not hold**.

a^+c^+	dependent
b^+c^+	dependent

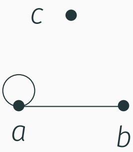


Dependent computations

Definition

A sequence of operations m is called dependent

if for all $o^{\pm}, u^{\pm}$ in m with $o \neq u$, $o \, \mathcal{I} \, u$ **does not hold**.



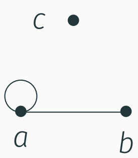
a^+c^+	dependent
b^+c^+	dependent
a^+b^+	not dependent

Dependent computations

Definition

A sequence of operations m is called dependent

if for all $o^{\pm}, u^{\pm}$ in m with $o \neq u$, $o \mathcal{I} u$ **does not hold**.

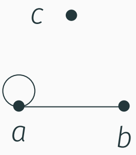


a^+c^+	dependent
b^+c^+	dependent
a^+b^+	not dependent
$a^+c^+b^+$	not dependent

Definition

A sequence of operations m is called dependent

if for all $o^{\pm}, u^{\pm}$ in m with $o \neq u$, $o \mathcal{I} u$ **does not hold**.



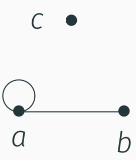
a^+c^+	dependent
b^+c^+	dependent
a^+b^+	not dependent
$a^+c^+b^+$	not dependent
a^+a^-	dependent

Dependent computations

Definition

A sequence of operations m is called dependent

if for all $o^{\pm}, u^{\pm}$ in m with $o \neq u$, $o \mathcal{I} u$ **does not hold**.



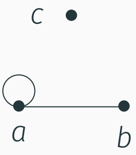
a^+c^+	dependent
b^+c^+	dependent
a^+b^+	not dependent
$a^+c^+b^+$	not dependent
a^+a^-	dependent
$a^+b^+b^-a^-$	not dependent

Dependent computations

Definition

A sequence of operations m is called dependent

if for all $o^{\pm}, u^{\pm}$ in m with $o \neq u$, $o \,\mathcal{I}\, u$ **does not hold**.



a^+c^+	dependent
b^+c^+	dependent
a^+b^+	not dependent
$a^+c^+b^+$	not dependent
a^+a^-	dependent
$a^+b^+b^-a^-$	not dependent
	but $a^+a^- \cong a^+b^+b^-a^-$!

Contexts & context switches

Let m be a sequence of operations

Let m be a sequence of operations

Its first context is its maximal dependent prefix

Contexts & context switches

Let m be a sequence of operations

Its first context is its maximal dependent prefix

Inductively:
The i^{th} context of m is the maximal dependent prefix of m with the first $i-1$ contexts removed

Contexts & context switches

Let m be a sequence of operations

Its first context is its maximal dependent prefix

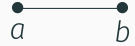
Inductively:
The i^{th} context of m is the maximal dependent prefix of m with the first $i-1$ contexts removed

The number of context switches $cs(m)$ is the number of contexts minus 1

In the examples

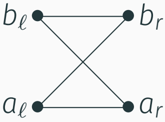
Assume the number of context switches is bounded by k

PDS — no restriction



VASS — changing the counter $\leqslant k$ times

integer VASS — changing the counter $\leqslant k$ times



MPDS — changing the stack $\leqslant k$ times

The result

BCSREACH for valence systems over graph monoids

Given: Valence system $\mathcal{A}$ over $\mathbb{M}_G$, number k (in unary)

Decide: Is there $(q_{init}, \varepsilon) \to (q_{final}, m)$
with $m \cong \varepsilon$ and $cs(m) \leqslant k$?

The result

BCSREACH for valence systems over graph monoids

Given: Valence system $\mathcal{A}$ over $\mathbb{M}_G$, number k (in unary)

Decide: Is there $(q_{init}, \varepsilon) \to (q_{final}, m)$ with $m \cong \varepsilon$ and $cs(m) \leqslant k$?

Theorem

BCSREACH for valence systems over graph monoids is in NP (for all graph monoids).

