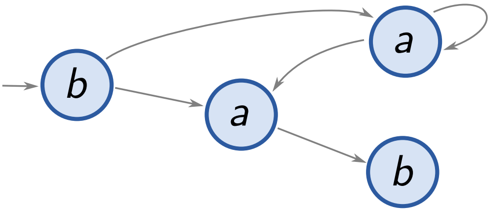
Tree Algebras and Bisimulation-Invariant MSO on Finite Graphs

Thomas Colcombet, Amina Doumane, Denis Kuperberg

CNRS, LIP, ENS Lyon

GdT Plume, 23/06/25

Transition systems



Specifying properties

MSO formulas:

$$\varphi, \psi := a(x) \mid E(x,y) \mid \exists x.\varphi \mid \exists X.\varphi \mid x \in X \mid \varphi \vee \psi \mid \neg\varphi$$

Example: $\varphi(r)$ for "$\exists$ ∞ path from r":
$\exists X.$
$r \in X \wedge$
$\forall x. x \in X \Rightarrow \exists y. E(x,y) \wedge y \in X$

Specifying properties

MSO formulas:

$$\varphi, \psi := a(x) \mid E(x,y) \mid \exists x.\varphi \mid \exists X.\varphi \mid x \in X \mid \varphi \vee \psi \mid \neg\varphi$$

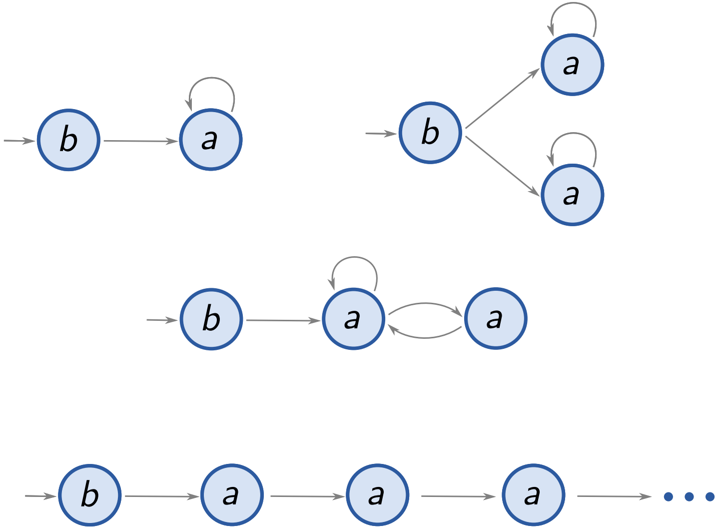
Example: $\varphi(r)$ for "$\exists$ ∞ path from r":
$\exists X.$
$r \in X \wedge$
$\forall x. x \in X \Rightarrow \exists y. E(x,y) \wedge y \in X$

μ-calculus formulas:

$$\varphi, \psi := a \mid \diamond\varphi \mid \Box\varphi \mid \mu X.\varphi \mid \nu X.\varphi \mid \varphi \vee \psi \mid \neg\varphi$$

Example: ψ for "$\exists$ ∞ path from the current vertex": $\nu X. \diamond X$
Fact: μ-calculus is bisimulation-invariant.

Bisimulation



Starting point

Theorem (Janin and Walukiewicz 1996)

For properties of systems, the following are equivalent:

1. Being MSO-definable and bisimulation-invariant.
2. Being μ-calculus-definable.

Starting point

Theorem (Janin and Walukiewicz 1996)

For properties of systems, the following are equivalent:

1. Being MSO-definable and bisimulation-invariant.
2. Being μ-calculus-definable.

μ-calculus $\rightarrow$ bisim-inv MSO : Easy

Bisim-inv MSO $\rightarrow$ μ-calculus : Hard

Proof sketch for bisim-inv $MSO \to \mu$-calculus

Let $\varphi \in$ bisim-inv MSO:

- φ is in particular a formula on **infinite trees**.

