
DECOMPOSITION THEOREMS IN LOGIC

Łukasz Kaiser

CNRS & LIAFA
Paris

CELEBRATING THE 100TH VOLUME OF
FUNDAMENTA INFORMATIÆ
Warsaw, 2010

Overview

Finite Decompositions

WMSO on Inductive Structures (with T. Ganzow)

Infinite Decompositions

Cardinality Quantifiers (with V. Bárány and A. Rabinovich)

What is a Logical Decomposition?

$$\mathfrak{A}_1 \oplus \dots \oplus \mathfrak{A}_n \models \varphi \iff \bigvee_{i < m} (\mathfrak{A}_1 \models \varphi_1^i \wedge \dots \wedge \mathfrak{A}_n \models \varphi_n^i)$$

- \mathfrak{A}_k are **relational structures**
- φ is a **formula** of **first-order** (FO) or **monadic second-order** (MSO) logic
- \oplus is an operation on structures such as \times or $\dot{\cup}$

What is a Logical Decomposition?

$$\mathfrak{A}_1 \oplus \dots \oplus \mathfrak{A}_n \models \varphi \iff \bigvee_{i < m} (\mathfrak{A}_1 \models \varphi_1^i \wedge \dots \wedge \mathfrak{A}_n \models \varphi_n^i)$$

- \mathfrak{A}_k are **relational structures**
- φ is a **formula** of **first-order** (FO) or **monadic second-order** (MSO) logic
- \oplus is an operation on structures such as \times or $\dot{\cup}$

Second-order Logic and Fragments

- **Full second-order logic (SO):** $\exists R (Rxy \rightarrow Exy \wedge \forall x \exists !y Rxy)$
- **Monadic second-order logic (MSO):**
 $\forall X (x \in X \wedge (\forall z, v (z \in X \wedge Ezv \rightarrow v \in X)) \rightarrow y \in X)$
- **Weak monadic second-order logic (WMSO):** only finite subsets

Theories

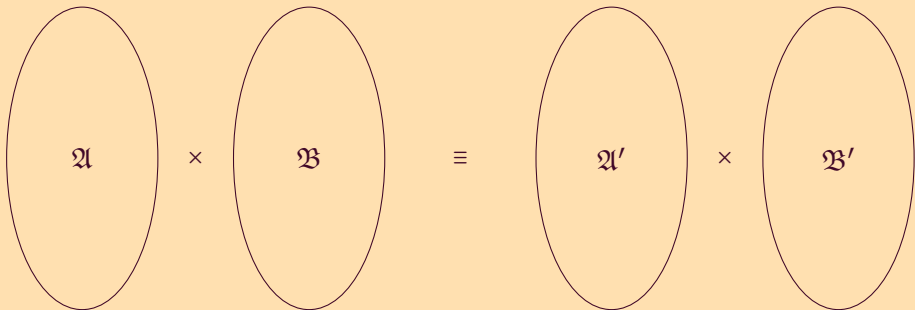
- $\text{th}^n(\mathfrak{A}) = \{\varphi \in \text{FO} \mid \mathfrak{A} \models \varphi, \text{qr}(\varphi) \leq n\}$
- $\text{Th}^n(\mathfrak{A}) = \{\varphi \in \text{MSO} \mid \mathfrak{A} \models \varphi, \text{qr}(\varphi) \leq n\}$

Basic Decomposition Theorems: FO

First-order logic can be decomposed on products

$$\text{th}^n(\mathfrak{A}) = \text{th}^n(\mathfrak{A}') \text{ and } \text{th}^n(\mathfrak{B}) = \text{th}^n(\mathfrak{B}') \Rightarrow \text{th}^n(\mathfrak{A} \times \mathfrak{B}) = \text{th}^n(\mathfrak{A}' \times \mathfrak{B}')$$

Proof (Ehrenfeucht-Fraïssé Games)

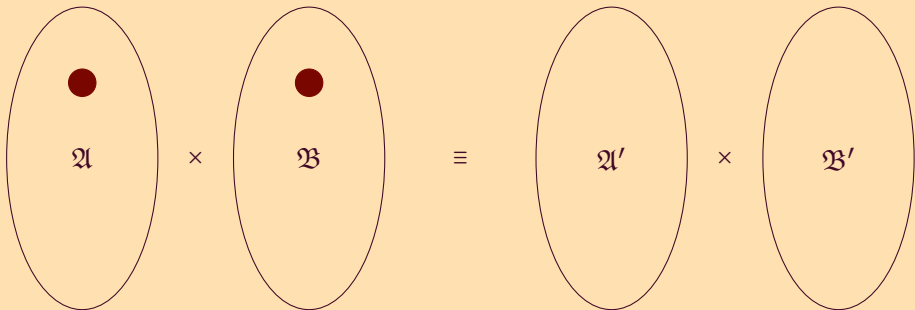


Basic Decomposition Theorems: FO

First-order logic can be decomposed on products

$$\text{th}^n(\mathfrak{A}) = \text{th}^n(\mathfrak{A}') \text{ and } \text{th}^n(\mathfrak{B}) = \text{th}^n(\mathfrak{B}') \Rightarrow \text{th}^n(\mathfrak{A} \times \mathfrak{B}) = \text{th}^n(\mathfrak{A}' \times \mathfrak{B}')$$

Proof (Ehrenfeucht-Fraïssé Games)



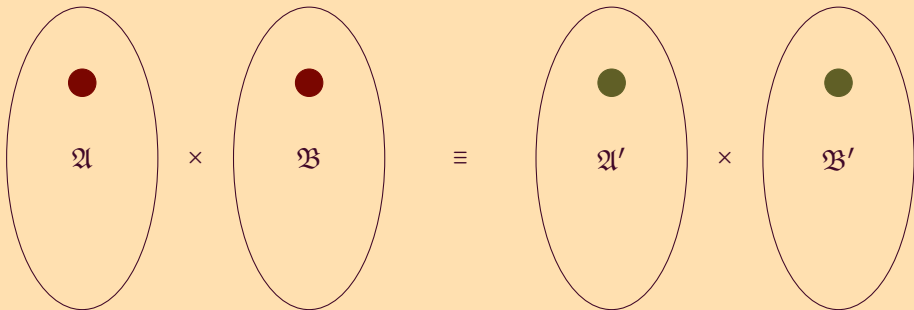
Spoiler: (x, y)

Basic Decomposition Theorems: FO

First-order logic can be decomposed on products

$$\text{th}^n(\mathfrak{A}) = \text{th}^n(\mathfrak{A}') \text{ and } \text{th}^n(\mathfrak{B}) = \text{th}^n(\mathfrak{B}') \Rightarrow \text{th}^n(\mathfrak{A} \times \mathfrak{B}) = \text{th}^n(\mathfrak{A}' \times \mathfrak{B}')$$

Proof (Ehrenfeucht-Fraïssé Games)



Spoiler: (x, y)

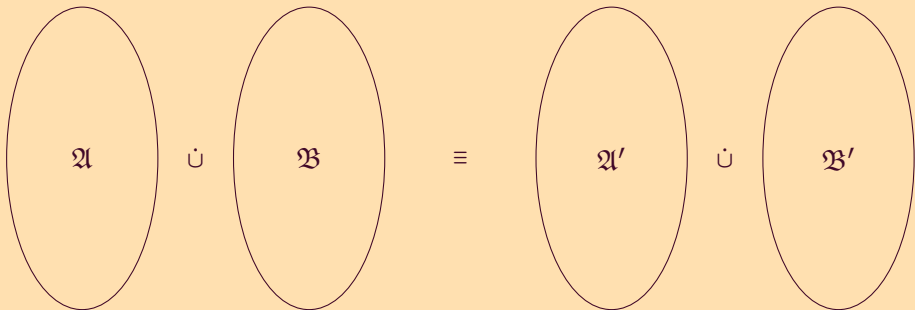
Duplicator: $(\text{dup}_{\mathfrak{A}, \mathfrak{A}'}(x), \text{dup}_{\mathfrak{B}, \mathfrak{B}'}(y))$

Basic Decomposition Theorems: MSO

Monadic second-order logic can be decomposed on disjoint unions

$$\text{Th}^n(\mathfrak{A}) = \text{Th}^n(\mathfrak{A}') \text{ and } \text{Th}^n(\mathfrak{B}) = \text{Th}^n(\mathfrak{B}') \Rightarrow \text{Th}^n(\mathfrak{A} \dot{\cup} \mathfrak{B}) = \text{Th}^n(\mathfrak{A}' \dot{\cup} \mathfrak{B}')$$

Proof (Ehrenfeucht-Fraïssé Games)

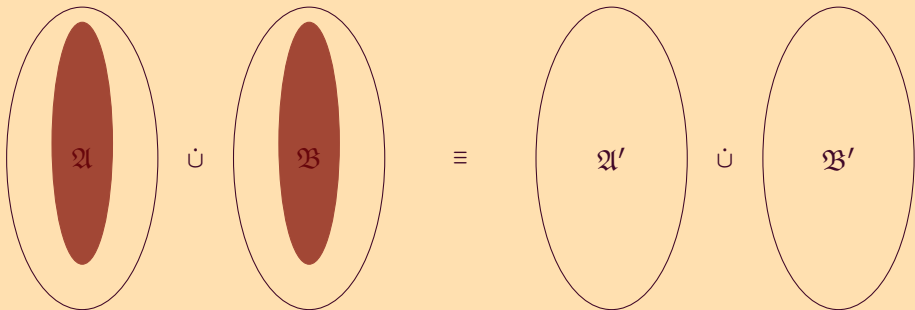


Basic Decomposition Theorems: MSO

Monadic second-order logic can be decomposed on disjoint unions

$$\text{Th}^n(\mathfrak{A}) = \text{Th}^n(\mathfrak{A}') \text{ and } \text{Th}^n(\mathfrak{B}) = \text{Th}^n(\mathfrak{B}') \Rightarrow \text{Th}^n(\mathfrak{A} \dot{\cup} \mathfrak{B}) = \text{Th}^n(\mathfrak{A}' \dot{\cup} \mathfrak{B}')$$

