APLAS 2018

2nd-6th December 2018, Wellington, NZ

Complexity Analysis of Tree Share Structure

Xuan-Bach Le
University of Oxford

Anthony W. Lin
University of Oxford

Aquinas Hobor
School of Computing, NUS & Yale-NUS

This talk is about...

Decidability and complexity results of
tree-like permission constraints
for concurrent programs

Permission accounting in concurrent programs

- Specify resource ownership of threads

Permission accounting in concurrent programs

- Specify resource ownership of threads
  - Example: Thread A creates lock L
    → A has full ownership of L.

Permission accounting in concurrent programs

- Specify resource ownership of threads
  - Example: Thread A creates lock L
    → A has full ownership of L.
- Assert the ownership transfer mechanism

Permission accounting in concurrent programs

- Specify resource ownership of threads
  - Example: Thread A creates lock L
    → A has full ownership of L.
- Assert the ownership transfer mechanism
  - Example: Thread A forks B and shares L with B
    → B should has partial ownership of L.

Permission reasoning in program verification

- Checking interference with fractional permissions (Boyland, SAS 2003).
- Permission accounting in separation logic (Bornat et al., POPL 2005).
- A fresh look at separation algebras and share accounting (Dockins et al., APLAS 2009).
- A Symbolic Approach to Permission Accounting for Concurrent Reasoning (Huisman & Mostowski, ISPDC 2015).
- Threads as resource for concurrency verification (Le et al., PEPM 2015).
- Viper: A Verification Infrastructure for Permission-Based Reasoning (Muller et al., VMCAI 2016).
- On Symbolic Heaps Modulo Permission Theories (Demri et al., FSTTCS 2017).
- Permission inference for array programs (Dohrau et al., CAV 2018).
- Logical reasoning over disjoint fractional permissions (Le et al, ESOP 2018).

Permissions for Concurrent Separation Logic

- Integers : {..., -2, -1, 0, 1, 2, ...}
- Rationals: { 0, 1/2, 1/4, ...}
- Symbolic: object references

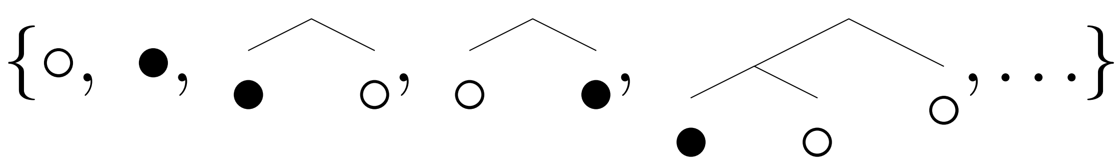
Permissions for Concurrent Separation Logic

- Integers : {..., -2, -1, 0, 1, 2, …}
- Rationals: { 0, 1/2, 1/4, ...}
- Symbolic: object references

Don‘t preserve the disjointness property of Separation Logic, which is crucial for modular reasoning!

Permissions for Concurrent Separation Logic

- Integers : {..., -2, -1, 0, 1, 2, …}
- Rationals: { 0, 1/2, 1/4, ...}
- Symbolic: object references
- Tree shares:



Don‘t preserve the disjointness property of Separation Logic, which is crucial for modular reasoning!

Previous works

- A fresh look at seperation algebras and share accounting (Dockins & Hobor & Appel, APLAS 2009).
- Decision procedures over sophisticated fractional permissions (Le & Gherghina & Hobor, APLAS 2012).
- Decidability and complexity of tree shares formulas (Le & Lin & Hobor, FSTTCS 2016).
- A certified decision procedure for tree shares (Le & Nguyen & Hobor & Chin, ICFEM 2017).
- Logical reasoning over disjoint fractional permissions (Le & Hobor, ESOP 2018).

Previous works

- A fresh look at seperation algebras and share accounting (Dockins & Hobor & Appel, APLAS 2009).

Decision procedures +

practical integration into Separation Logic

- Logical reasoning over disjoint fractional permissions (Le & Hobor, ESOP 2018).

This time

- Tight complexity bound for:
  - Tree share Boolean structure.
  - Tree share multiplication structure.
- Combined structure has non-elementary complexity.

This time

- Tight complexity bound for:
  - Tree share Boolean structure.
  - Tree share multiplication structure.

- Combined structure has non-elementary complexity.

Upper complexity bounds provide complete decision procedures.

Agenda

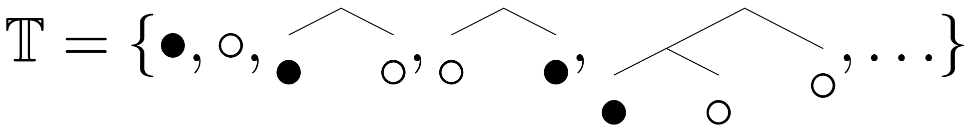
1. Introduction
2. Complexity for Boolean structure
3. Complexity for multiplication structure
4. Non-elementary bound for combined structure
5. Conclusion

Agenda

1. Introduction
2. Complexity for Boolean structure
3. Complexity for multiplication structure
4. Non-elementary bound for combined structure
5. Conclusion

Boolean structure

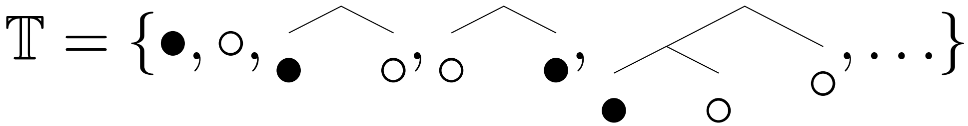
$$\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot} \rangle$$

- Domain: 
  - Canonical form:

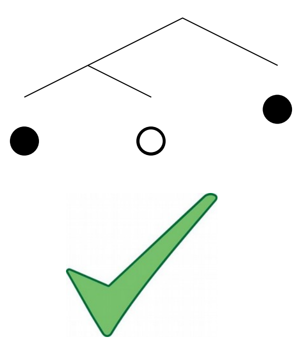
Boolean structure

$$\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot} \rangle$$

- Domain:



  - Canonical form:

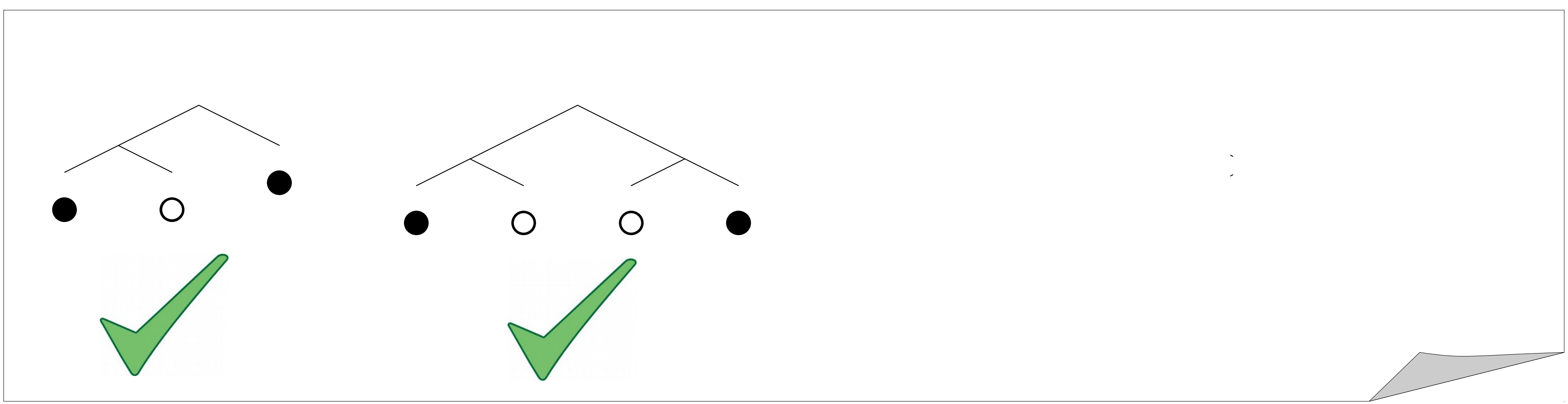


Boolean structure

$$\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot} \rangle$$

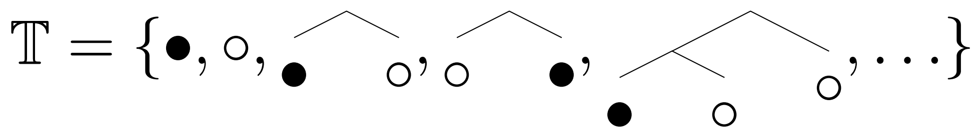
- Domain: $\mathbb{T} = \{\bullet, \circ, (\bullet\ \circ), (\circ\ \bullet), ((\bullet\ \circ)\ \circ), \ldots\}$
  - Canonical form:

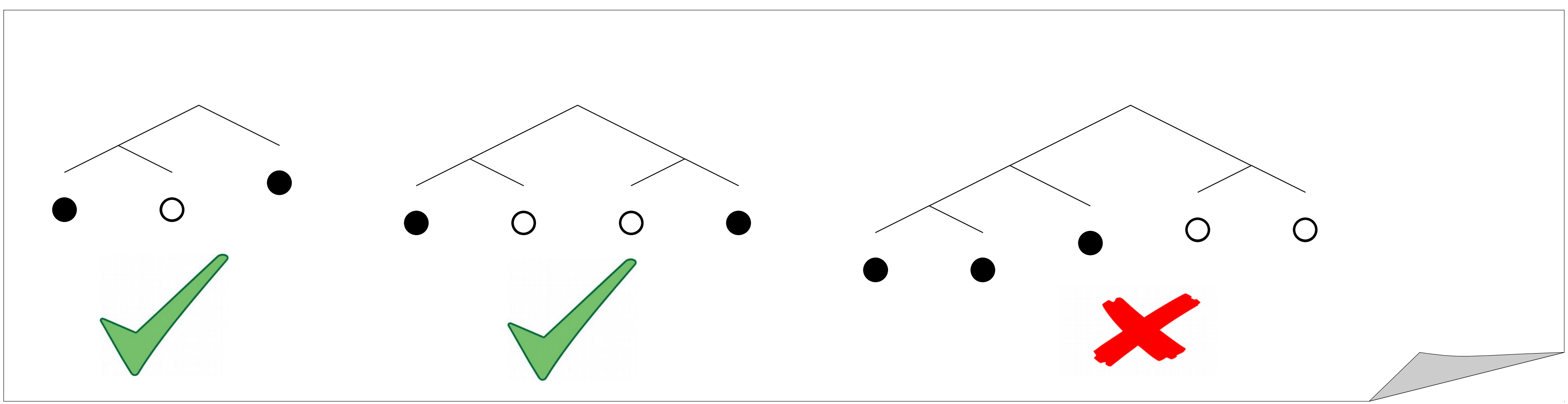
$$(\bullet\ \bullet) \cong \bullet \qquad (\circ\ \circ) \cong \circ$$



Boolean structure

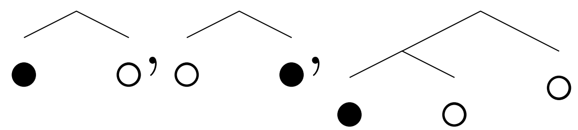
$$\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot} \rangle$$