Proof sketch for bisim-inv $MSO \to \mu$-calculus

Let $\varphi \in$ bisim-inv MSO:

- φ is in particular a formula on **infinite trees**.
- $\varphi \rightsquigarrow$ automaton $\mathcal{A}$ on **infinite trees**. [Rabin 1968]

Proof sketch for bisim-inv *MSO* $\rightarrow$ μ-calculus

Let $\varphi \in$ bisim-inv MSO:

- φ is in particular a formula on **infinite trees**.
- $\varphi \rightsquigarrow$ automaton $\mathcal{A}$ on **infinite trees**. [Rabin 1968]
- $\mathcal{A} \rightsquigarrow$ μ-calculus formula ψ. [Janin-Walukiewicz 1996]

Proof sketch for bisim-inv *MSO* $\to$ μ-calculus

Let $\varphi \in$ bisim-inv MSO:

- φ is in particular a formula on **infinite trees**.
- $\varphi \rightsquigarrow$ automaton $\mathcal{A}$ on **infinite trees**. [Rabin 1968]
- $\mathcal{A} \rightsquigarrow$ μ-calculus formula ψ. [Janin-Walukiewicz 1996]

Correctness:
Infinite trees suffice to define bisim-inv properties of systems.

Finite systems

The proof of Janin-Walukiewicz needs bisim-inv on infinite systems.

Finite systems

The proof of Janin-Walukiewicz needs bisim-inv on infinite systems.

Finite model property for μ-calculus:
If ψ has a model then it has a finite one.

Finite systems

The proof of Janin-Walukiewicz needs bisim-inv on infinite systems.

Finite model property for μ-calculus:
If ψ has a model then it has a finite one.

Can we restrict the theorem to finite systems ?

Finite systems

The proof of Janin-Walukiewicz needs bisim-inv on infinite systems.

Finite model property for μ-calculus:
If ψ has a model then it has a finite one.

Can we restrict the theorem to finite systems ?

Main Contribution

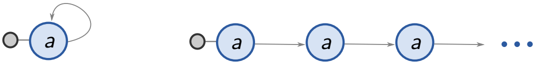
For properties of **finite** systems, the following are equivalent:

1. Being MSO-definable and bisimulation-invariant.
2. Being μ-calculus-definable.

Example of the difference

MSO formula φ for "$\exists$ cycle":

- φ is not bisim-invariant on all systems.

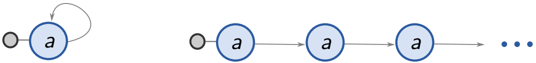


- φ is bisim-invariant on **finite** systems.
- Equivalent to $\psi = \nu X. \diamond X$ on finite systems.

Example of the difference

MSO formula φ for "$\exists$ cycle":

- φ is not bisim-invariant on all systems.



- φ is bisim-invariant on **finite** systems.
- Equivalent to $\psi = \nu X. \diamond X$ on finite systems.

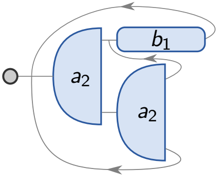
$\Longrightarrow$ using Janin-Walukiewicz does not work for finite systems.

Ranked systems

"Bisimulation = **unfold** + children duplication"

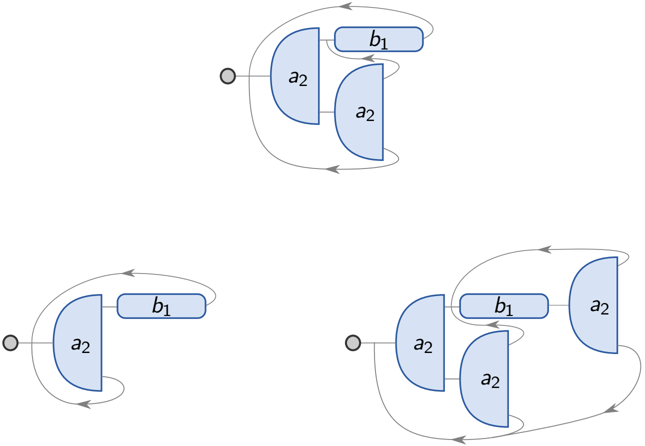
Ranked systems

“Bisimulation = **unfold** + children duplication”



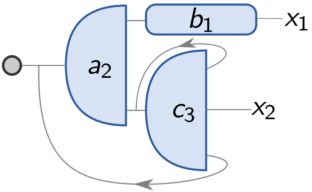
Ranked systems

"Bisimulation = **unfold** + children duplication"



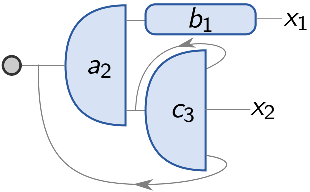
The free algebra of systems

Systems have open ports and arities:

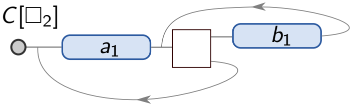


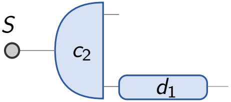
The free algebra of systems

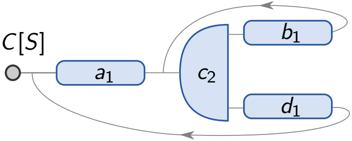
Systems have open ports and arities:



Operation: Plug into context.