The proof / The algorithm

Proof outline

Need to find a computation $(q_{init}, \varepsilon) \to^* (q_{final}, m)$ with $m \cong \varepsilon$

Proof outline

Need to find a computation $(q_{init}, \varepsilon) \rightarrow^* (q_{final}, m)$ with $m \cong \varepsilon$

Good: Bound $cs(m) \leqslant k$

Proof outline

Need to find a computation $(q_{init}, \varepsilon) \rightarrow^* (q_{final}, m)$ with $m \cong \varepsilon$

Good: Bound $cs(m) \leqslant k$

Bad: No bound on length of length of m

Proof outline

Need to find a computation $(q_{init}, \varepsilon) \to^* (q_{final}, m)$ with $m \cong \varepsilon$

Good: Bound $cs(m) \leqslant k$

Bad: No bound on length of length of m

Consider blockwise-reduction

Proof outline

Need to find a computation $(q_{init}, \varepsilon) \to^* (q_{final}, m)$ with $m \cong \varepsilon$

Good: Bound $cs(m) \leqslant k$

Bad: No bound on length of length of m

Consider blockwise-reduction

If contexts irreducible, get existence of a reducible block decomposition of length $\leqslant k^2$

Proof outline

Need to find a computation $(q_{init}, \varepsilon) \to^* (q_{final}, m)$ with $m \cong \varepsilon$

Good: Bound $cs(m) \leqslant k$

Bad: No bound on length of length of m

Consider blockwise-reduction

If contexts irreducible, get existence of a reducible block decomposition of length $\leqslant k^2$

Ensure irreducibility by saturating system

Proof outline

Need to find a computation $(q_{init}, \varepsilon) \rightarrow^* (q_{final}, m)$ with $m \cong \varepsilon$

Good: Bound $cs(m) \leqslant k$

Bad: No bound on length of length of m

Consider blockwise-reduction

If contexts irreducible, get existence of a reducible block decomposition of length $\leqslant k^2$

Ensure irreducibility by saturating system

Then check existence of reducible block decomposition using guessing and representing blocks as finite automata

Block decomposition

If $m \cong \varepsilon$, then there is a reduction of m that

swaps letters

cancels letters.

Block decomposition

If $m \cong \varepsilon$, then there is a reduction of m that

swaps letters

cancels letters.

Can define similarly a notation of reduction that

swaps blocks (infixes)

cancels blocks

in one step.

Block decomposition

If $m \cong \varepsilon$, then there is a reduction of m that

swaps letters

cancels letters.

Can define similarly a notation of reduction that

swaps blocks (infixes)

cancels blocks

in one step.

E.g. $m_1.m_2 \to m_2.m_1$ if every symbol in m_1 commutes with every symbol in m_2

Block decomposition

Let $m = m_1, m_2, \ldots, m_n$ be a decomposition of m into blocks.

Block decomposition

Let $m = m_1, m_2, \ldots, m_n$ be a decomposition of m into blocks.

If m can be reduced to ε by blockwise operations,
call it freely reducible.

Block decomposition

Let $m = m_1, m_2, \ldots, m_n$ be a decomposition of m into blocks.

If m can be reduced to ε by blockwise operations,
call it freely reducible.

> If $m \cong \varepsilon$, then its decomposition into letters is always freely reducible.

Block decomposition

Let $m = m_1, m_2, \ldots, m_n$ be a decomposition of m into blocks.

If m can be reduced to ε by blockwise operations, call it freely reducible.

> If $m \cong \varepsilon$, then its decomposition into letters is always freely reducible.
>
> Coarser decompositions might not be freely reducible:
>
> $$o^+u^+ \;,\; u^- \;,\; o^-$$

Block decomposition

Sequence is irreducible if it is not congruent to a shorter one

Block decomposition

Sequence is irreducible if it is not congruent to a shorter one

Theorem

Let m be a sequence of operations with

- *k contexts*
- *each of them irreducible, and*
- *$m \cong \varepsilon$.*

Then there is a decomposition of m into $\leqslant k^2$ blocks that is freely reducible.

Block decomposition

Sequence is irreducible if it is not congruent to a shorter one

Theorem

Let m be a sequence of operations with

- *k contexts*
- *each of them irreducible, and*
- $m \;\cong\; \varepsilon.$

Then there is a decomposition of m into $\leqslant k^2$ blocks that is freely reducible.