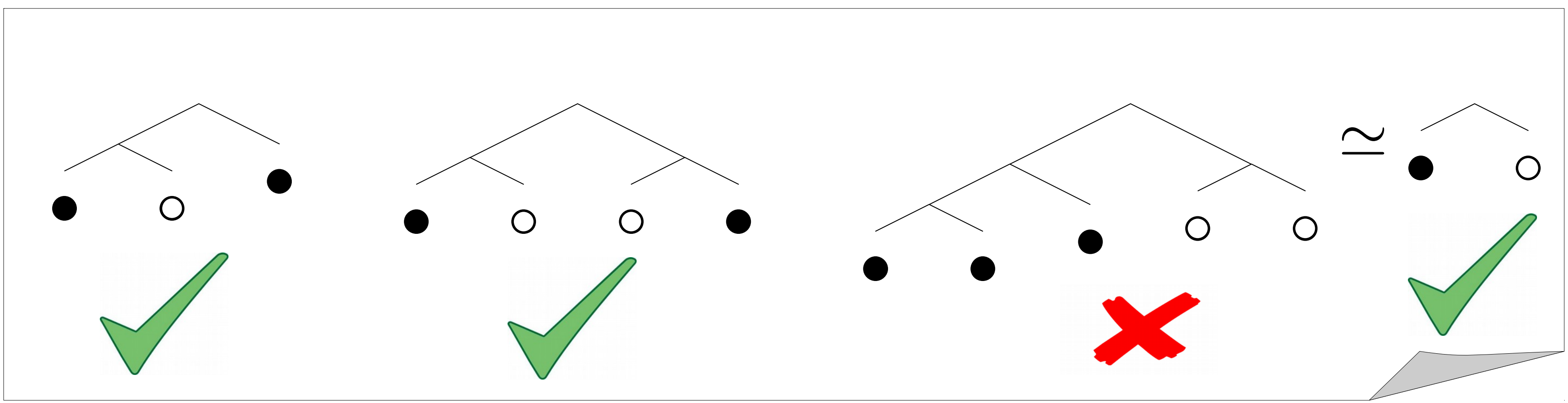
- Domain: 
  - Canonical form:



Boolean structure

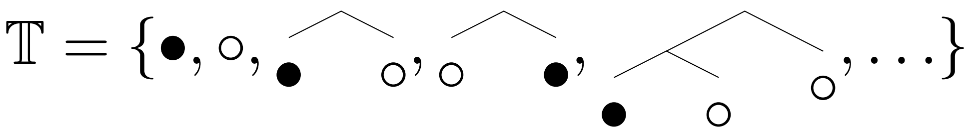
$$\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot} \rangle$$

- Domain: $\mathbb{T} = \{\bullet, \circ,$  $, \ldots\}$
  - Canonical form:

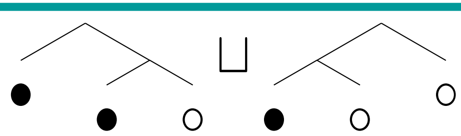


Boolean structure

$$\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot} \rangle$$

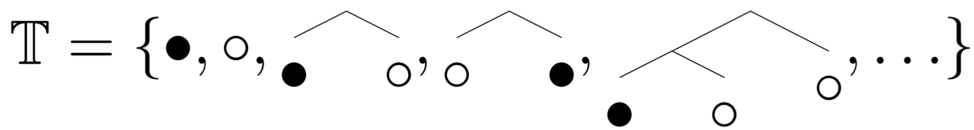
- Domain: 
  - Canonical form:

- Union:

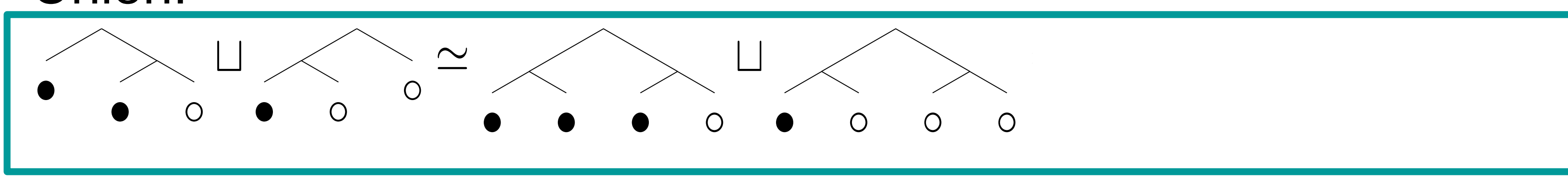


Boolean structure

$$\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot} \rangle$$

- Domain: 
  - Canonical form:

- Union:

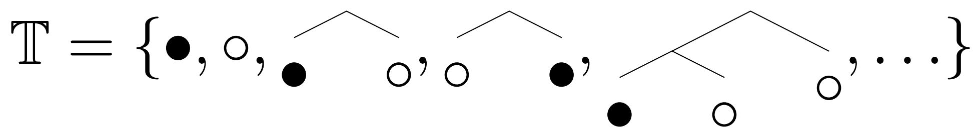


Unfold

Boolean structure

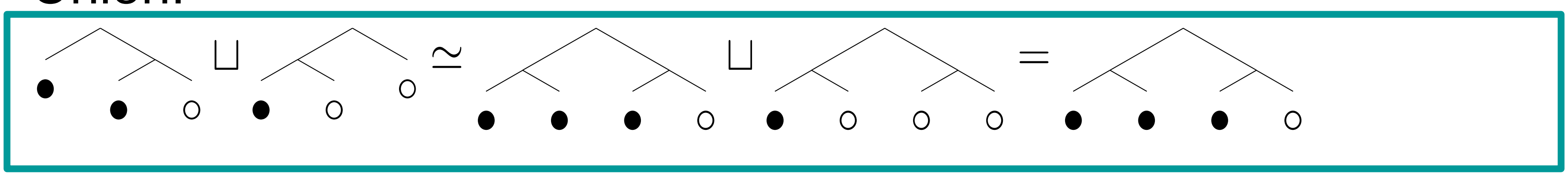
$$\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot} \rangle$$

- Domain:



  - Canonical form:

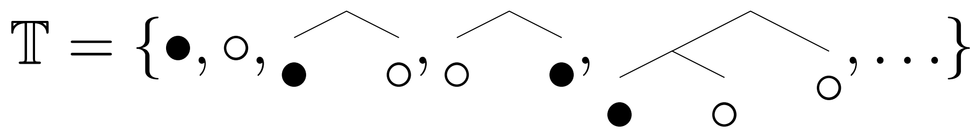
- Union:



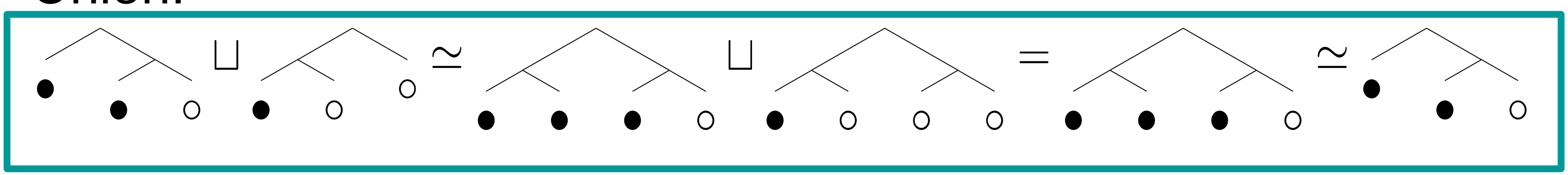
Union leaf-wise

Boolean structure

$$\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot} \rangle$$

- Domain: 
  - Canonical form:

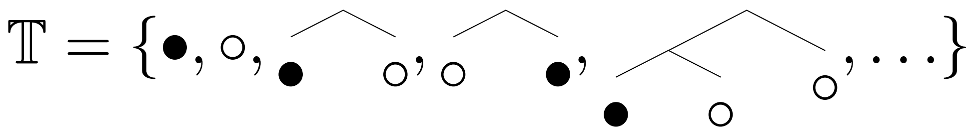
- Union:



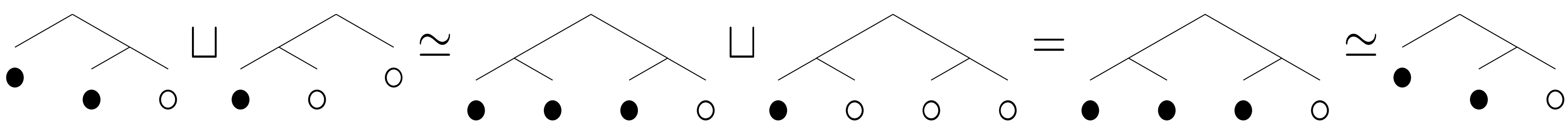
Fold

Boolean structure

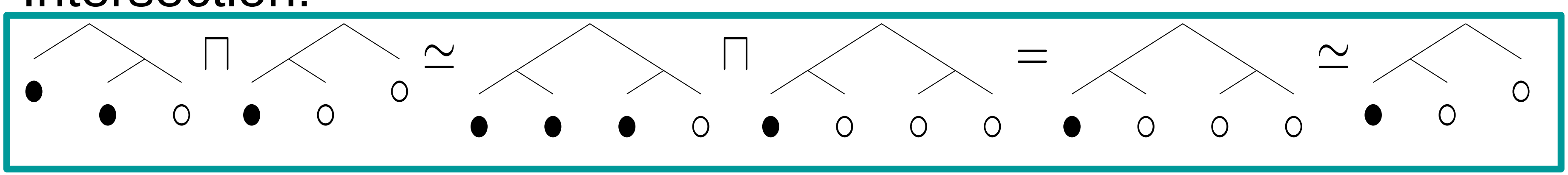
$$\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot} \rangle$$

- Domain: 
  - Canonical form:

- Union:

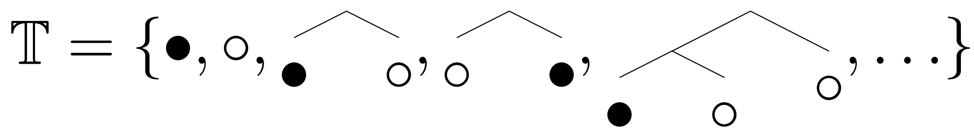


- Intersection:

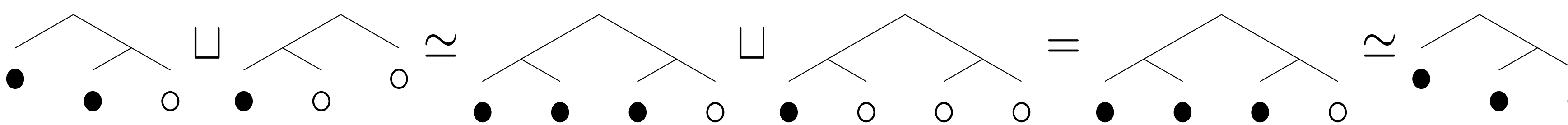


Boolean structure

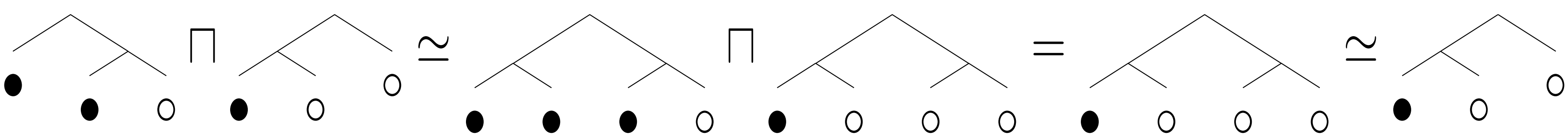
$$\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot} \rangle$$

- Domain: 
  - Canonical form:

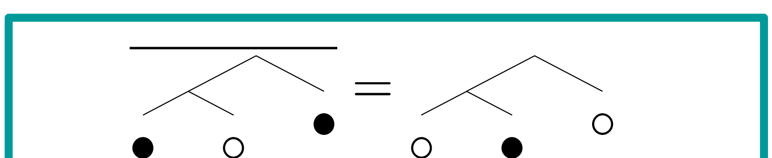
- Union:



- Intersection:

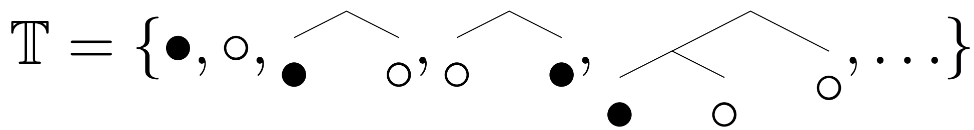


- Complement:

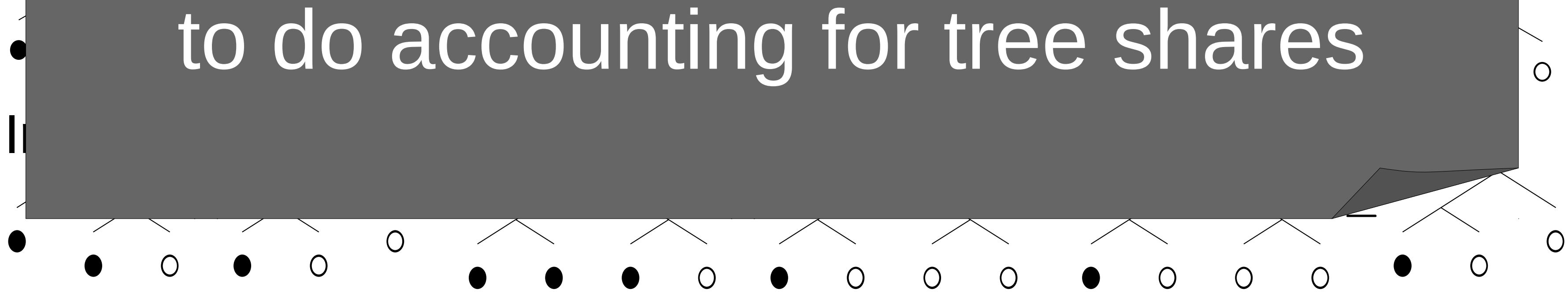


Boolean structure

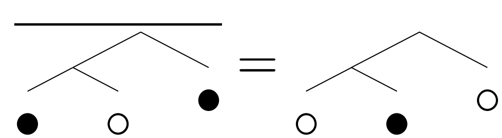
$$\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot} \rangle$$

- Domain: 
  - Canonical form:

These operators allow us
to do accounting for tree shares



- Complement:



Main result

(Le et al., 2016) The first-order complexity of $\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot} \rangle$ is $\mathsf{STA}(*, 2^{n^{O(1)}}, n)$-complete for restricted constants $\{\bullet, \circ\}$ only.

$\mathsf{STA}(*, 2^{n^{O(1)}}, n)$: Alternating Turing machines that use at most $2^{n^{O(1)}}$ time and n alternations between universal and existential states.

Main result

(Le et al., 2016) The first-order complexity of $\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot} \rangle$ is $\mathsf{STA}(*, 2^{n^{O(1)}}, n)$-complete for restricted constants $\{\bullet, \circ\}$ only.

21% of the tree share constraints
from HIP/SLEEK
contain other tree share constants

$\mathsf{STA}(*, 2^{n^{O(1)}}, n)$: Alternating Turing machines that use at most $2^{n^{O(1)}}$ time and n alternations between universal and existential states.

Main result

(Le et al., 2016) The first-order complexity of $\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot} \rangle$ is $\mathsf{STA}(*, 2^{n^{O(1)}}, n)$-complete for restricted constants $\{\bullet, \circ\}$ only.

Theorem 1. The first-order complexity of $\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot} \rangle$ is $\mathsf{STA}(*, 2^{n^{O(1)}}, n)$-complete, even with arbitrary tree share constants.

$\mathsf{STA}(*, 2^{n^{O(1)}}, n)$: Alternating Turing machines that use at most $2^{n^{O(1)}}$ time and n alternations between universal and existential states.

Main result

Decision procedure can now handle complex tree share constraints with arbitrary constants

Theorem 1. The first-order complexity of $\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot} \rangle$ is $\mathsf{STA}(*, 2^{n^{O(1)}}, n)$-complete, even with arbitrary tree share constants.

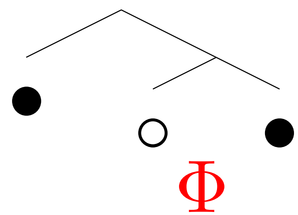
$\mathsf{STA}(*, 2^{n^{O(1)}}, n)$: Alternating Turing machines that use at most $2^{n^{O(1)}}$ time and n alternations between universal and existential states.

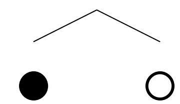
Key idea

- Transform a Boolean tree formula into equivalent formula whose constants are $\{\bullet, \circ\}$.

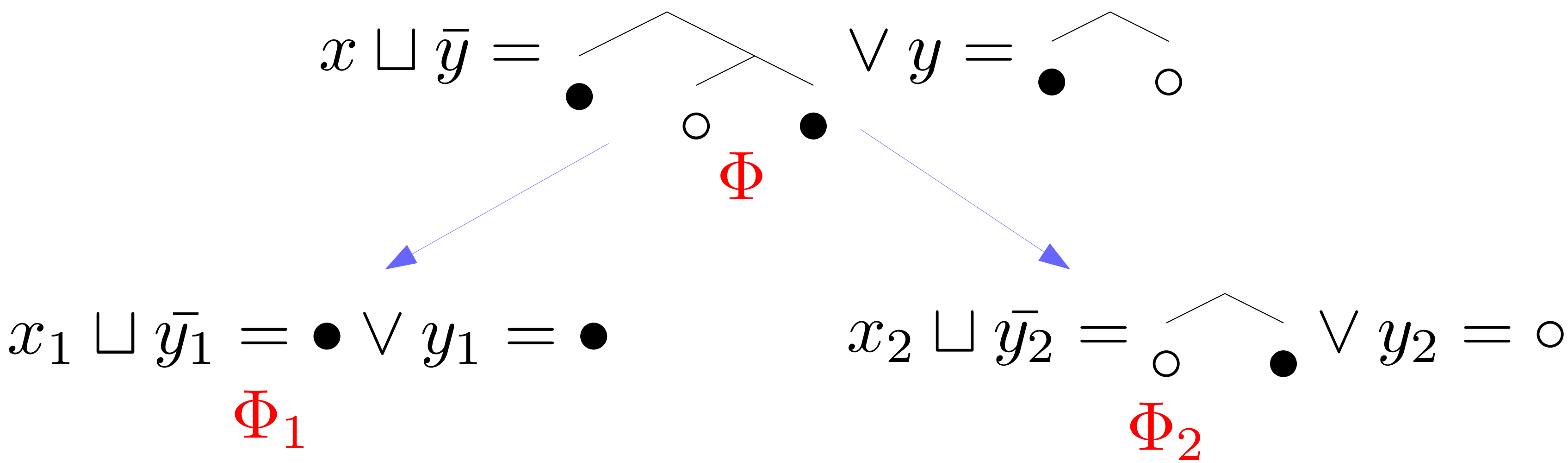
- The transformation only takes $O(n^2)$ time.