Algebra: compressing information

Idea: Remember only relevant information about a system.

Algebra: compressing information

Idea: Remember only relevant information about a system.

Arity stratification $\rightsquigarrow$ Algebra $\mathcal{A} = (A_n)_{n \in \mathbb{N}}$.

Algebra: compressing information

Idea: Remember only relevant information about a system.

Arity stratification $\rightsquigarrow$ Algebra $\mathcal{A} = (A_n)_{n \in \mathbb{N}}$.

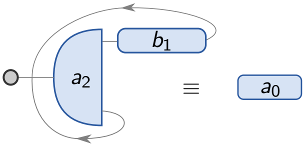
Example: $L = \{\text{Systems with an } a\}$. $A_n = \{a_n, b_n\}$

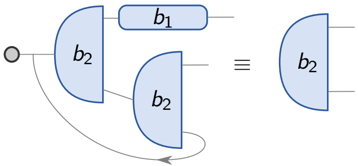
Algebra: compressing information

Idea: Remember only relevant information about a system.

Arity stratification $\rightsquigarrow$ Algebra $\mathcal{A} = (A_n)_{n\in\mathbb{N}}$.

Example: $L = \{\text{Systems with an } a\}$. $A_n = \{a_n, b_n\}$



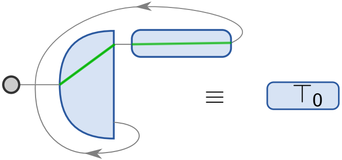


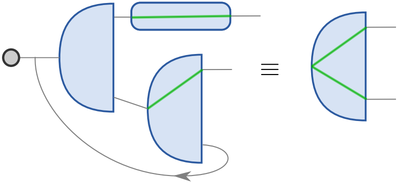
Then $L = \{\text{Systems evaluating to } a_0\}$, via h :Systems$\rightarrow \mathcal{A}$.

Another example of algebra

Language $L = \{\exists \text{ cycle containing } a\}$.

Then $A_n = \{\top_n\} \cup \mathcal{P}(\{1, \ldots, n\})$



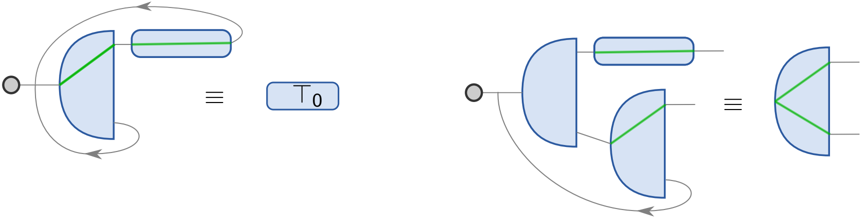


$L = h^{-1}(\top_0)$

Another example of algebra

Language $L = \{\exists \text{ cycle containing } a\}$.

Then $A_n = \{\top_n\} \cup \mathcal{P}(\{1, \dots, n\})$



$L = h^{-1}(\top_0)$

$\mathcal{A}$ is rankwise-finite and unfold-invariant.
Intuition: Enough for regularity.

Recognizability

Main Contribution 2

If L is recognized by a rankwise-finite unfold-invariant algebra, then L is recognized by some automaton model.

Recognizability

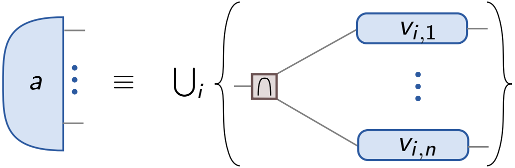
Main Contribution 2

If L is recognized by a rankwise-finite unfold-invariant algebra, then L is recognized by some automaton model.

Add operators such as intersection $\cap$ to the algebra.