Proof (Ehrenfeucht-Fraïssé Games)



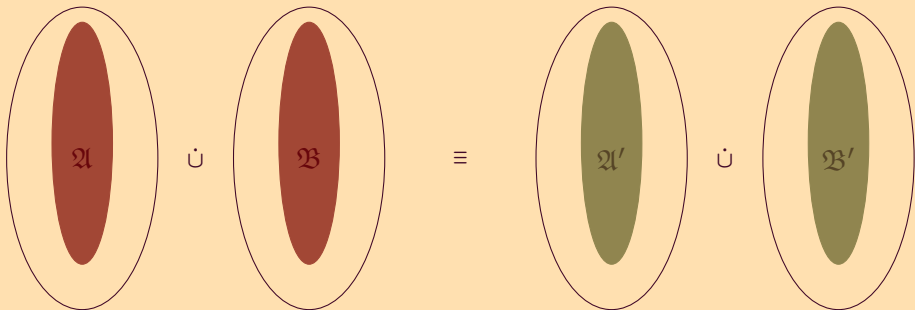
Spoiler: $X_A \dot{\cup} X_B$

Basic Decomposition Theorems: MSO

Monadic second-order logic can be decomposed on disjoint unions

$$\text{Th}^n(\mathfrak{A}) = \text{Th}^n(\mathfrak{A}') \text{ and } \text{Th}^n(\mathfrak{B}) = \text{Th}^n(\mathfrak{B}') \Rightarrow \text{Th}^n(\mathfrak{A} \dot{\cup} \mathfrak{B}) = \text{Th}^n(\mathfrak{A}' \dot{\cup} \mathfrak{B}')$$

Proof (Ehrenfeucht-Fraïssé Games)



Spoiler: $X_A \dot{\cup} X_B$

Duplicator: $\text{dup}_{\mathfrak{A}, \mathfrak{A}'}(X_A) \dot{\cup} \text{dup}_{\mathfrak{B}, \mathfrak{B}'}(X_B)$

No Decomposition: MSO on products, SO

Monadic second-order logic cannot be decomposed on products

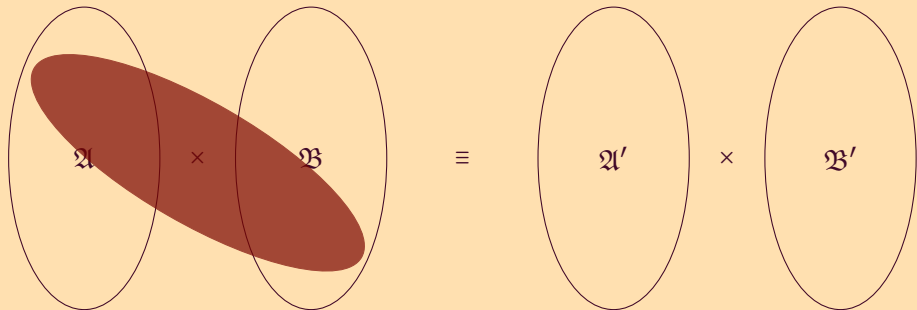
Full second-order logic cannot be decomposed on disjoint unions

No Decomposition: MSO on products, SO

Monadic second-order logic cannot be decomposed on products

Full second-order logic cannot be decomposed on disjoint unions

Intuition



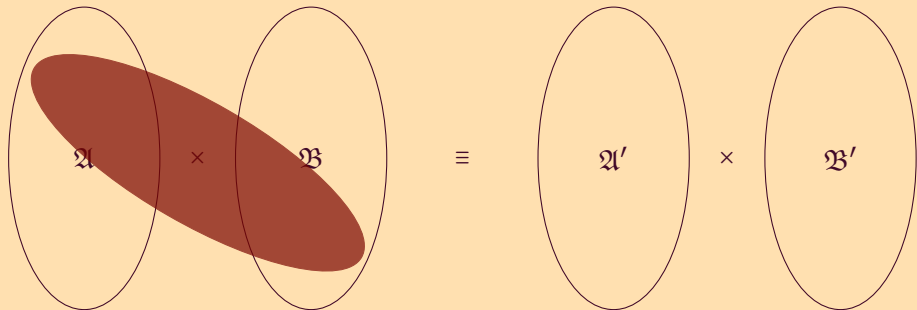
Spoiler: $X \neq X_A \times X_B$

No Decomposition: MSO on products, SO

Monadic second-order logic cannot be decomposed on products

Full second-order logic cannot be decomposed on disjoint unions

Intuition



Spoiler: $X \neq X_A \times X_B$

Duplicator: ???

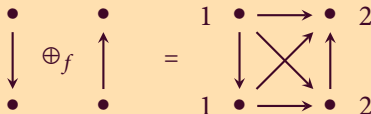
Computing MSO Decomposition

Given: φ with $FV(\varphi) \in \{X_1, \dots, X_r\}$ and

$$\mathfrak{A} = \mathfrak{A}_1 \oplus \dots \oplus \mathfrak{A}_n \text{ with } f_{R_1}, \dots, f_{R_l}$$

Compute: $\forall_{i < m} (\mathfrak{A}_1 \models \varphi_1^i \wedge \dots \wedge \mathfrak{A}_n \models \varphi_n^i) \iff \mathfrak{A} \models \varphi.$

Generalised Disjoint Union: let $f_E(i, j) = \top \iff (i, j) = (1, 2)$ (all-to-all)



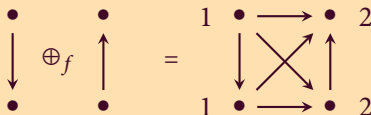
Computing MSO Decomposition

Given: φ with $FV(\varphi) \in \{X_1, \dots, X_r\}$ and

$$\mathfrak{A} = \mathfrak{A}_1 \oplus \dots \oplus \mathfrak{A}_n \text{ with } f_{R_1}, \dots, f_{R_l}$$

Compute: $\forall_{i < m} (\mathfrak{A}_1 \models \varphi_1^i \wedge \dots \wedge \mathfrak{A}_n \models \varphi_n^i) \iff \mathfrak{A} \models \varphi.$

Generalised Disjoint Union: let $f_E(i, j) = \top \iff (i, j) = (1, 2)$ (all-to-all)



- (1) “split” variables in φ into several ones implicitly ranging over the components of the structure
- (2) replace trivially true or false atoms and simplify (wrt. the restriction to components and the functions defining relations across components)
- (3) compute the TNF (type normal form) of the resulting formula and transform it into DNF

Step 1 — Splitting

Given: φ with $FV(\varphi) \in \{X_1, \dots, X_r\}$ and

$$\mathfrak{A} = \mathfrak{A}_1 \oplus \dots \oplus \mathfrak{A}_n \text{ with } f_{R_1}, \dots, f_{R_l}$$

Compute: $\bigvee_{i < m} (\mathfrak{A}_1 \models \varphi_1^i \wedge \dots \wedge \mathfrak{A}_n \models \varphi_n^i) \iff \mathfrak{A} \models \varphi.$

$\text{split}(\varphi)$ is computed by transforming

$$\exists X \psi \mapsto \exists X^1 \dots X^k \psi[x \in X / \bigvee_{i=1}^k x \in X^i]$$

$$\exists x \psi \mapsto \bigvee_{i=1}^k \exists x^i \psi[x/x^i]$$

$$\forall X \psi \mapsto \forall X^1 \dots X^k \psi[x \in X / \bigvee_{i=1}^k x \in X^i]$$

$$\forall x \psi \mapsto \bigwedge_{i=1}^k \forall x^i \psi[x/x^i]$$

Semantics: x^i, X^i are implicitly restricted to the domain of the i -th component of the structure.