Example

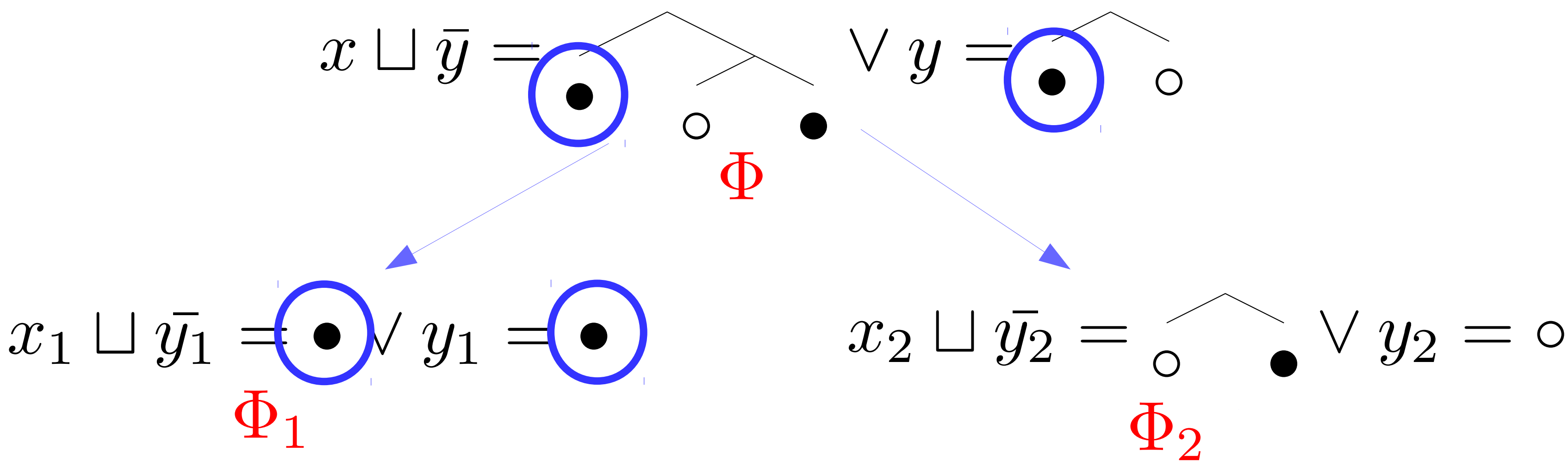
$x \sqcup \bar{y} =$ 

$\vee\ y =$ 

Example

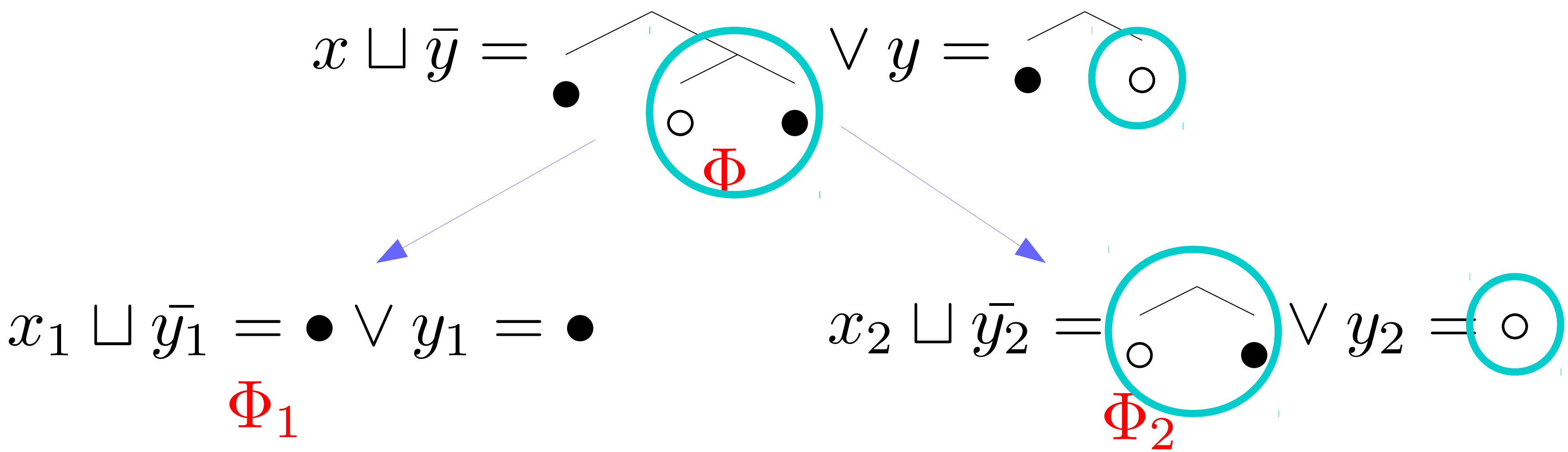

$x \sqcup \bar{y} =$
$\vee\ y =$
Φ
$x_1 \sqcup \bar{y_1} = \bullet \vee y_1 = \bullet$
Φ_1
$x_2 \sqcup \bar{y_2} =$
$\vee\ y_2 = \circ$
Φ_2

Example



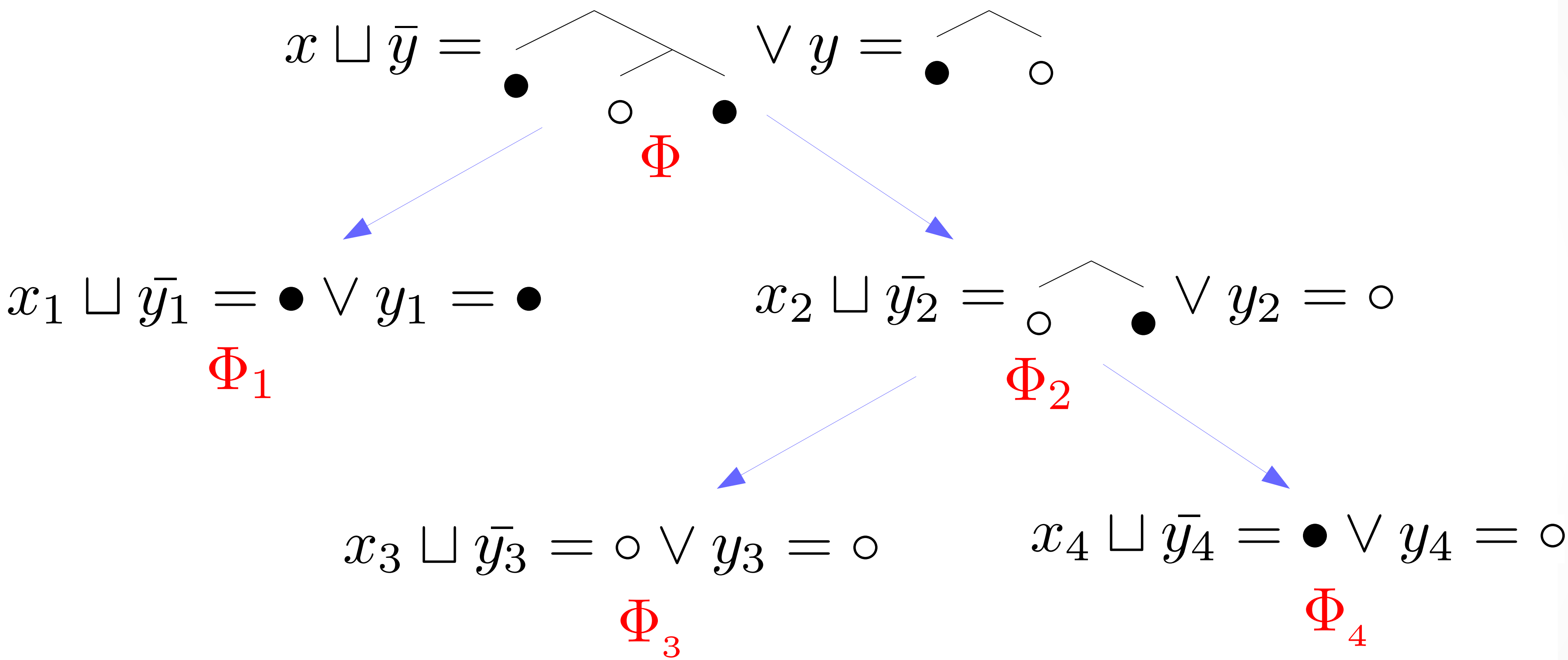
$x \sqcup \bar{y} =$
$\vee\ y =$
Φ
$x_1 \sqcup \bar{y_1} =$
$\vee\ y_1 =$
Φ_1
$x_2 \sqcup \bar{y_2} =$
$\vee\ y_2 =$
Φ_2

Example



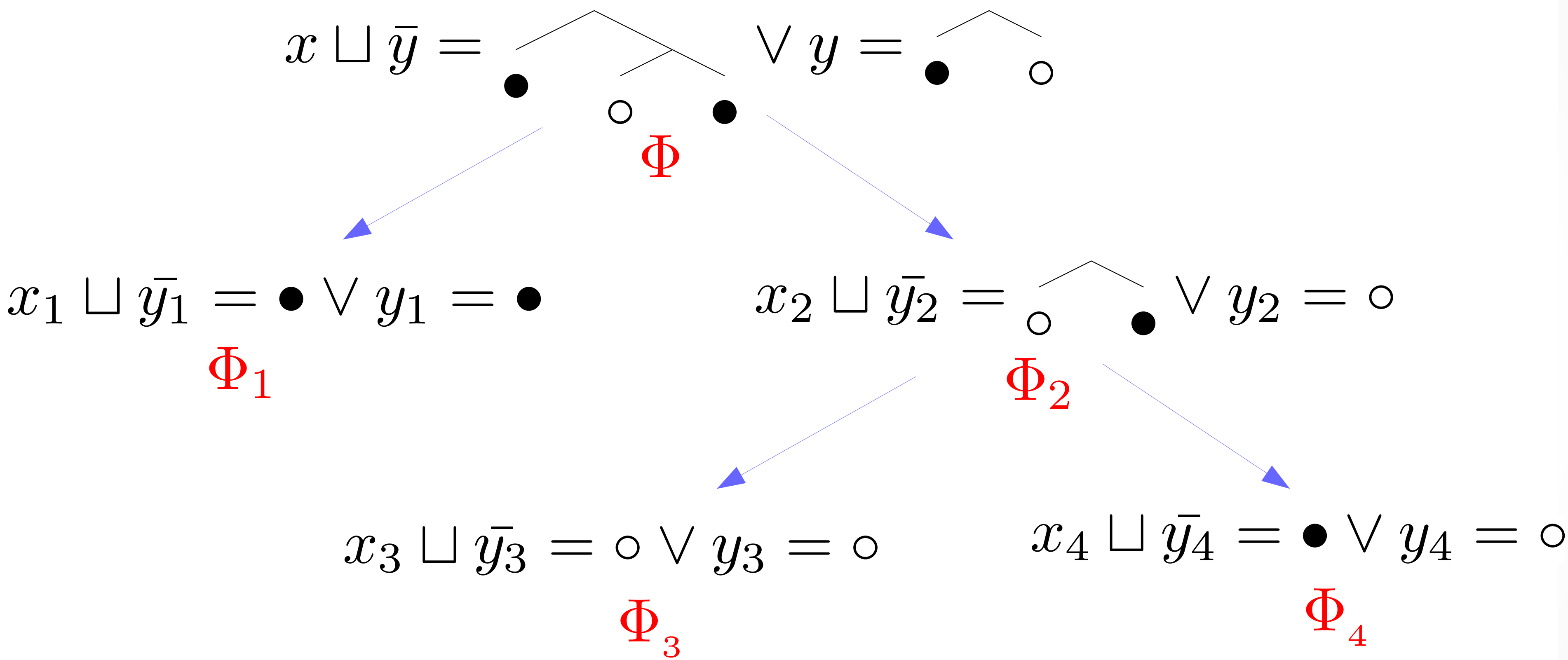
$x \sqcup \bar{y} =$
$\vee\ y =$
Φ
$x_1 \sqcup \bar{y_1} = \bullet \vee y_1 = \bullet$
Φ_1
$x_2 \sqcup \bar{y_2} =$
$\vee\ y_2 =$
Φ_2