Size of the decomposition is independent of the length of m

Block decomposition

Sequence is irreducible if it is not congruent to a shorter one

Theorem

Let m be a sequence of operations with

- *k contexts*
- *each of them irreducible, and*
- *$m \;\cong\; \varepsilon$.*

Then there is a decomposition of m into $\leqslant k^2$ blocks that is freely reducible.

Size of the decomposition is independent of the length of m

Existence can be checked algorithmically

The algorithm

Given: valence system $\mathcal{A}$, bound k

The algorithm, Step I

The algorithm

Given: valence system $\mathcal{A}$, bound k

Part I: Enforcing irreducibility

The algorithm

Given: valence system $\mathcal{A}$, bound k

Part I: Enforcing irreducibility

1. Guess $\leqslant k$ dependent parts of $\mathcal{A}$

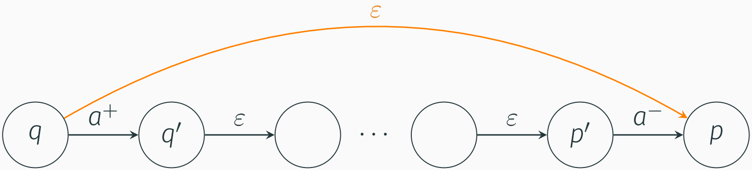
The algorithm, Step I

The algorithm

Given: valence system $\mathcal{A}$, bound k

Part I: Enforcing irreducibility

1. Guess $\leqslant k$ dependent parts of $\mathcal{A}$
2. Saturate each part:



Let $\mathcal{A}_{\mathsf{sat}}$ be the resulting valence system

The algorithm, Step I

Let $\mathcal{A}_{\mathrm{sat}}$ be the resulting valence system

Theorem

$(q_{init}, \varepsilon) \rightarrow^* (q_{final}, m)$ in $\mathcal{A}$ with $m \cong \varepsilon$ and $cs(m) \leqslant k$,

The algorithm, Step I

Let $\mathcal{A}_{\mathsf{sat}}$ be the resulting valence system

Theorem

$(q_{init}, \varepsilon) \to^* (q_{final}, m)$ *in* $\mathcal{A}$ *with* $m \cong \varepsilon$ *and* $cs(m) \leqslant k$,

iff

$(q_{init}, \varepsilon) \to^* (q_{final}, m')$ *in* $\mathcal{A}_{sat}$ *with* $m' \cong \varepsilon$, $cs(m') \leqslant k$,
and contexts of m' *irreducible.*

The algorithm, Step II

Part II:
Checking the existence of a freely reducible block decomposition

Part II:
Checking the existence of a freely reducible block decomposition

3. For each context i, guess part of $\mathcal{A}_{\mathrm{sat}}$ that is used in block $m_{i,j}$ as NFA $\mathcal{A}_{i,j}$

The algorithm, Step II

Part II:
Checking the existence of a freely reducible block decomposition

3. For each context i, guess part of $\mathcal{A}_{\mathsf{sat}}$ that is used in block $m_{i,j}$ as NFA $\mathcal{A}_{i,j}$
4. Guess reduction on blocks (NFAs) of polynomial length

The algorithm, Step II

Part II:
Checking the existence of a freely reducible block decomposition

3. For each context i, guess part of $\mathcal{A}_{\text{sat}}$ that is used in block $m_{i,j}$ as NFA $\mathcal{A}_{i,j}$
4. Guess reduction on blocks (NFAs) of polynomial length
5. Verify reduction step-by-step

The algorithm, Step II

Part II:
Checking the existence of a freely reducible block decomposition

3. For each context i, guess part of $\mathcal{A}_{\text{sat}}$ that is used in block $m_{i,j}$ as NFA $\mathcal{A}_{i,j}$