Step 1 — Splitting on Binary Tree = Tree \oplus Root \oplus Tree

$$\text{split}\left(\exists X \forall x (x \in X \rightarrow S_L x \wedge \forall y (y < x \rightarrow (Ry \vee S_R y)))\right)$$

Step 1 — Splitting on Binary Tree = Tree \oplus Root \oplus Tree

$$\text{split}\left(\exists X \forall x(x \in X \rightarrow S_L x \wedge \forall y(y < x \rightarrow (Ry \vee S_R y)))\right)$$

$$\begin{aligned} &= \exists X^1 \exists X^2 \exists X^3 \forall x^1 \left(\bigvee_{i=1}^3 x^i \in X^i \rightarrow S_L x^1 \wedge \left(\begin{array}{l} \forall y^1(y^1 < x^1 \rightarrow (Ry^1 \vee S_R y^1)) \\ \wedge \forall y^2(y^2 < x^1 \rightarrow (Ry^2 \vee S_R y^2)) \\ \wedge \forall y^3(y^3 < x^1 \rightarrow (Ry^3 \vee S_R y^3)) \end{array} \right) \right) \\ &\quad \wedge \forall x^2 \left(\bigvee_{i=1}^3 x^i \in X^i \rightarrow S_L x^2 \wedge \left(\begin{array}{l} \forall y^1(y^1 < x^2 \rightarrow (Ry^1 \vee S_R y^1)) \\ \wedge \forall y^2(y^2 < x^2 \rightarrow (Ry^2 \vee S_R y^2)) \\ \wedge \forall y^3(y^3 < x^2 \rightarrow (Ry^3 \vee S_R y^3)) \end{array} \right) \right) \\ &\quad \wedge \forall x^3 \left(\bigvee_{i=1}^3 x^i \in X^i \rightarrow S_L x^3 \wedge \left(\begin{array}{l} \forall y^1(y^1 < x^3 \rightarrow (Ry^1 \vee S_R y^1)) \\ \wedge \forall y^2(y^2 < x^3 \rightarrow (Ry^2 \vee S_R y^2)) \\ \wedge \forall y^3(y^3 < x^3 \rightarrow (Ry^3 \vee S_R y^3)) \end{array} \right) \right) \end{aligned}$$

Step 2 — Reducing the Split Formula

$$\text{split}\left(\exists X \forall x(x \in X \rightarrow S_L x \wedge \forall y(y < x \rightarrow (Ry \vee S_R y)))\right)$$

$$\begin{aligned} &= \exists X^1 \exists X^2 \exists X^3 \forall x^1 \left(\bigvee_{i=1}^3 x^i \in X^i \rightarrow S_L x^1 \wedge \left(\begin{array}{l} \forall y^1(y^1 < x^1 \rightarrow (Ry^1 \vee S_R y^1)) \\ \wedge \forall y^2(y^2 < x^1 \rightarrow (Ry^2 \vee S_R y^2)) \\ \wedge \forall y^3(y^3 < x^1 \rightarrow (Ry^3 \vee S_R y^3)) \end{array} \right) \right) \\ &\quad \wedge \forall x^2 \left(\bigvee_{i=1}^3 x^i \in X^i \rightarrow S_L x^2 \wedge \left(\begin{array}{l} \forall y^1(y^1 < x^2 \rightarrow (Ry^1 \vee S_R y^1)) \\ \wedge \forall y^2(y^2 < x^2 \rightarrow (Ry^2 \vee S_R y^2)) \\ \wedge \forall y^3(y^3 < x^2 \rightarrow (Ry^3 \vee S_R y^3)) \end{array} \right) \right) \\ &\quad \wedge \forall x^3 \left(\bigvee_{i=1}^3 x^i \in X^i \rightarrow S_L x^3 \wedge \left(\begin{array}{l} \forall y^1(y^1 < x^3 \rightarrow (Ry^1 \vee S_R y^1)) \\ \wedge \forall y^2(y^2 < x^3 \rightarrow (Ry^2 \vee S_R y^2)) \\ \wedge \forall y^3(y^3 < x^3 \rightarrow (Ry^3 \vee S_R y^3)) \end{array} \right) \right) \end{aligned}$$

$x^j \in X^i \equiv \perp$
if $i \neq j$

Step 2 — Reducing the Split Formula

$$\text{split}\left(\exists X \forall x (x \in X \rightarrow S_L x \wedge \forall y (y < x \rightarrow (Ry \vee S_R y)))\right)$$

$$\begin{aligned} &= \exists X^1 \exists X^2 \exists X^3 \forall x^1 \left(x^1 \in X^1 \rightarrow S_L x^1 \wedge \left(\begin{array}{l} \forall y^1 (y^1 < x^1 \rightarrow (Ry^1 \vee S_R y^1)) \\ \wedge \forall y^2 (y^2 < x^1 \rightarrow (Ry^2 \vee S_R y^2)) \\ \wedge \forall y^3 (y^3 < x^1 \rightarrow (Ry^3 \vee S_R y^3)) \end{array} \right) \right) \\ &\quad \wedge \forall x^2 \left(x^2 \in X^2 \rightarrow S_L x^2 \wedge \left(\begin{array}{l} \forall y^1 (y^1 < x^2 \rightarrow (Ry^1 \vee S_R y^1)) \\ \wedge \forall y^2 (y^2 < x^2 \rightarrow (Ry^2 \vee S_R y^2)) \\ \wedge \forall y^3 (y^3 < x^2 \rightarrow (Ry^3 \vee S_R y^3)) \end{array} \right) \right) \\ &\quad \wedge \forall x^3 \left(x^3 \in X^3 \rightarrow S_L x^3 \wedge \left(\begin{array}{l} \forall y^1 (y^1 < x^3 \rightarrow (Ry^1 \vee S_R y^1)) \\ \wedge \forall y^2 (y^2 < x^3 \rightarrow (Ry^2 \vee S_R y^2)) \\ \wedge \forall y^3 (y^3 < x^3 \rightarrow (Ry^3 \vee S_R y^3)) \end{array} \right) \right) \end{aligned}$$

Step 2 — Reducing the Split Formula

$$\text{split}\left(\exists X \forall x(x \in X \rightarrow S_L x \wedge \forall y(y < x \rightarrow (Ry \vee S_R y)))\right)$$

$$\begin{aligned} &= \exists X^1 \exists X^2 \exists X^3 \forall x^1 \left(x^1 \in X^1 \rightarrow S_L x^1 \wedge \left(\begin{array}{l} \forall y^1(y^1 < x^1 \rightarrow (Ry^1 \vee S_R y^1)) \\ \wedge \forall y^2(y^2 < x^1 \rightarrow (Ry^2 \vee S_R y^2)) \\ \wedge \forall y^3(y^3 < x^1 \rightarrow (Ry^3 \vee S_R y^3)) \end{array} \right) \right) \\ &\quad \wedge \forall x^2 \left(x^2 \in X^2 \rightarrow S_L x^2 \wedge \left(\begin{array}{l} \forall y^1(y^1 < x^2 \rightarrow (Ry^1 \vee S_R y^1)) \\ \wedge \forall y^2(y^2 < x^2 \rightarrow (Ry^2 \vee S_R y^2)) \\ \wedge \forall y^3(y^3 < x^2 \rightarrow (Ry^3 \vee S_R y^3)) \end{array} \right) \right) \\ &\quad \wedge \forall x^3 \left(x^3 \in X^3 \rightarrow S_L x^3 \wedge \left(\begin{array}{l} \forall y^1(y^1 < x^3 \rightarrow (Ry^1 \vee S_R y^1)) \\ \wedge \forall y^2(y^2 < x^3 \rightarrow (Ry^2 \vee S_R y^2)) \\ \wedge \forall y^3(y^3 < x^3 \rightarrow (Ry^3 \vee S_R y^3)) \end{array} \right) \right) \end{aligned}$$

for $i \neq j$: $y^i < x^j \equiv f_<(i, j)$

Step 2 — Reducing the Split Formula

$$\text{split}\left(\exists X \forall x (x \in X \rightarrow S_L x \wedge \forall y (y < x \rightarrow (Ry \vee S_R y)))\right)$$

$$\begin{aligned} &= \exists X^1 \exists X^2 \exists X^3 \quad \forall x^1 \left(x^1 \in X^1 \rightarrow S_L x^1 \wedge \left(\begin{array}{l} \forall y^1 (y^1 < x^1 \rightarrow (Ry^1 \vee S_R y^1)) \\ \wedge \forall y^2 (Ry^2 \vee S_R y^2) \end{array} \right) \right) \\ &\quad \wedge \forall x^2 \left(x^2 \in X^2 \rightarrow S_L x^2 \wedge \left(\forall y^2 (y^2 < x^2 \rightarrow (Ry^2 \vee S_R y^2)) \right) \right) \\ &\quad \wedge \forall x^3 \left(x^3 \in X^3 \rightarrow S_L x^3 \wedge \left(\begin{array}{l} \forall y^2 (Ry^2 \vee S_R y^2) \\ \wedge \forall y^3 (y^3 < x^3 \rightarrow (Ry^3 \vee S_R y^3)) \end{array} \right) \right) \end{aligned}$$

Step 3 — Type Normal Form

Positive Boolean combination of formulas of the form

$$\tau = R_i(\bar{x}) \mid \neg R_i(\bar{x}) \mid x \in X \mid x \notin X \mid \\ \exists x \mathcal{B}^+(\tau) \mid \exists X \mathcal{B}^+(\tau) \mid \forall x \mathcal{B}^+(\tau) \mid \forall X \mathcal{B}^+(\tau)$$

such that for **each** τ_i in the Boolean combination $\mathcal{B}^+(\tau_i)$, $x \in FV(\tau_i)$.

Step 3 — Type Normal Form

Positive Boolean combination of formulas of the form

$$\tau = R_i(\bar{x}) \mid \neg R_i(\bar{x}) \mid x \in X \mid x \notin X \mid \\ \exists x \mathcal{B}^+(\tau) \mid \exists X \mathcal{B}^+(\tau) \mid \forall x \mathcal{B}^+(\tau) \mid \forall X \mathcal{B}^+(\tau)$$

such that for **each** τ_i in the Boolean combination $\mathcal{B}^+(\tau_i)$, $x \in FV(\tau_i)$.