Example



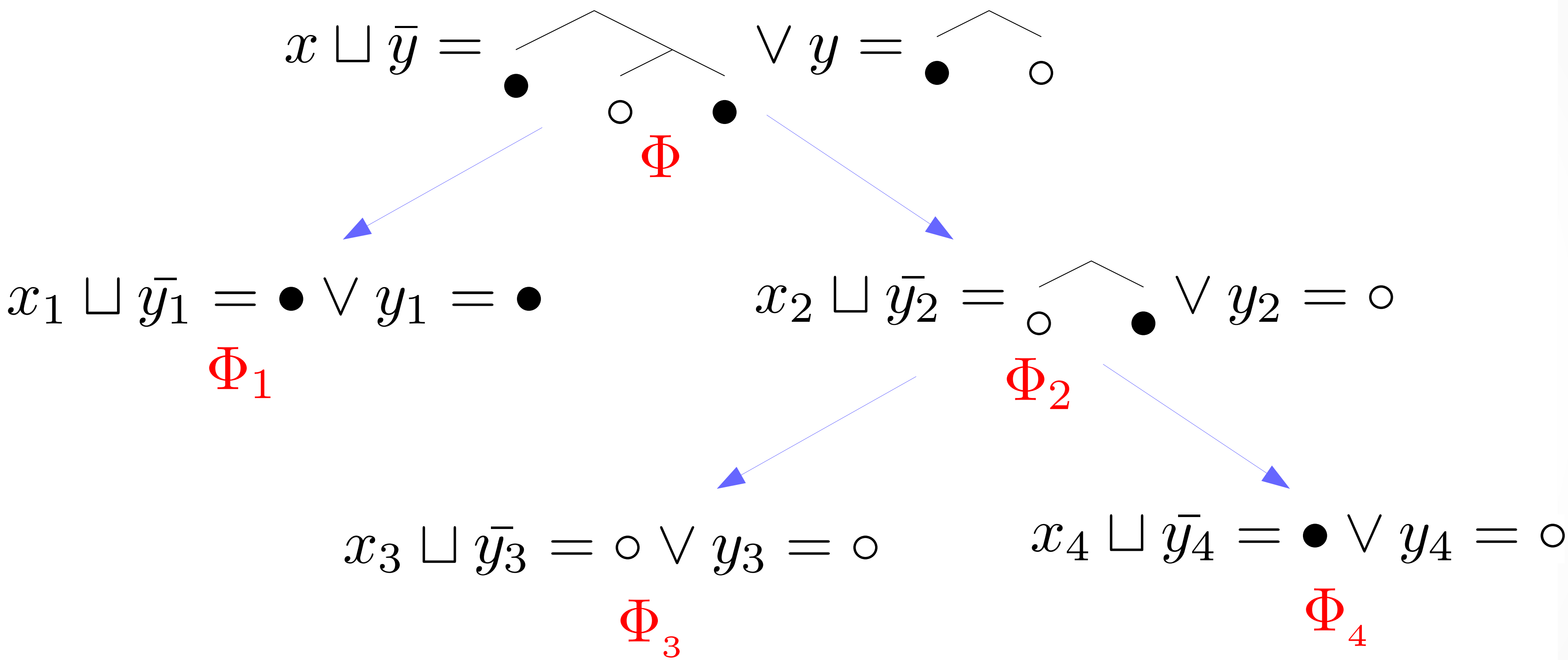
Φ
$x_1 \sqcup \bar{y_1} = \bullet \vee y_1 = \bullet$
Φ1
Φ2
$x_3 \sqcup \bar{y_3} = \circ \vee y_3 = \circ$
Φ3
$x_4 \sqcup \bar{y_4} = \bullet \vee y_4 = \circ$
Φ4

Example



$$\Phi \equiv \Phi_1 \wedge \Phi_3 \wedge \Phi_4$$

Example



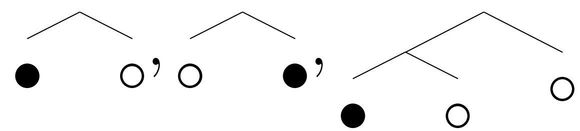
$$\forall x \exists y.\ \Phi \quad \equiv \forall x_1 \forall x_3 \forall x_4 \exists y_1 \exists y_3 \exists y_4.\ \Phi_1 \wedge \Phi_3 \wedge \Phi_4$$

Agenda

1. Introduction
2. Complexity for Boolean structure
3. Complexity for multiplication structure
4. Non-elementary bound for combined structure
5. Conclusion

Multiplication structure

$$\langle \mathbb{T}, \bowtie \rangle$$

- Domain: $\mathbb{T} = \{\bullet, \circ,$  $, \ldots\}$

- Multiplication:
  - $\tau_1 \bowtie \tau_2$: replace each $\bullet$ in τ_1 with τ_2 .

Multiplication structure

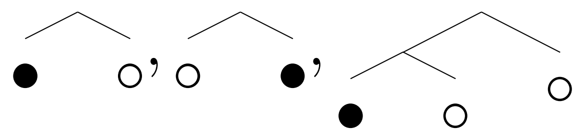
$$\langle \mathbb{T}, \bowtie \rangle$$

Bowtie is analogous to rational multiplication

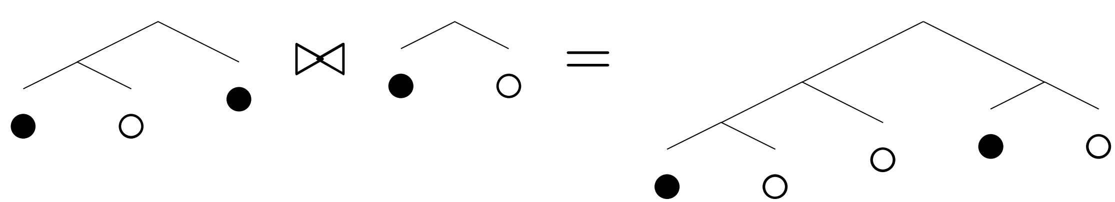
- $\tau_1 \bowtie \tau_2$: replace each • in τ_1 with τ_2 .

Multiplication structure

$$\langle \mathbb{T}, \bowtie \rangle$$

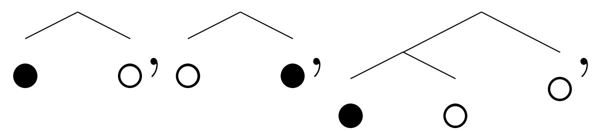
- Domain: $\mathbb{T} = \{\bullet, \circ,$  $, \ldots\}$

- Multiplication:
  - $\tau_1 \bowtie \tau_2$: replace each $\bullet$ in τ_1 with τ_2 .
  - Example:

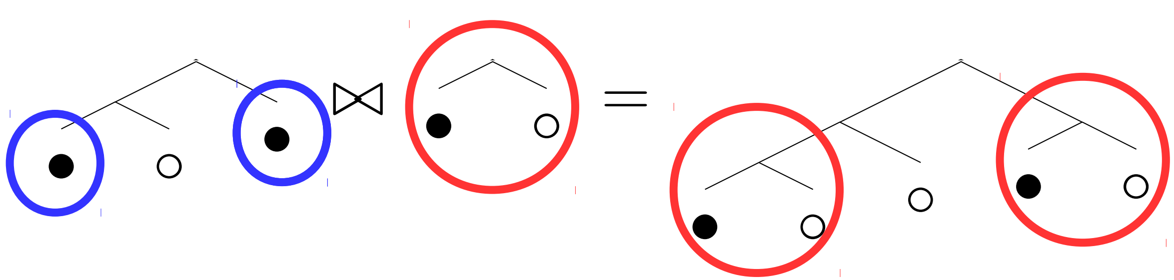


Multiplication structure

$$\langle \mathbb{T}, \bowtie \rangle$$

- Domain: $\mathbb{T} = \{\bullet, \circ,$  $, \ldots\}$

- Multiplication:
  - $\tau_1 \bowtie \tau_2$: replace each $\bullet$ in τ_1 with τ_2 .
  - Example:



Main result

(Le et al., 2016) The existential theory of $\langle \mathbb{T}, \bowtie \rangle$ is NP-hard and in PSPACE while its FO theory is undecidable.

Main result

(Le et al., 2016) The existential theory of $\langle \mathbb{T}, \bowtie \rangle$ is NP-hard and in PSPACE while its FO theory is undecidable.

In practice, we need
more than just existential formulas

Main result

(Le et al., 2016) The existential theory of $\langle \mathbb{T}, \bowtie \rangle$ is NP-hard and in PSPACE while its FO theory is undecidable.

Theorem 2. The FO theory of $\langle \mathbb{T}, \bowtie_\tau, {}_\tau\bowtie \rangle$ is $\mathsf{STA}(*, 2^{O(n)}, n)$-complete.

$\langle \mathbb{T}, \bowtie_\tau, {}_\tau\bowtie \rangle$: one of the operands is constant.

Main result

$$\bowtie_\tau (x) = x \bowtie \tau$$

Right bowtie

$${}_\tau\!\bowtie(x) = \tau \bowtie x$$

Left bowtie

Theorem 2. The FO theory of $\langle \mathbb{T}, \bowtie_\tau, {}_\tau\!\bowtie \rangle$ is $\mathsf{STA}(*, 2^{O(n)}, n)$-complete.

$\langle \mathbb{T}, \bowtie_\tau, {}_\tau\!\bowtie \rangle$: one of the operands is constant.

Key idea

- There exists an isomorphism between $\langle \mathbb{T}, \bowtie_{\tau}, {}_{\tau}\bowtie \rangle$ and the string structure $\langle \{0,1,2\}^*, P_t, S_t \rangle$

Key idea

- There exists an isomorphism between $\langle \mathbb{T}, \bowtie_{\mathcal{T}}, {}_{\mathcal{T}}\bowtie \rangle$ and the string structure $\langle \{0,1,2\}^*, P_t, S_t \rangle$
  - Domain: $\epsilon, 0, 1, 2, 00, 01, \ldots$
  - Prefix relation: $P_t(x) = tx$, e.g. $P_{01}(21) = 0121$
  - Suffix relation: $S_t(x) = xt$, e.g. $S_{01}(21) = 2101$
  - First-order complexity:

    STA$(*, 2^{O(n)}, n)$-complete (Tatiana Rybina & Andrei Voronkov, ICALP 2003)

Key idea

- There exists an isomorphism between $\langle \mathbb{T}, \bowtie_\tau, {}_\tau\bowtie \rangle$ and the string structure $\langle \{0,1,2\}^*, P_t, S_t \rangle$
  - Domain: $\epsilon, 0, 1, 2, 00, 01, \ldots$
  - Prefix relation: $P_t(x) = tx$, e.g. $P_{01}(21) = 0121$
  - Suffix relation: $S_t(x) = xt$, e.g. $S_{01}(21) = 2101$
  - First-order complexity:

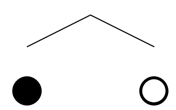
    STA$(*, 2^{O(n)}, n)$-complete (Tatiana Rybina & Andrei Voronkov, ICALP 2003)
- The isomorphism and its inverse can be efficiently constructed on-the-fly.

Key idea

- Suppose $x \bowtie (\tau_1 \bowtie \tau_2 \bowtie \tau_3 \bowtie \ldots \bowtie \tau_n) = y$ where each τ_i is a prime tree.

Key idea

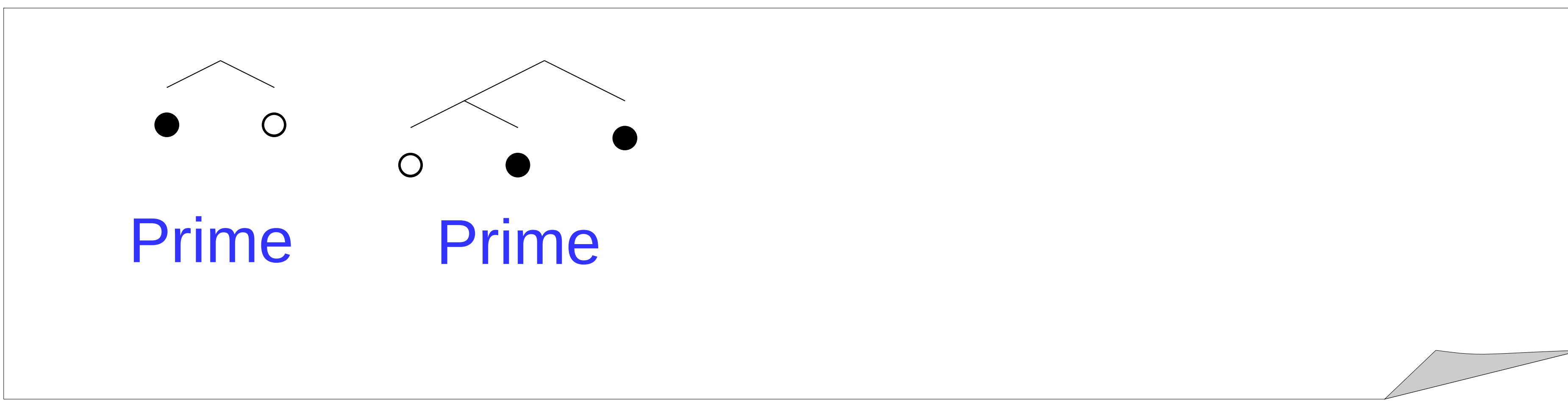
- Suppose $x \bowtie (\tau_1 \bowtie \tau_2 \bowtie \tau_3 \bowtie \ldots \bowtie \tau_n) = y$ where each τ_i is a prime tree.