4. Guess reduction on blocks (NFAs) of polynomial length

5. Verify reduction step-by-step

   Swap rule applicable to $\mathcal{A}_{i,j}$, $\mathcal{A}_{i',j'}$
   if $\forall o^{\pm} \in \text{Alphabet}(\mathcal{A}_{i,j})\ \forall u^{\pm} \in \text{Alphabet}(\mathcal{A}_{i',j'})\colon o\ \mathcal{I}\ u$

The algorithm, Step II

Part II:
Checking the existence of a freely reducible block decomposition

3. For each context i, guess part of $\mathcal{A}_{\mathrm{sat}}$ that is used in block $m_{i,j}$ as NFA $\mathcal{A}_{i,j}$
4. Guess reduction on blocks (NFAs) of polynomial length
5. Verify reduction step-by-step

Swap rule applicable to $\mathcal{A}_{i,j}$, $\mathcal{A}_{i',j'}$
if $\forall o^{\pm} \in \text{Alphabet}(\mathcal{A}_{i,j})\ \forall u^{\pm} \in \text{Alphabet}(\mathcal{A}_{i',j'})\colon o\ \mathcal{I}\ u$

Cancel rule applicable to $\mathcal{A}_{i,j}$, $\mathcal{A}_{i',j'}$
if $\mathcal{L}(\mathcal{A}_{i,j}) \cap \mathcal{L}(\mathcal{A}_{i',j'})^{\text{inverse}}$ is non-empty

Complexity for fixed graphs

Complexity for fixed graphs

Now, assume that the graph G is fixed, consider BCSREACH(G)

Complexity for fixed graphs

Now, assume that the graph G is fixed, consider BCSREACH(G)

Let G^- denote G with self-loops removed.

Complexity for fixed graphs

Now, assume that the graph G is fixed, consider BCSREACH(G)

Let G^- denote G with self-loops removed.

Theorem

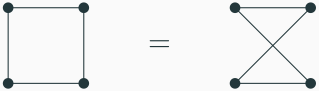
If G^- is a clique, then BCSREACH(G) *is* NL-*complete.*

Complexity for fixed graphs

Now, assume that the graph G is fixed, consider BCSREACH(G)

Let G^{-} denote G with self-loops removed.

P4



C4

Complexity for fixed graphs

Now, assume that the graph G is fixed, consider BCSREACH(G)

Let G^- denote G with self-loops removed.

P4

C4

Theorem

If G^- contains C4 as induced subgraph,
then BCSREACH(G) *is NP-complete.*

Complexity for fixed graphs

Now, assume that the graph G is fixed, consider BCSREACH(G)

Let G^- denote G with self-loops removed.

P4

C4

Theorem

If G^- contains C4 as induced subgraph, then BCSREACH(G) *is NP-complete.*

Theorem

If G^- contains neither C4 nor P4 as induced subgraphs, then BCSREACH(G) *is in* P.

Conclusion

Theorem

Reachability under bounded context switching

for valence systems over graph monoids

is always in NP.

+ almost complete classification of complexity for fixed graphs.

Conclusion

Theorem

Reachability under bounded context switching

for valence systems over graph monoids

is always in NP.

+ almost complete classification of complexity for fixed graphs.

Open problems / future work:

Conclusion

Theorem

Reachability under bounded context switching

for valence systems over graph monoids

is always in NP.

\+ almost complete classification of complexity for fixed graphs.

Open problems / future work:

Complexity for valence systems over P4?

Conclusion

Theorem

Reachability under bounded context switching

for valence systems over graph monoids

is always in NP.

+ almost complete classification of complexity for fixed graphs.

Open problems / future work:

Complexity for valence systems over P4?

Bounded phase switching?

Conclusion

Theorem

Reachability under bounded context switching

for valence systems over graph monoids

is always in NP.

+ almost complete classification of complexity for fixed graphs.

Open problems / future work:

Complexity for valence systems over P4?

Bounded phase switching?

BCS for reachability games?

Conclusion

Theorem

Reachability under bounded context switching

for valence systems over graph monoids

is always in NP.

+ almost complete classification of complexity for fixed graphs.

Open problems / future work:

Complexity for valence systems over P4?

Bounded phase switching?

BCS for reachability games?

Richer model supporting queues, higher order?

Thank you!

Questions?