Example $\varphi = \exists x(Px \wedge (Qy \vee Rx))$

Step 3 — Type Normal Form

Positive Boolean combination of formulas of the form

$$\begin{aligned}\tau = & R_i(\bar{x}) \mid \neg R_i(\bar{x}) \mid x \in X \mid x \notin X \mid \\ & \exists x \mathcal{B}^+(\tau) \mid \exists X \mathcal{B}^+(\tau) \mid \forall x \mathcal{B}^+(\tau) \mid \forall X \mathcal{B}^+(\tau)\end{aligned}$$

such that for **each** τ_i in the Boolean combination $\mathcal{B}^+(\tau_i)$, $x \in FV(\tau_i)$.

Example

$$\begin{aligned}\varphi &= \exists x(Px \wedge (Qy \vee Rx)) \\ &\equiv \exists x((Px \wedge Qy) \vee (Px \wedge Rx))\end{aligned}$$

Step 3 — Type Normal Form

Positive Boolean combination of formulas of the form

$$\begin{aligned}\tau = & R_i(\bar{x}) \mid \neg R_i(\bar{x}) \mid x \in X \mid x \notin X \mid \\ & \exists x \mathcal{B}^+(\tau) \mid \exists X \mathcal{B}^+(\tau) \mid \forall x \mathcal{B}^+(\tau) \mid \forall X \mathcal{B}^+(\tau)\end{aligned}$$

such that for **each** τ_i in the Boolean combination $\mathcal{B}^+(\tau_i)$, $x \in FV(\tau_i)$.

Example

$$\begin{aligned}\varphi &= \exists x(Px \wedge (Qy \vee Rx)) \\ &\equiv \exists x((Px \wedge Qy) \vee (Px \wedge Rx)) \\ &\equiv \exists x(Px \wedge Qy) \vee \exists x(Px \wedge Rx)\end{aligned}$$

Step 3 — Type Normal Form

Positive Boolean combination of formulas of the form

$$\begin{aligned}\tau = & R_i(\bar{x}) \mid \neg R_i(\bar{x}) \mid x \in X \mid x \notin X \mid \\ & \exists x \mathcal{B}^+(\tau) \mid \exists X \mathcal{B}^+(\tau) \mid \forall x \mathcal{B}^+(\tau) \mid \forall X \mathcal{B}^+(\tau)\end{aligned}$$

such that for **each** τ_i in the Boolean combination $\mathcal{B}^+(\tau_i)$, $x \in FV(\tau_i)$.

Example

$$\begin{aligned}\varphi &= \exists x(Px \wedge (Qy \vee Rx)) \\ &\equiv \exists x((Px \wedge Qy) \vee (Px \wedge Rx)) \\ &\equiv \exists x(Px \wedge Qy) \vee \exists x(Px \wedge Rx) \\ \text{TNF}(\varphi) &= (Qy \wedge \exists x Px) \vee \exists x(Px \wedge Rx)\end{aligned}$$

Step 3 — Type Normal Form

Lemma

For each φ there exists an equivalent ψ in TNF such that

- $\text{qr}(\psi) \leq \text{qr}(\varphi)$
- $\text{atoms}(\psi) \subseteq \text{atoms}(\varphi)$.

Step 3 — Type Normal Form

Lemma

For each φ there exists an equivalent ψ in TNF such that

- $\text{qr}(\psi) \leq \text{qr}(\varphi)$
- $\text{atoms}(\psi) \subseteq \text{atoms}(\varphi)$.

Lemma

Let φ be in TNF, V_1, \dots, V_n a partition of the variables such that variables appearing in the same atom are in the same V_i .

Then φ is a Boolean combination of formulas τ such that each τ contains only atoms with variables from one of the sets V_i .

↪ TNF-conversion “sorts” subformulas according to the components they speak about, and we obtain a DNF $\bigvee_i \bigwedge_j \psi_j^i$.

Overview

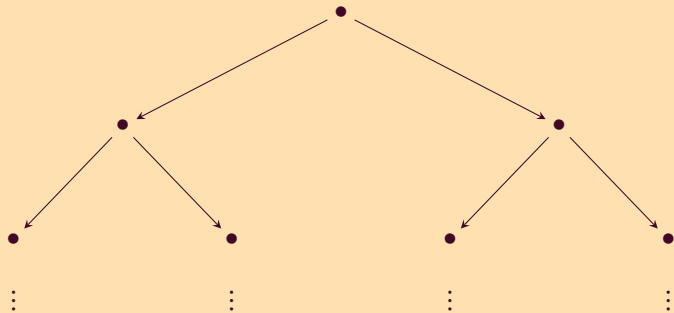
Finite Decompositions

WMSO on Inductive Structures (with T. Ganzow)

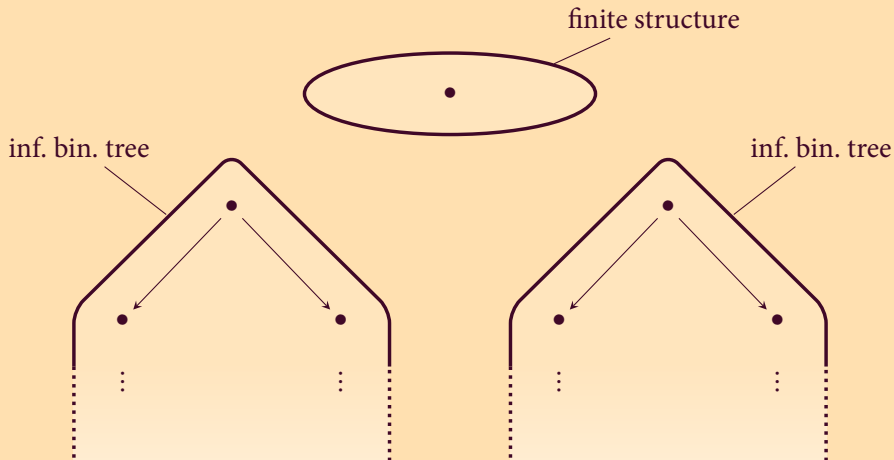
Infinite Decompositions

Cardinality Quantifiers (with V. Bárány and A. Rabinovich)

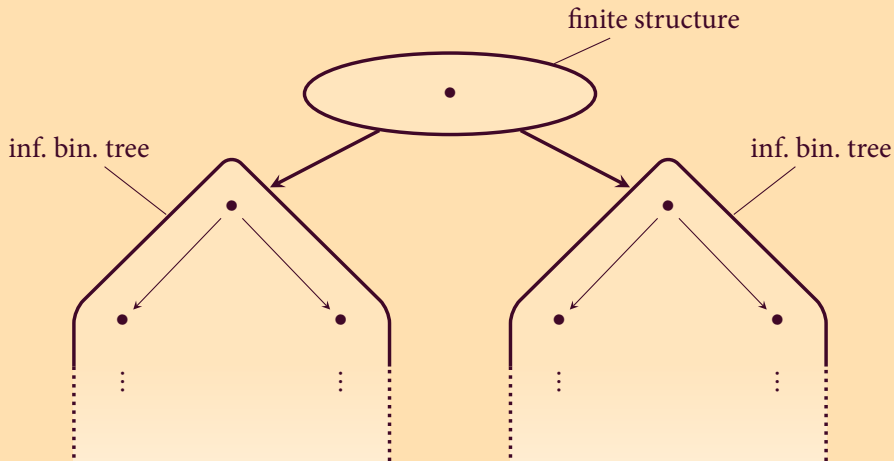
Inductive Structures — Intuition



Inductive Structures — Intuition



Inductive Structures — Intuition



Structure Equations

System of structure equations \mathcal{D} over $\tau = \{R_1, \dots, R_\ell\}$:

$$\mathcal{D} = \begin{cases} \Lambda^1 & = \mathfrak{A}_1^1 \oplus \mathfrak{A}_2^1 \oplus \dots \oplus \mathfrak{A}_{k_1}^1 & \text{with } f_1^1, \dots, f_\ell^1 \\ \vdots & & \vdots \\ \Lambda^n & = \mathfrak{A}_1^n \oplus \mathfrak{A}_2^n \oplus \dots \oplus \mathfrak{A}_{k_n}^n & \text{with } f_1^n, \dots, f_\ell^n \end{cases}$$

- each \mathfrak{A}_j^i is a **finite structure** or a **formal variable** in $\Lambda^1, \dots, \Lambda^n$
- each f_j^i is a function $\{1, \dots, k_i\}^{r_j} \rightarrow \{\perp, \top\}$

Solution of \mathcal{D} : $\mathcal{S}(\mathcal{D}) = (\mathfrak{A}_1, \dots, \mathfrak{A}_n)$

Example: Binary Tree with Predicates

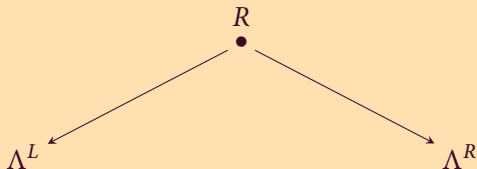
$$\mathcal{D} = \begin{cases} \Lambda^T = \Lambda^L \oplus R \bullet \oplus \Lambda^R \\ \Lambda^L = \Lambda^L \oplus S_L \bullet \oplus \Lambda^R \\ \Lambda^R = \Lambda^L \oplus S_R \bullet \oplus \Lambda^R \end{cases} \quad f_{<}(i, j) = \begin{cases} \top & \text{if } i = 2 \\ & j \in \{1, 3\} \\ \perp & \text{otherwise} \end{cases}$$