Prime

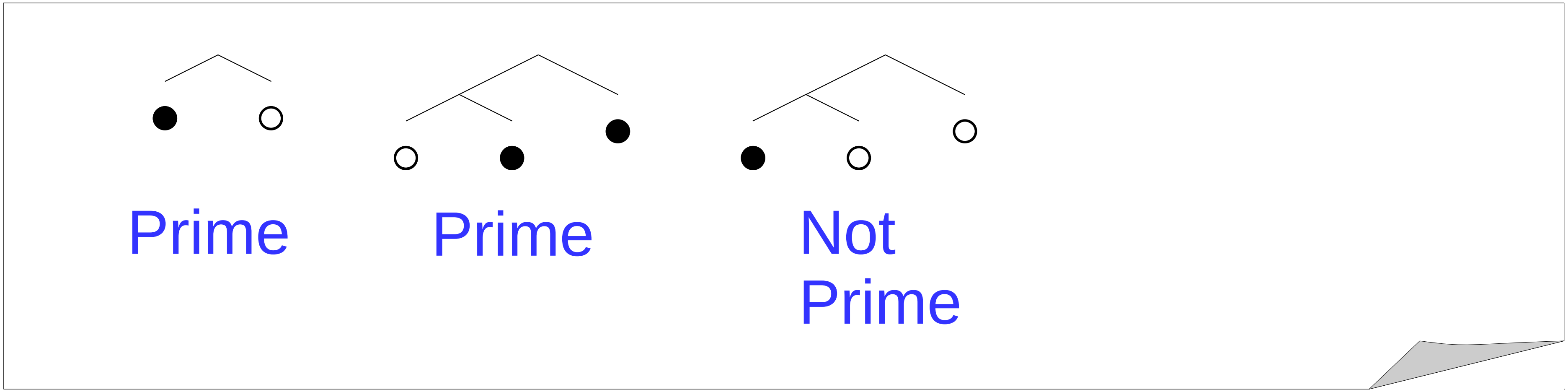
Key idea

- Suppose $x \bowtie (\tau_1 \bowtie \tau_2 \bowtie \tau_3 \bowtie \ldots \bowtie \tau_n) = y$ where each τ_i is a prime tree.



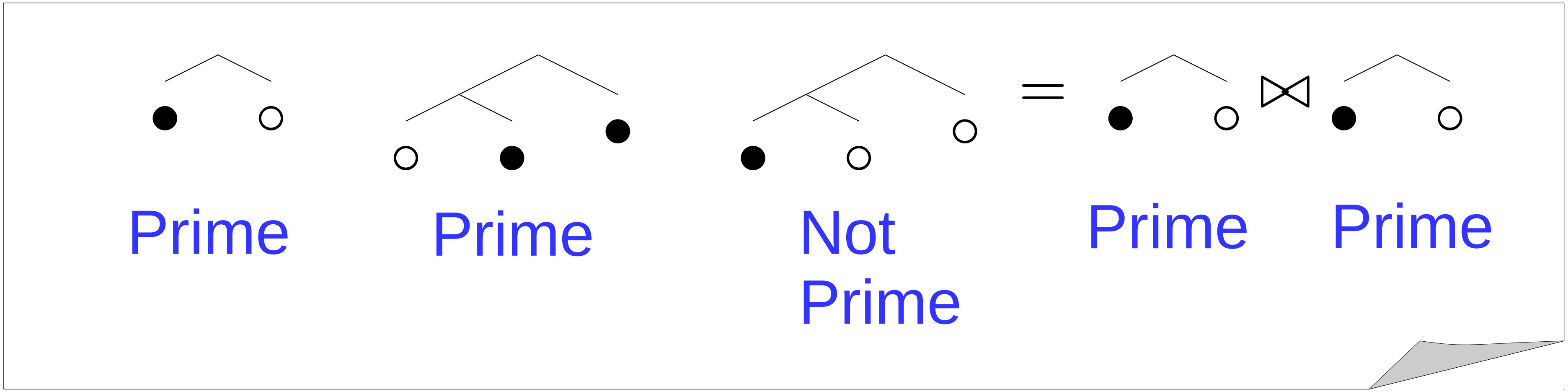
Key idea

- Suppose $x \bowtie (\tau_1 \bowtie \tau_2 \bowtie \tau_3 \bowtie \ldots \bowtie \tau_n) = y$ where each τ_i is a prime tree.



Key idea

- Suppose $x \bowtie (\tau_1 \bowtie \tau_2 \bowtie \tau_3 \bowtie \ldots \bowtie \tau_n) = y$ where each τ_i is a prime tree.



Key idea

- Suppose $x \bowtie (\tau_1 \bowtie \tau_2 \bowtie \tau_3 \bowtie \ldots \bowtie \tau_n) = y$ where each τ_i is a prime tree.
- Map each distinct τ_i to a binary string {0,1}* in increasing length-lexicalgraphic order:

$$\tau_1 \mapsto \epsilon \qquad \tau_2 \mapsto 0 \qquad \tau_3 \mapsto 1 \qquad \tau_4 \mapsto 00 \quad \ldots$$
$$\bowtie \quad \mapsto 2$$

Key idea

- Suppose $x \bowtie (\tau_1 \bowtie \tau_2 \bowtie \tau_3 \bowtie \ldots \bowtie \tau_n) = y$ where each τ_i is a prime tree.
- Map each distinct τ_i to a binary string {0,1}* in increasing length-lexicalgraphic order:

$$\tau_1 \mapsto \epsilon \quad \tau_2 \mapsto 0 \quad \tau_3 \mapsto 1 \quad \tau_4 \mapsto 00 \quad \ldots$$
$$\bowtie \quad \mapsto 2$$

- Then $x 2 \epsilon 20212002 \ldots = y$

Key idea

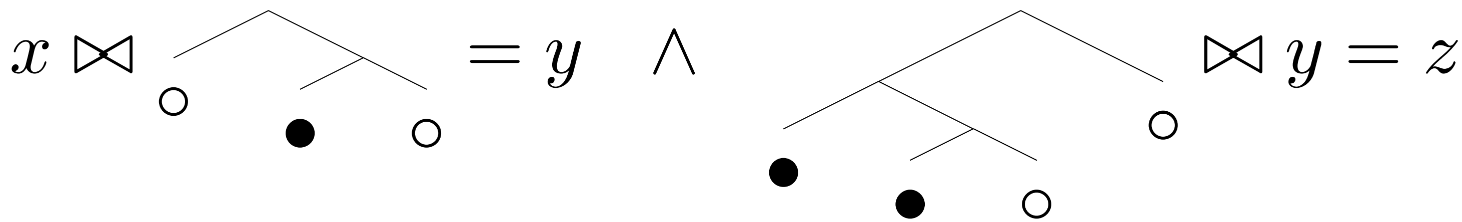
- Suppose $x \bowtie (\tau_1 \bowtie \tau_2 \bowtie \tau_3 \bowtie \ldots \bowtie \tau_n) = y$ where each τ_i is a prime tree.
- Map each distinct τ_i to a binary string {0,1}* in increasing length-lexicalgraphic order:

$$\tau_1 \mapsto \epsilon \quad \tau_2 \mapsto 0 \quad \tau_3 \mapsto 1 \quad \tau_4 \mapsto 00 \quad \ldots$$
$$\bowtie \quad \mapsto 2$$