Key Lemma

$\forall a \in A_n$, $\exists (v_{i,j})$ from A_1 such that:



Recognizability

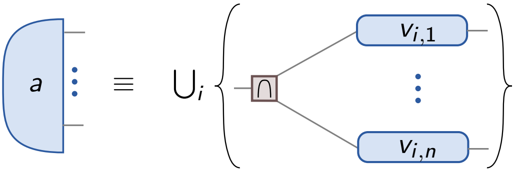
Main Contribution 2

If L is recognized by a rankwise-finite unfold-invariant algebra, then L is recognized by some automaton model.

Add operators such as intersection $\boxed{\cap}$ to the algebra.

Key Lemma

$\forall a \in A_n$, $\exists (v_{i,j})$ from A_1 such that:

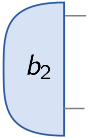


Consequences

- A_1 actually contains all the information about A_n.
- Algebras can be turned into automata.

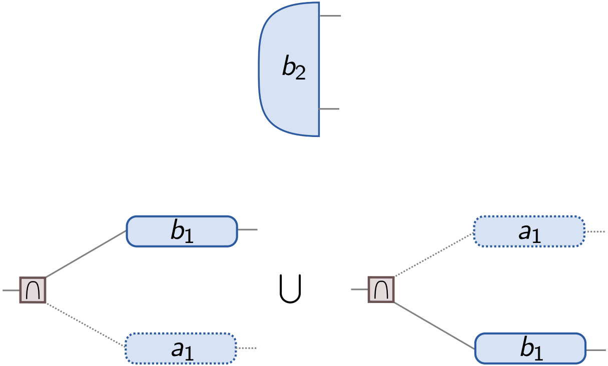
Key Lemma Example

Algebra: $A_n = \{a_n, b_n\}$, for "$\exists a$".



Key Lemma Example

Algebra: $A_n = \{a_n, b_n\}$, for "$\exists a$".



Proof of Main Theorem

μ-**calculus** $\rightarrow$ **bisim-inv MSO** is easy, same as before.

Proof of Main Theorem

μ-**calculus** $\rightarrow$ **bisim-inv MSO** is easy, same as before.

Bisim-inv MSO $\rightarrow$ μ-**calculus**:

- MSO $\rightarrow$ algebra by standard compositional methods [Feferman-Vaught 1959, Shelah 1975].

Proof of Main Theorem

μ-calculus $\to$ **bisim-inv MSO** is easy, same as before.

Bisim-inv MSO $\to$ **μ-calculus**:

- MSO $\to$ algebra by standard compositional methods [Feferman-Vaught 1959, Shelah 1975].
- Algebra $\to$ unfold-invariant automata by the key lemma.

Proof of Main Theorem

μ-**calculus** $\to$ **bisim-inv MSO** is easy, same as before.

Bisim-inv MSO $\to$ μ-**calculus**:

- MSO $\to$ algebra by standard compositional methods [Feferman-Vaught 1959, Shelah 1975].
- Algebra $\to$ unfold-invariant automata by the key lemma.
- Unfold-invariant $\to$ bisimulation-invariant automata by adding duplication as in [Janin-Walukiewicz 1996].

Proof of Main Theorem

μ-**calculus** $\rightarrow$ **bisim-inv MSO** is easy, same as before.

Bisim-inv MSO $\rightarrow$ μ-**calculus**:

- MSO $\rightarrow$ algebra by standard compositional methods [Feferman-Vaught 1959, Shelah 1975].
- Algebra $\rightarrow$ unfold-invariant automata by the key lemma.
- Unfold-invariant $\rightarrow$ bisimulation-invariant automata by adding duplication as in [Janin-Walukiewicz 1996].
- Bisim-invariant automata $\rightarrow$ μ-calculus as in [Janin-Walukiewicz 1996].

Proof of Main Theorem

μ-**calculus** $\to$ **bisim-inv MSO** is easy, same as before.

Bisim-inv MSO $\to$ μ-**calculus**:

- MSO $\to$ algebra by standard compositional methods [Feferman-Vaught 1959, Shelah 1975].
- Algebra $\to$ unfold-invariant automata by the key lemma.
- Unfold-invariant $\to$ bisimulation-invariant automata by adding duplication as in [Janin-Walukiewicz 1996].
- Bisim-invariant automata $\to$ μ-calculus as in [Janin-Walukiewicz 1996].

Thanks for your attention!