$\mathcal{S}_1(\mathcal{D}) :$

Example: Binary Tree with Predicates

$$\mathcal{D} = \begin{cases} \Lambda^T = \Lambda^L \oplus R \bullet \oplus \Lambda^R \\ \Lambda^L = \Lambda^L \oplus S_L \bullet \oplus \Lambda^R \\ \Lambda^R = \Lambda^L \oplus S_R \bullet \oplus \Lambda^R \end{cases} \quad f_{<}(i, j) = \begin{cases} \top & \text{if } i = 2 \\ & j \in \{1, 3\} \\ \perp & \text{otherwise} \end{cases}$$

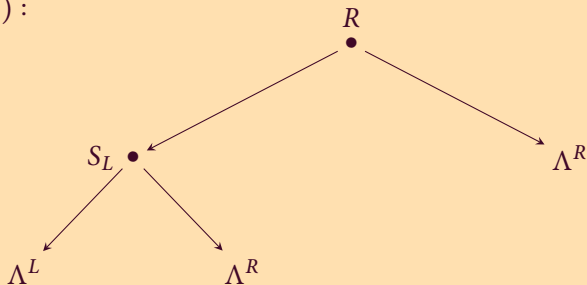
$\mathcal{S}_1(\mathcal{D}) :$



Example: Binary Tree with Predicates

$$\mathcal{D} = \begin{cases} \Lambda^T = \Lambda^L \oplus R \bullet \oplus \Lambda^R \\ \Lambda^L = \Lambda^L \oplus S_L \bullet \oplus \Lambda^R \\ \Lambda^R = \Lambda^L \oplus S_R \bullet \oplus \Lambda^R \end{cases} \quad f_{<}(i, j) = \begin{cases} \top & \text{if } i = 2 \\ & j \in \{1, 3\} \\ \perp & \text{otherwise} \end{cases}$$

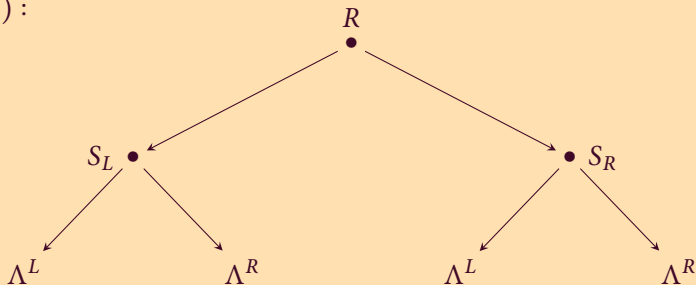
$\mathcal{S}_1(\mathcal{D}) :$



Example: Binary Tree with Predicates

$$\mathcal{D} = \begin{cases} \Lambda^T = \Lambda^L \oplus R \bullet \oplus \Lambda^R \\ \Lambda^L = \Lambda^L \oplus S_L \bullet \oplus \Lambda^R \\ \Lambda^R = \Lambda^L \oplus S_R \bullet \oplus \Lambda^R \end{cases} \quad f_{<}(i, j) = \begin{cases} \top & \text{if } i = 2 \\ & j \in \{1, 3\} \\ \perp & \text{otherwise} \end{cases}$$

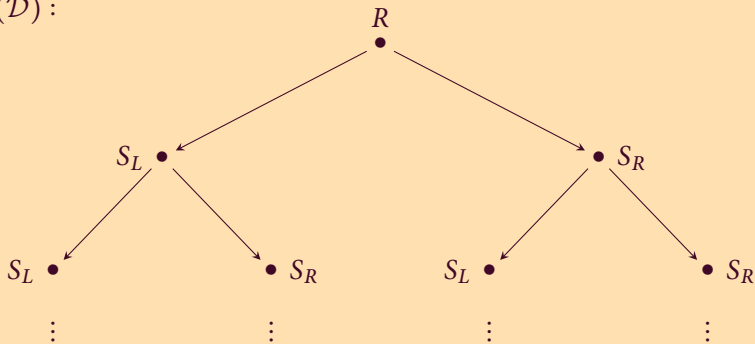
$\mathcal{S}_1(\mathcal{D}) :$



Example: Binary Tree with Predicates

$$\mathcal{D} = \begin{cases} \Lambda^T = \Lambda^L \oplus R \bullet \oplus \Lambda^R \\ \Lambda^L = \Lambda^L \oplus S_L \bullet \oplus \Lambda^R \\ \Lambda^R = \Lambda^L \oplus S_R \bullet \oplus \Lambda^R \end{cases} \quad f_{<}(i, j) = \begin{cases} \top & \text{if } i = 2 \\ & j \in \{1, 3\} \\ \perp & \text{otherwise} \end{cases}$$

$\mathcal{S}_1(\mathcal{D}) :$



\mathcal{D}_m -Decomposition

$$\mathcal{D} = \begin{Bmatrix} \vdots \\ \Lambda^m \\ \vdots \end{Bmatrix} = \mathfrak{A}_1^m \oplus \dots \oplus \mathfrak{A}_k^m$$

$$\mathcal{S}(\mathcal{D}) = (\mathfrak{A}_1, \dots, \mathfrak{A}_n)$$

A \mathcal{D}_m -**decomposition** of φ is a sequence

$$(\psi_1^1, \dots, \psi_k^1), \dots, (\psi_1^\ell, \dots, \psi_k^\ell)$$

such that

- $\mathfrak{A}_m \models \varphi \iff \exists i \forall j : \mathfrak{A}_m[j] \models \psi_j^i$
- $FV(\psi_j^i) \subseteq FV(\varphi)$
- $\text{qr}(\psi_j^i) \leq \text{qr}(\varphi)$

\mathcal{D}_m -Decomposition

$$\mathcal{D} = \begin{Bmatrix} \vdots \\ \Lambda^m \\ \vdots \end{Bmatrix} = \mathfrak{A}_1^m \oplus \dots \oplus \mathfrak{A}_k^m$$

$$\mathcal{S}(\mathcal{D}) = (\mathfrak{A}_1, \dots, \mathfrak{A}_n)$$

A \mathcal{D}_m -**decomposition** of φ is a sequence

$$(\psi_1^1, \dots, \psi_k^1), \dots, (\psi_1^\ell, \dots, \psi_k^\ell) \triangleq \bigvee_i \bigwedge_j \psi_j^i$$

such that

- $\mathfrak{A}_m \models \varphi \iff \exists i \forall j : \mathfrak{A}_m[j] \models \psi_j^i$
- $FV(\psi_j^i) \subseteq FV(\varphi)$
- $\text{qr}(\psi_j^i) \leq \text{qr}(\varphi)$

Model Checking Algorithm

Given: a WMSO sentence φ and \mathcal{D} .

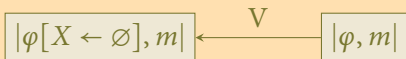
Problem: check whether $\mathcal{S}_m(\mathcal{D}) \models \varphi$.

- **Atomic sentences** \top and \perp : trivial
- **Boolean combinations:** evaluate subformulas
- $\exists X\varphi(X)$ or $\exists x\varphi(x)$:
determine the **winner** of the game $\mathcal{G}^\exists(\varphi, i)$ between
the **Verifier** and the **Falsifier**
- $\forall X\varphi(X)$ or $\forall x\varphi(x)$:
check $\neg\exists X\neg\varphi(X)$ by determining the **loser** of $\mathcal{G}^\exists(\neg\varphi, i)$

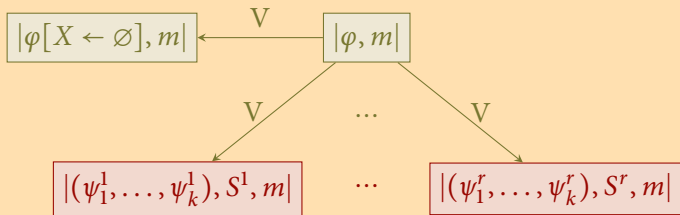
Game $\mathcal{G}^\exists(\varphi, m)$ for $\exists X\varphi(X)$

$|\varphi, m|$

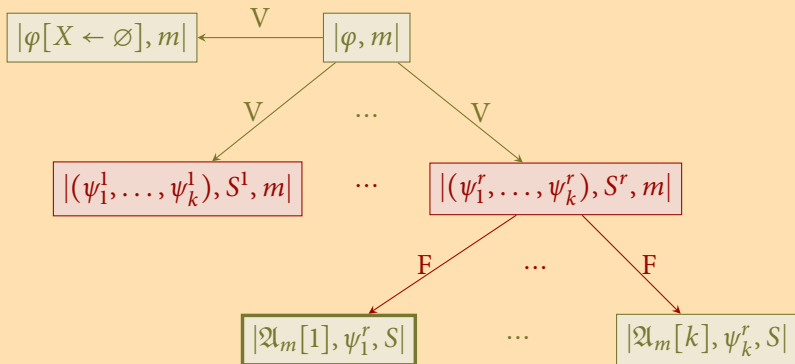
Game $\mathcal{G}^\exists(\varphi, m)$ for $\exists X\varphi(X)$



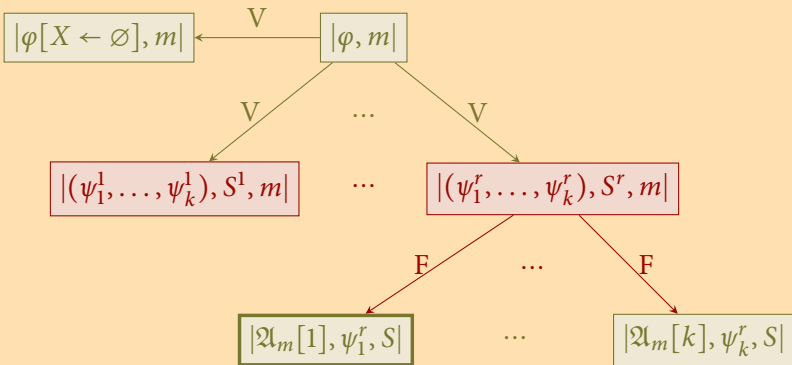
Game $\mathcal{G}^\exists(\varphi, m)$ for $\exists X\varphi(X)$