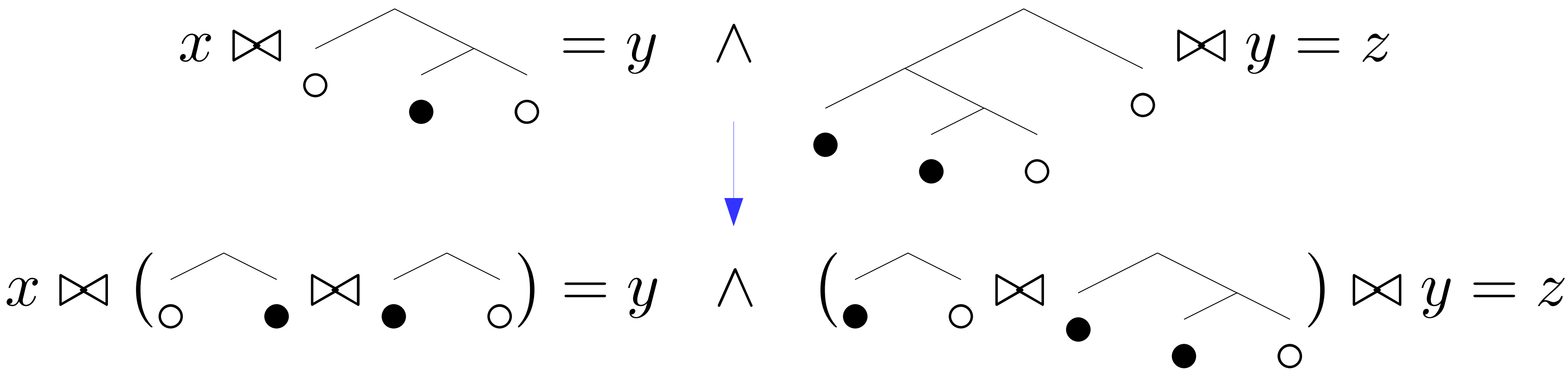
- Then $x2\epsilon20212002\ldots = y$

Similar to left-bowtie and can be generalized to arbitrary formula

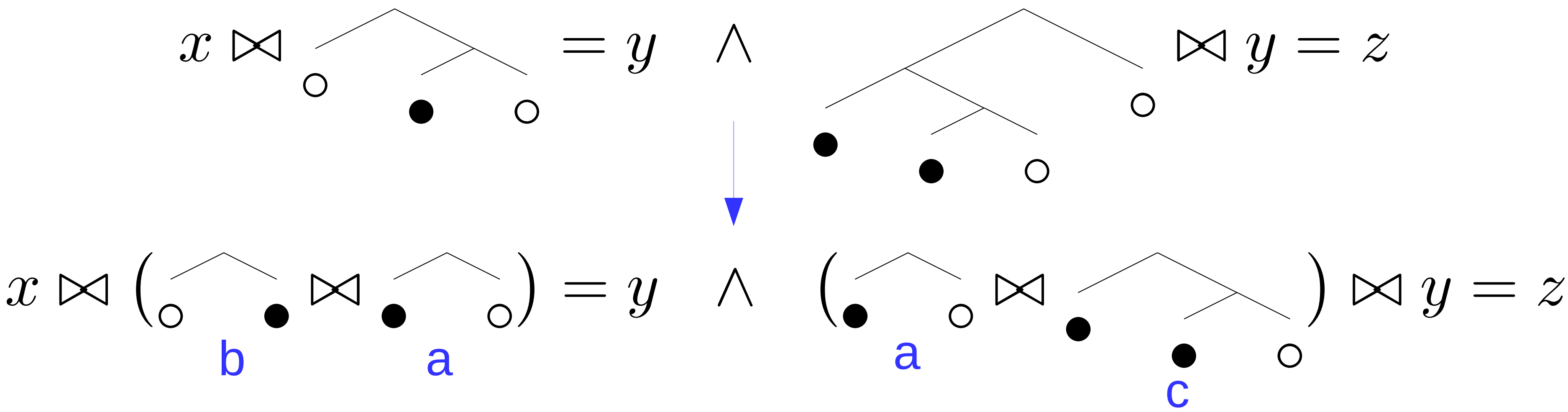
Example



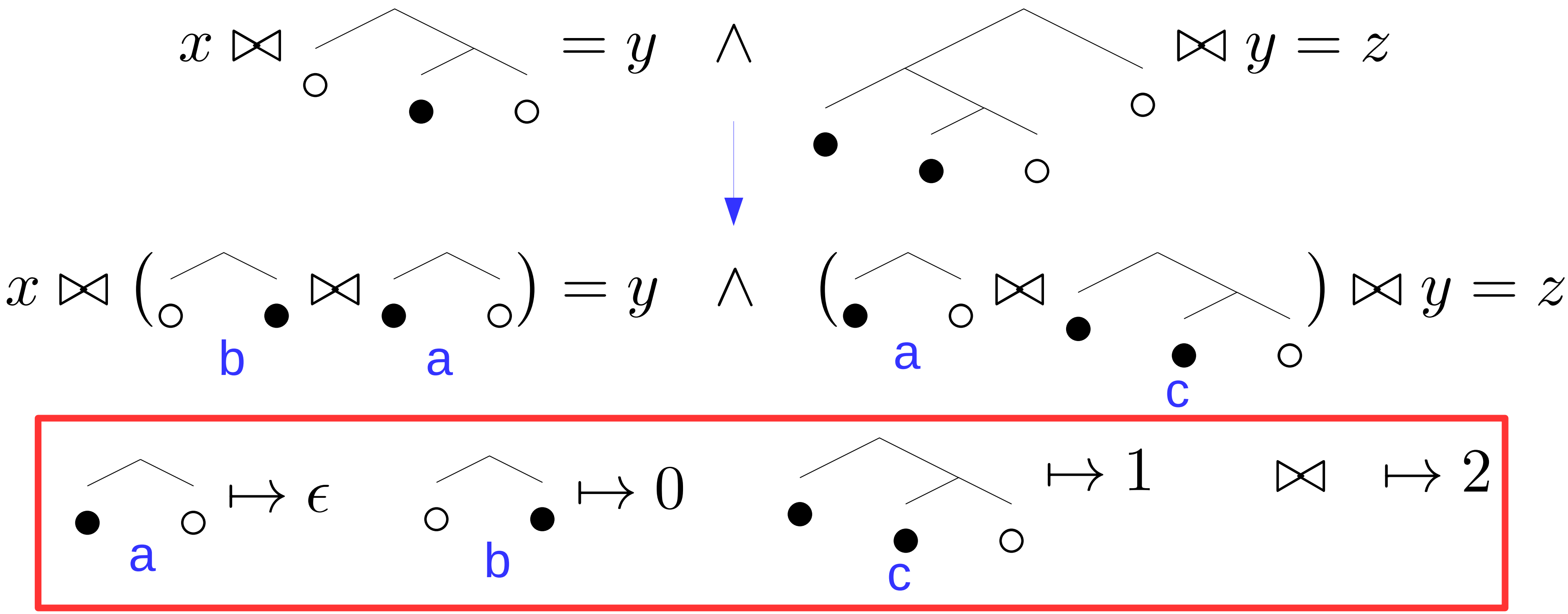
Example



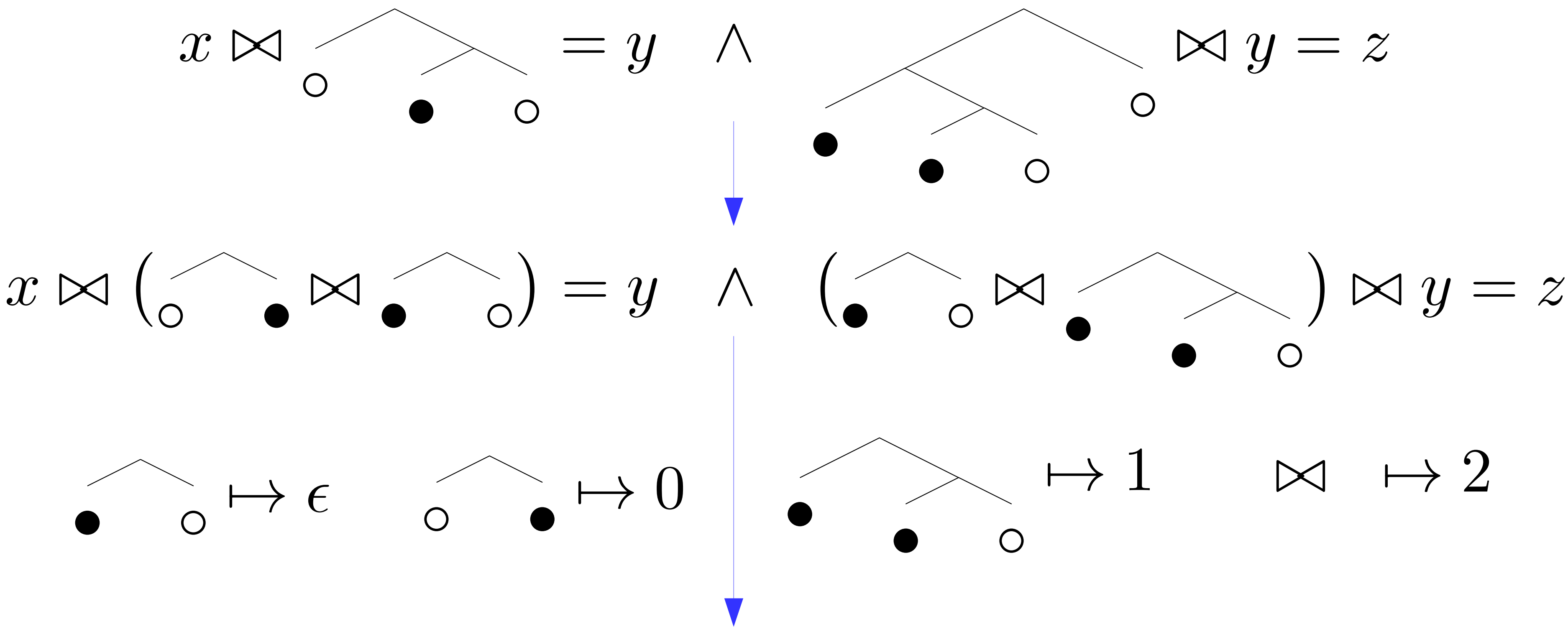
Example



Example

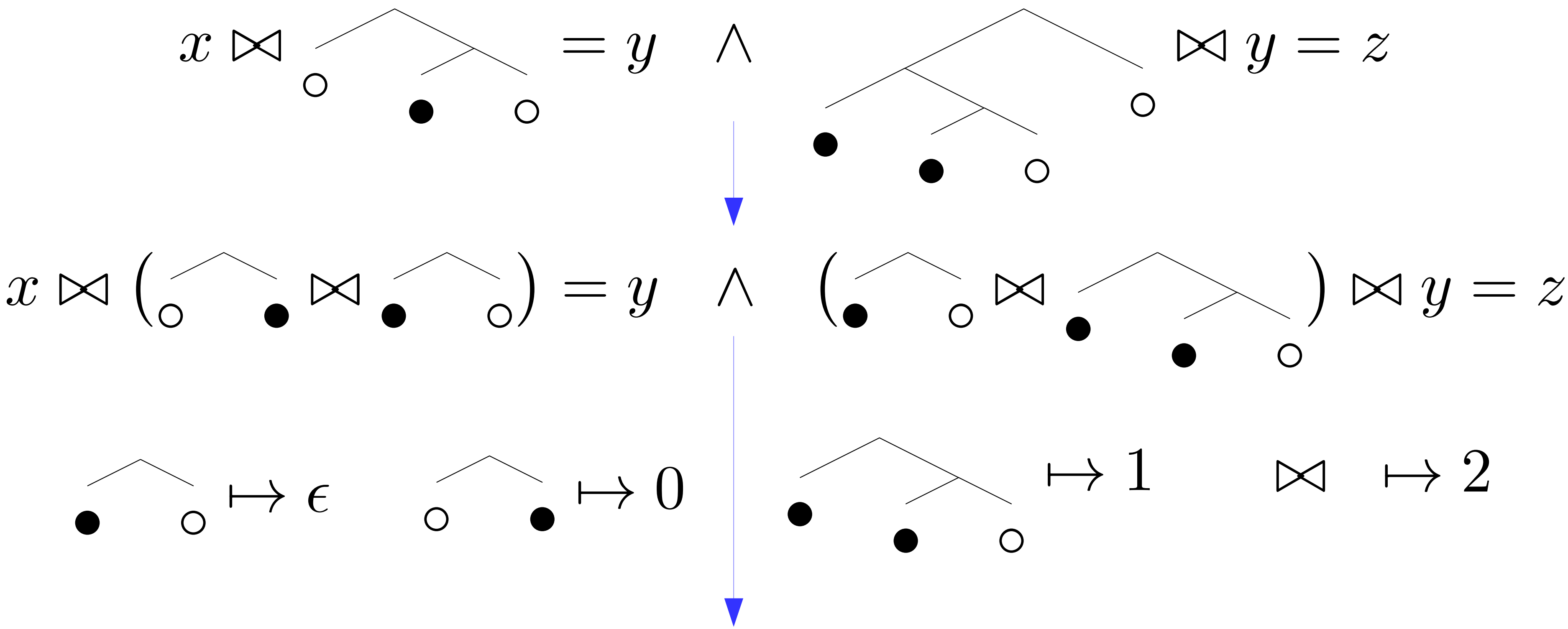


Example



$$x202 = y \quad \wedge \quad 212y = z$$

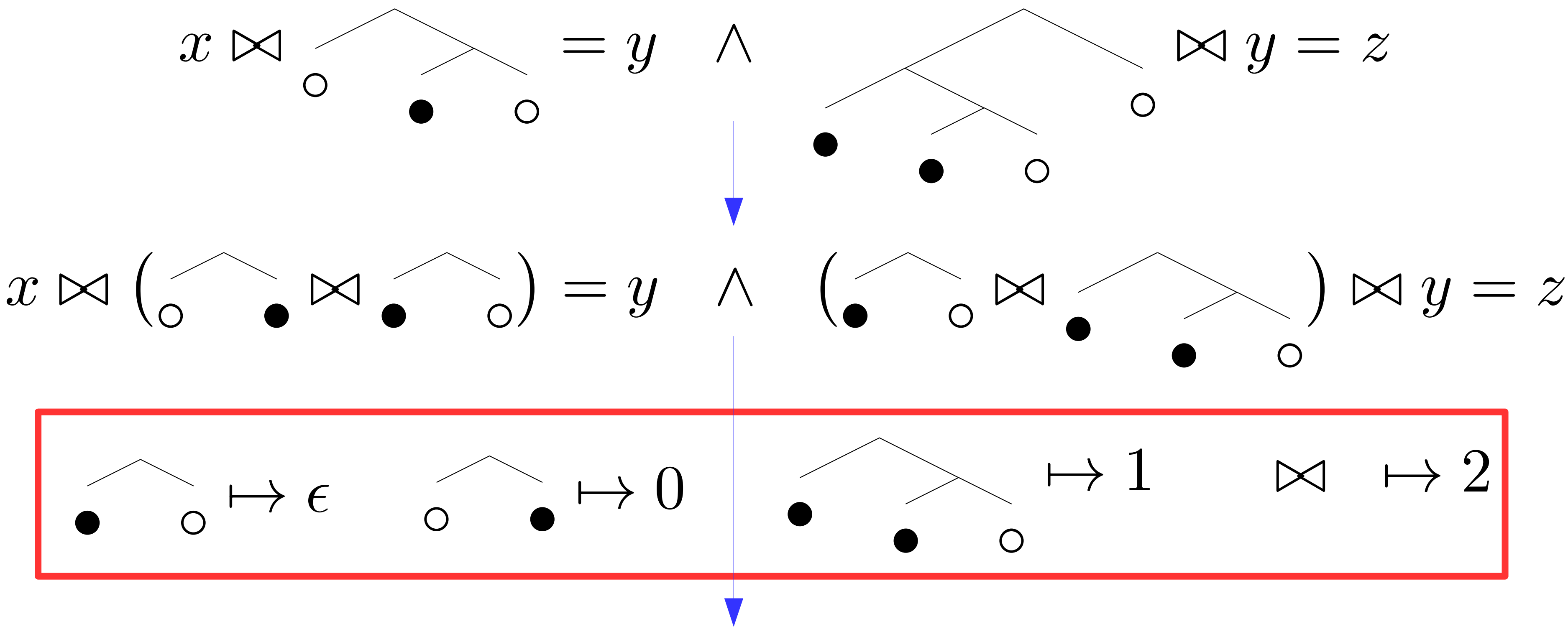
Example



$$x202 = y \quad \wedge \quad 212y = z$$

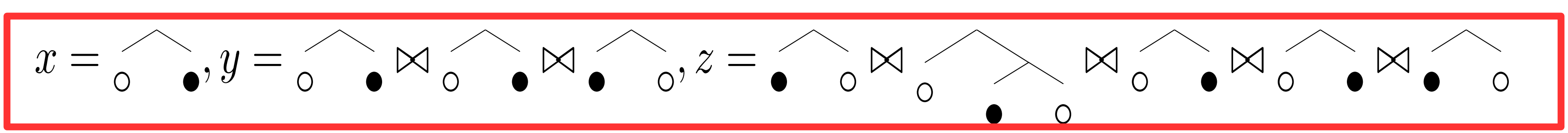
Solution: $x = 0, y = 0202, z = 2120202$

Example



$$x202 = y \quad \wedge \quad 212y = z$$

Solution: $x = 0, y = 0202, z = 2120202$



Agenda

1. Introduction
2. Complexity for Boolean structure
3. Complexity for multiplication structure
4. Non-elementary bound for combined structure
5. Conclusion

Combined tree share structure

- Both Boolean structure $\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot} \rangle$ and multiplication structure $\langle \mathbb{T}, \bowtie_{\tau}, {}_{\tau}\!\bowtie \rangle$ have elementary complexity.
- In practice, we may need both, e.g. recursive programs (Le et al., ESOP 2018).
- What is the decidability and complexity of the combined structure $\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot}, \bowtie_{\tau}, {}_{\tau}\!\bowtie \rangle$?

Main result

(Le et al., 2016) The FO theory of combined structure $\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot}, \bowtie_\tau \rangle$ is decidable.

Main result

(Le et al., 2016) The FO theory of combined structure $\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot}, \bowtie_\tau \rangle$ is decidable.

Theorem 3. The FO theory of $\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot}, \bowtie_\tau \rangle$ cannot be bounded by any tower exponent function $2^n, 2^{2^n}, 2^{2^{2^n}} \ldots$

Main result

In other words, its FO theory is non-elementary!

Theorem 3. The FO theory of $\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot}, \bowtie_\tau \rangle$ cannot be bounded by any tower exponent function $2^n, 2^{2^n}, 2^{2^{2^n}} \ldots$

Main result

Solvers need to choose between completeness and performance

Theorem 3. The FO theory of $\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot}, \bowtie_\tau \rangle$ cannot be bounded by any tower exponent function $2^n, 2^{2^n}, 2^{2^{2^n}} \ldots$

Agenda

1. Introduction
2. Complexity for Boolean structure
3. Complexity for multiplication structure
4. Non-elementary bound for combined structure
5. Conclusion

Conclusion and future work

- We proved two tight complexity bounds for the FO theory of the Boolean tree share structure $\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot} \rangle$ and of the multiplication structure $\langle \mathbb{T}, \bowtie_{\mathcal{T}}, {}_{\mathcal{T}}\bowtie \rangle$.
- We showed that the FO theory of combined structure $\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot}, \bowtie_{\mathcal{T}} \rangle$, although decidable, has non-elementary complexity.
- Future work:
  - Investigate the full combined structure $\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot}, \bowtie_{\mathcal{T}}, {}_{\mathcal{T}}\bowtie \rangle$.
  - Solver for $\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot}, \bowtie_{\mathcal{T}} \rangle$ using MONA.

Thank you for your attention!

Main result

Solvers need to choose between completeness and performance

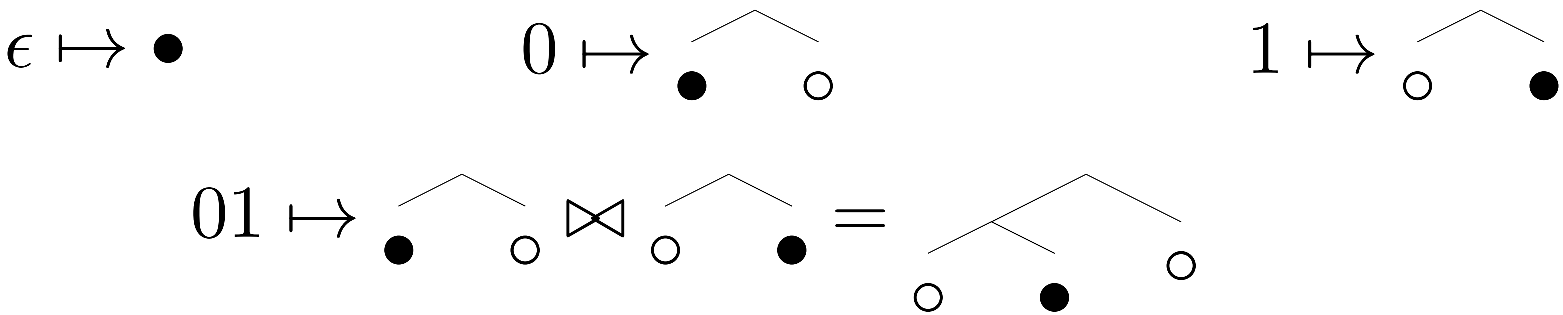
Theorem 3. The FO theory of $\langle \mathbb{T}, \sqcup, \sqcap, \bar{\cdot}, \bowtie_\tau \rangle$ cannot be bounded by any tower exponent function $2^n, 2^{2^n}, 2^{2^{2^n}} \ldots$

Key idea

- Reduce from binary string structure with successors and prefix relation $\langle \{0,1\}^*, S_0, S_1, \leq \rangle$
  - $S_0(x) = x0$, e.g. $S_0(101) = 1010$
  - $S_1(x) = x1$, e.g. $S_1(101) = 1011$
  - $x \leq y$ iff x is prefix of y, e.g. $10 \leq 1001$
  - FO theory is nonelementary (Stockmeyer, PhD thesis 1974)

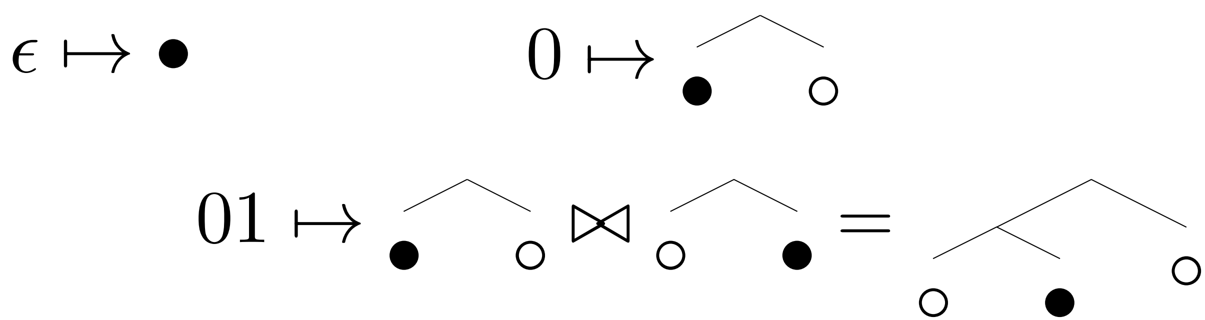
Key idea

- Map each binary string to a unary tree share, i.e. tree share with exactly one black leaf, e.g.

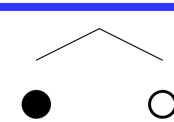
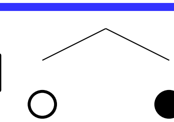


Key idea

- Map each binary string to a unary tree share, i.e. tree share with exactly one black leaf, e.g.



- The predicate isUnary can be expressed using Boolean and multiplication operators:

$$\text{isUnary}(\tau) \stackrel{\text{def}}{=} \tau \neq \circ \wedge \forall \tau'.\ \tau' \bowtie$$  $$\sqsubset \tau \Leftrightarrow \tau' \bowtie$$  $$\sqsubset \tau$$

where

$$\tau_1 \sqsubset \tau_2 \stackrel{\text{def}}{=} \tau_1 \sqcup \tau_2 = \tau_2 \wedge \tau_1 \neq \tau_2$$

Key idea

- One can transform a FO formula from string structure into tree structure, which justifies the lower bound.

Key idea

- One can transform a FO formula from string structure into tree structure, which justifies the lower bound.

- Example:

$$\forall x \exists y.\ x \leq y \vee S_1(x) = y$$

is equivalent to

$$\forall x \in \mathsf{isUnary}, \exists y \in \mathsf{isUnary}.\ x \sqcup y = x \vee x \bowtie \widehat{\circ \quad \bullet} = y$$