Game $\mathcal{G}^\exists(\varphi, m)$ for $\exists X\varphi(X)$

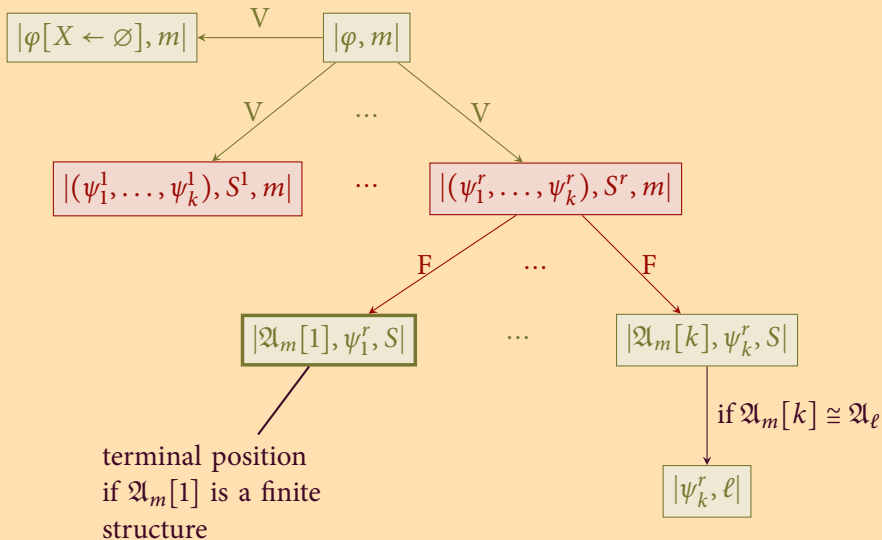


Game $\mathcal{G}^\exists(\varphi, m)$ for $\exists X\varphi(X)$



terminal position
if $\mathcal{Q}_m[1]$ is a finite
structure

Game $\mathcal{G}^\exists(\varphi, m)$ for $\exists X\varphi(X)$



Unbounding Quantifier

$UX\varphi(X)$: “for all $n \in \mathbb{N}$, $\exists X\varphi(X)$ with X finite and $|X| \geq n$ ”

- introduced by Bojańczyk in 2004
- WMSO+U proved to be decidable on trees only with very restricted quantification patterns
- general decidability results for WMSO+U only for words [B. 2009]

Unbounding Quantifier

$UX\varphi(X)$: “for all $n \in \mathbb{N}$, $\exists X\varphi(X)$ with X finite and $|X| \geq n$ ”

- introduced by Bojańczyk in 2004
- WMSO+U proved to be decidable on trees only with very restricted quantification patterns
- general decidability results for WMSO+U only for words [B. 2009]

A modification of the winning condition yields a game checking $UX\varphi(X)$.

↪ **Decidability of the WMSO+U theory of inductive structures.**

Overview

Finite Decompositions

WMSO on Inductive Structures (with T. Ganzow)

Infinite Decompositions

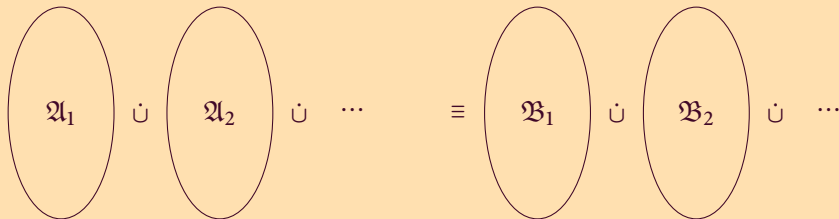
Cardinality Quantifiers (with V. Bárány and A. Rabinovich)

Decompositions on Infinite Sums and Products

Ehrenfeucht-Fraïsse Method works for infinite Sums and Products

$$\text{th}^n(\mathcal{A}_i) = \text{th}^n(\mathcal{B}_i) \text{ for all } i \implies \text{th}^n(\prod \mathcal{A}_i) = \text{th}^n(\prod \mathcal{B}_i)$$

$$\text{Th}^n(\mathcal{A}_i) = \text{Th}^n(\mathcal{B}_i) \text{ for all } i \implies \text{Th}^n(\bigcup_i \mathcal{A}_i) = \text{Th}^n(\bigcup_i \mathcal{B}_i)$$

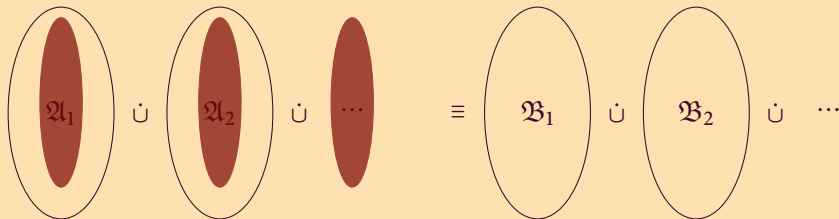


Decompositions on Infinite Sums and Products

Ehrenfeucht-Fraïssé Method works for infinite Sums and Products

$$\text{th}^n(\mathcal{A}_i) = \text{th}^n(\mathcal{B}_i) \text{ for all } i \implies \text{th}^n(\prod_i \mathcal{A}_i) = \text{th}^n(\prod_i \mathcal{B}_i)$$

$$\text{Th}^n(\mathcal{A}_i) = \text{Th}^n(\mathcal{B}_i) \text{ for all } i \implies \text{Th}^n(\bigcup_i \mathcal{A}_i) = \text{Th}^n(\bigcup_i \mathcal{B}_i)$$



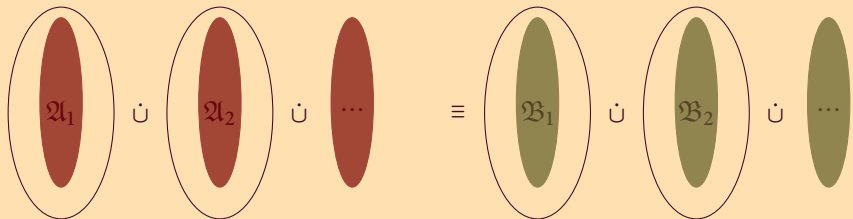
Spoiler: $X_1 \dot{\cup} X_2 \dot{\cup} \dots$

Decompositions on Infinite Sums and Products

Ehrenfeucht-Fraïsse Method works for infinite Sums and Products

$$\text{th}^n(\mathcal{A}_i) = \text{th}^n(\mathcal{B}_i) \text{ for all } i \implies \text{th}^n(\prod_i \mathcal{A}_i) = \text{th}^n(\prod_i \mathcal{B}_i)$$

$$\text{Th}^n(\mathcal{A}_i) = \text{Th}^n(\mathcal{B}_i) \text{ for all } i \implies \text{Th}^n(\bigcup_i \mathcal{A}_i) = \text{Th}^n(\bigcup_i \mathcal{B}_i)$$



Spoiler: $X_1 \dot{\cup} X_2 \dot{\cup} \dots$

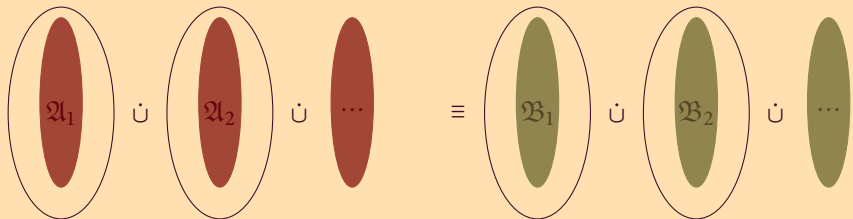
Duplicator: $\text{dup}_1(X_1) \dot{\cup} \text{dup}_2(X_2) \dot{\cup} \dots$

Decompositions on Infinite Sums and Products

Ehrenfeucht-Fraïssé Method works for infinite Sums and Products

$$\text{th}^n(\mathcal{A}_i) = \text{th}^n(\mathcal{B}_i) \text{ for all } i \implies \text{th}^n(\prod_i \mathcal{A}_i) = \text{th}^n(\prod_i \mathcal{B}_i)$$

$$\text{Th}^n(\mathcal{A}_i) = \text{Th}^n(\mathcal{B}_i) \text{ for all } i \implies \text{Th}^n(\bigcup_i \mathcal{A}_i) = \text{Th}^n(\bigcup_i \mathcal{B}_i)$$



Spoiler: $X_1 \dot{\cup} X_2 \dot{\cup} \dots$

Duplicator: $\text{dup}_1(X_1) \dot{\cup} \text{dup}_2(X_2) \dot{\cup} \dots$

Effective? Yes. Feferman, Vaught, Shelah, Läuchli, Gurevich, ...

Effective Decomposition on Infinite Sums

Theories as Finite Objects. Fix:

- Relational signature,
- Quantifier rank n ,
- Free variables X_1, \dots, X_l .

Hintikka formulas: a finite set $H_{n,l}$ such that:

- For every \mathcal{A} there is a **unique** $\tau \in H_{n,l}$ such that $\mathcal{A} \models \tau$.
- If $\tau_1, \tau_2 \in H_{n,l}$ and $\tau_1 \neq \tau_2$ then $\tau_1 \wedge \tau_2$ **is unsatisfiable**.
- If $\tau \in H_{n,l}$ and $\text{qr}(\varphi) \leq n$, then either $\tau \models \varphi$ **or** $\tau \models \neg\varphi$.

Effective Decomposition on Infinite Sums

Theories as Finite Objects. Fix:

- Relational signature,
- Quantifier rank n ,
- Free variables X_1, \dots, X_l .

Hintikka formulas: a finite set $H_{n,l}$ such that:

- For every \mathfrak{A} there is a **unique** $\tau \in H_{n,l}$ such that $\mathfrak{A} \models \tau$.
- If $\tau_1, \tau_2 \in H_{n,l}$ and $\tau_1 \neq \tau_2$ then $\tau_1 \wedge \tau_2$ is **unsatisfiable**.
- If $\tau \in H_{n,l}$ and $\text{qr}(\varphi) \leq n$, then either $\tau \models \varphi$ or $\tau \models \neg\varphi$.

Representation: labelling with Hintikka formulas (types)



Composition Theorem for Linear Orders

Ordered Sum: $\sum_{i \in I} \mathfrak{L}_i = \underbrace{\mathfrak{L}_{i_1} < \mathfrak{L}_{i_2} < \dots < \mathfrak{L}_{i_{\text{last}}}}_I$

Composition Theorem for Linear Orders

$$\text{Ordered Sum: } \sum_{i \in I} \mathcal{L}_i = \underbrace{\mathcal{L}_{i_1} < \mathcal{L}_{i_2} < \dots < \mathcal{L}_{i_{\text{last}}}}_I$$

Theorem [Shelah]:

- $\varphi(\overline{X}) \in \text{MSO}$, $|X| = m$, $\text{qr}(\varphi) = n$,
- $H_{n, m+1} = \tau_1(\overline{X}), \dots, \tau_k(\overline{X})$.

There exists $\vartheta(Q_1, \dots, Q_k)$ (**computable**) such that:

- For every linear order $\mathfrak{J} = (I, <^I)$ and $\{\mathcal{L}_i \mid i \in I\}$,
- and subsets V_1, \dots, V_m of $\sum_{i \in I} \mathcal{L}_i$ holds:

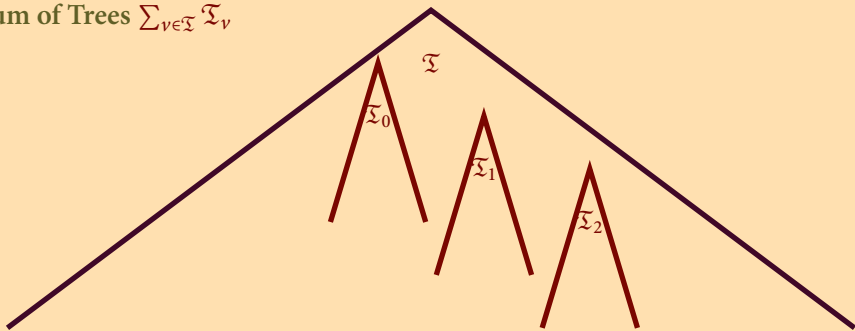
$$\sum_{i \in I} \mathcal{L}_i \models \varphi(\overline{V}) \iff \mathfrak{J} \models \vartheta(Q_1, \dots, Q_k)$$

where the predicates \overline{Q} are:

$$Q_r = \{i \in I \mid \text{Th}^n(\mathcal{L}_i, \overline{V}) \equiv \tau_r\}.$$

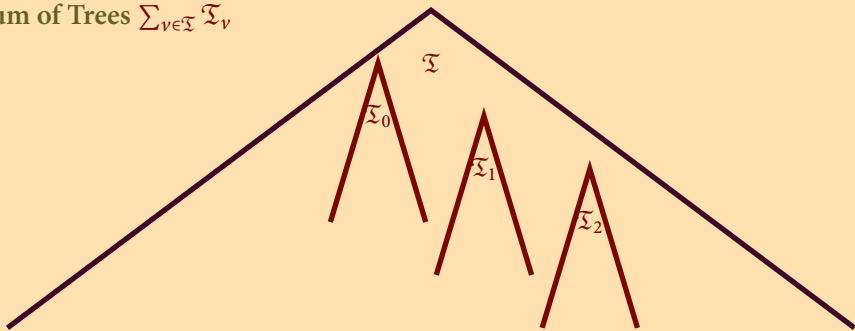
Composition Theorem for Trees

Sum of Trees $\sum_{v \in \mathcal{T}} \mathcal{T}_v$



Composition Theorem for Trees

Sum of Trees $\sum_{v \in \mathfrak{T}} \mathfrak{T}_v$



Composition Theorem for Trees [Shelah]

Let $H_{n,m+1} = \{\tau_1, \dots, \tau_k\}$ and $\text{qr}(\varphi) \leq n$. There exist ϑ :

$$\sum_{v \in \mathfrak{T}} \mathfrak{T}_i \models \varphi(\bar{X}) \iff \mathfrak{T} \models \vartheta(Q_1, \dots, Q_k)$$

where $Q_r = \{v \in \mathfrak{T} \mid \text{Th}^k(\mathfrak{T}_v, \bar{X}) \equiv \tau_r\}$. Analogous for Structure Sums.

Overview

Finite Decompositions

WMSO on Inductive Structures (with T. Ganzow)

Infinite Decompositions

Cardinality Quantifiers (with V. Bárány and A. Rabinovich)

Eliminating Second-Order Cardinality Quantifiers

Theorem (Elimination of cardinality quantifiers)

For each $\varphi(\bar{Y}) \in \text{MSO}(\exists^{\aleph_0}, \exists^{\aleph_1}, \exists^{2^{\aleph_0}})$ there exists (and is computable) $\psi(\bar{Y}) \in \text{MSO}$ which is equivalent to $\varphi(\bar{Y})$ over

- (1) *the class of finitely branching trees*
- (2) *the class of all ordinals*
- (3) *the class of all countable linear orders*

Eliminating Second-Order Cardinality Quantifiers

Theorem (Elimination of cardinality quantifiers)

For each $\varphi(\bar{Y}) \in \text{MSO}(\exists^{\aleph_0}, \exists^{\aleph_1}, \exists^{2^{\aleph_0}})$ there exists (and is computable) $\psi(\bar{Y}) \in \text{MSO}$ which is equivalent to $\varphi(\bar{Y})$ over

- (1) the class of *finitely branching trees*
- (2) the class of *all ordinals*
- (3) the class of *all countable linear orders*

Theorem (Continuum Hypothesis for MSO on trees and linear orders)

For every $\varphi(X, \bar{Y}) \in \text{MSO}$ over *finitely branching trees and countable linear orders* holds: $\exists^{\aleph_1} X \varphi(X, \bar{Y}) \equiv \exists^{2^{\aleph_0}} X \varphi(X, \bar{Y})$

Eliminating Second-Order Cardinality Quantifiers

Theorem (Elimination of cardinality quantifiers)

For each $\varphi(\bar{Y}) \in \text{MSO}(\exists^{\aleph_0}, \exists^{\aleph_1}, \exists^{2^{\aleph_0}})$ there exists (and is computable) $\psi(\bar{Y}) \in \text{MSO}$ which is equivalent to $\varphi(\bar{Y})$ over

- (1) the class of *finitely branching trees*
- (2) the class of *all ordinals*
- (3) the class of *all countable linear orders*

Theorem (Continuum Hypothesis for MSO on trees and linear orders)

For every $\varphi(X, \bar{Y}) \in \text{MSO}$ over *finitely branching trees and countable linear orders* holds: $\exists^{\aleph_1} X \varphi(X, \bar{Y}) \equiv \exists^{2^{\aleph_0}} X \varphi(X, \bar{Y})$

Theorem (Eliminating Second-Order Infinity Quantifier)

The following are equivalent: (on all structures)

- (1) There are only finitely many X satisfying $\varphi(X, \bar{Y})$
- (2) There is a finite Z such that any two $X_1 \neq X_2$ which both satisfy $\varphi(X_i, \bar{Y})$ differ on Z

Uncountability Quantifier on ω

When does $(\omega, <) \models \exists^{\aleph_1} X \varphi(X)$?

There exists an X with infinitely many disjoint **D-intervals**:

$$I \quad : \quad \exists X' \neq X \quad \text{Th}^k(X'|_I) = \text{Th}^k(X|_I)$$



Uncountability Quantifier on ω

When does $(\omega, <) \models \exists^{\aleph_1} X \varphi(X)$?

There exists an X with infinitely many disjoint **D-intervals**:

$$I \quad : \quad \exists X' \neq X \quad \text{Th}^k(X'|_I) = \text{Th}^k(X|_I)$$



(\Leftarrow): Composition Theorem $\text{Th}^k(I_1 + I_2 + \dots) = \text{Th}^k(I_1) \oplus \text{Th}^k(I_2) \oplus \dots$

Uncountability Quantifier on ω

When does $(\omega, <) \models \exists^{\aleph_1} X \varphi(X)$?

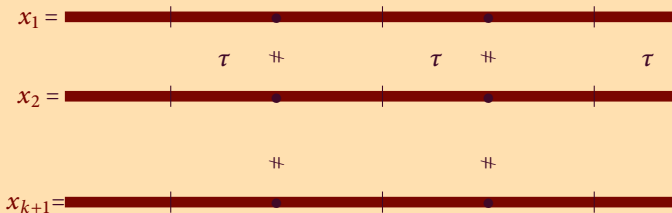
There exists an X with infinitely many disjoint **D-intervals**:

$$I : \exists X' \neq X \quad \text{Th}^k(X'|_I) = \text{Th}^k(X|_I)$$



(\Leftarrow): Composition Theorem $\text{Th}^k(I_1 + I_2 + \dots) = \text{Th}^k(I_1) \oplus \text{Th}^k(I_2) \oplus \dots$

(\Rightarrow): Finite number of Hintikka Formulas and Ramsey Theorem



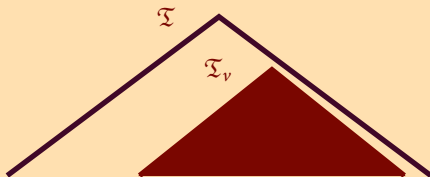
D-nodes and D-paths

D-trees: $\mathfrak{T}' \subseteq \mathfrak{T}$ is a **D-tree** for X if:

$$\exists X' \neq X \quad \text{Th}^k(X'|_{\mathfrak{T}'}) = \text{Th}^k(X|_{\mathfrak{T}'})$$

In the other case \mathfrak{T}' is a **U-tree**.

D-node v : subtree rooted at v is a **D-tree** (In MSO: $\varphi_{\text{DNODE}}(v, X)$)



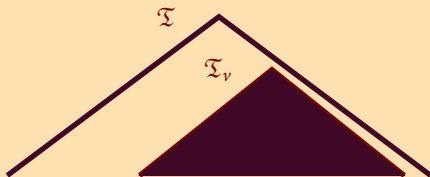
D-nodes and D-paths

D-trees: $\mathfrak{T}' \subseteq \mathfrak{T}$ is a **D-tree** for X if:

$$\exists X' \neq X \quad \text{Th}^k(X'|_{\mathfrak{T}'}) = \text{Th}^k(X|_{\mathfrak{T}'})$$

In the other case \mathfrak{T}' is a **U-tree**.

D-node v : subtree rooted at v is a **D-tree** (In MSO: $\varphi_{\text{DNODE}}(v, X)$)



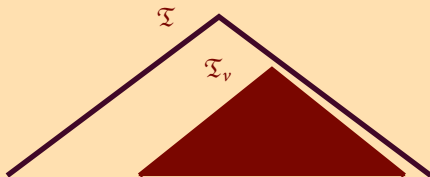
D-nodes and D-paths

D-trees: $\mathfrak{T}' \subseteq \mathfrak{T}$ is a **D-tree** for X if:

$$\exists X' \neq X \quad \text{Th}^k(X'|_{\mathfrak{T}'}) = \text{Th}^k(X|_{\mathfrak{T}'})$$

In the other case \mathfrak{T}' is a **U-tree**.

D-node v : subtree rooted at v is a **D-tree** (In MSO: $\varphi_{\text{DNODE}}(v, X)$)



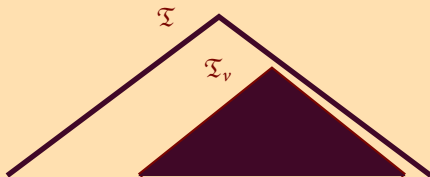
D-nodes and D-paths

D-trees: $\mathfrak{T}' \subseteq \mathfrak{T}$ is a **D-tree** for X if:

$$\exists X' \neq X \quad \text{Th}^k(X'|_{\mathfrak{T}'}) = \text{Th}^k(X|_{\mathfrak{T}'})$$

In the other case \mathfrak{T}' is a **U-tree**.

D-node v : subtree rooted at v is a **D-tree** (In MSO: $\varphi_{\text{DNODE}}(v, X)$)



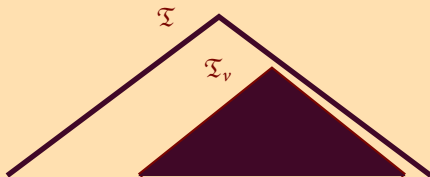
D-nodes and D-paths

D-trees: $\mathfrak{T}' \subseteq \mathfrak{T}$ is a **D-tree** for X if:

$$\exists X' \neq X \quad \text{Th}^k(X'|_{\mathfrak{T}'}) = \text{Th}^k(X|_{\mathfrak{T}'})$$

In the other case \mathfrak{T}' is a **U-tree**.

D-node v : subtree rooted at v is a **D-tree** (In MSO: $\varphi_{\text{DNODE}}(v, X)$)



D-path: a path of **D-nodes**.

By **Composition Theorem** the set of D-nodes is **prefix-closed**.

The Three Conditions

$\mathcal{T}, \bar{Y} \models \exists^{\aleph_1} X \varphi(X, \bar{Y})$ iff

The Three Conditions

$\mathcal{T}, \bar{Y} \models \exists^{\aleph_1} X \varphi(X, \bar{Y})$ iff

- (A) For some X satisfying φ there is an **infinite antichain** of **D-nodes**

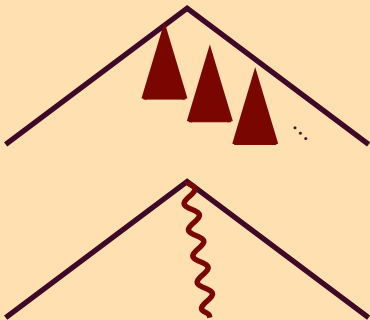


The Three Conditions

$\mathcal{T}, \bar{Y} \models \exists^{\aleph_1} X \varphi(X, \bar{Y})$ iff

- (A) For some X satisfying φ there is an **infinite antichain** of **D-nodes**

- (B) There is one **infinite D-path** for **uncountably many** X



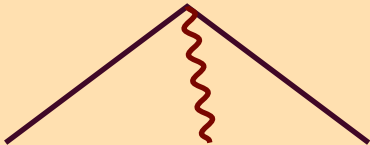
The Three Conditions

$\mathcal{T}, \bar{Y} \models \exists^{\aleph_1} X \varphi(X, \bar{Y})$ iff

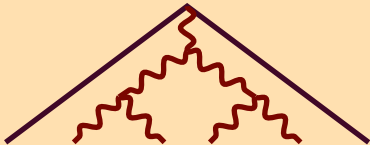
(A) For some X satisfying φ there is an **infinite antichain** of **D-nodes**



(B) There is one **infinite D-path** for **uncountably many** X

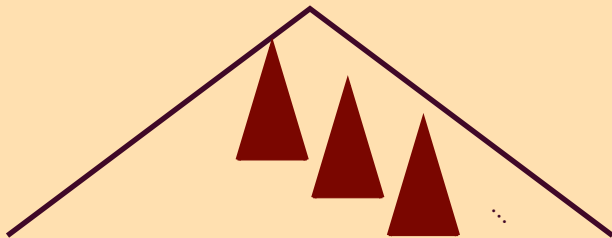


(C) The **set of infinite D-paths** for **all** X is **uncountable**



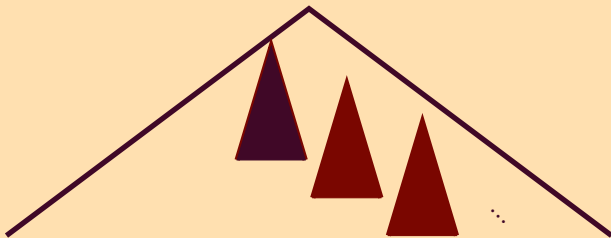
The Antichain Case

Condition (A) is sufficient by **Composition Theorem**



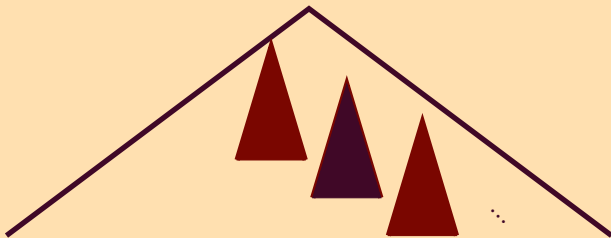
The Antichain Case

Condition (A) is sufficient by **Composition Theorem**



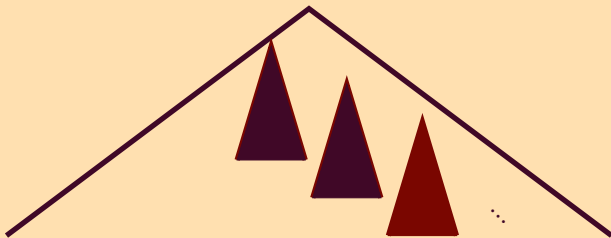
The Antichain Case

Condition (A) is sufficient by **Composition Theorem**



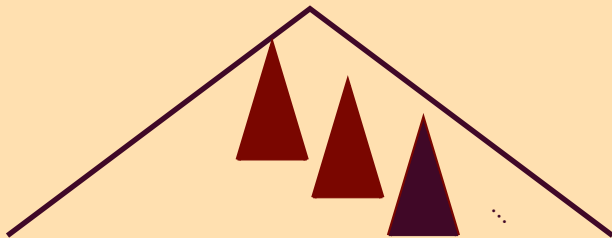
The Antichain Case

Condition (A) is sufficient by **Composition Theorem**



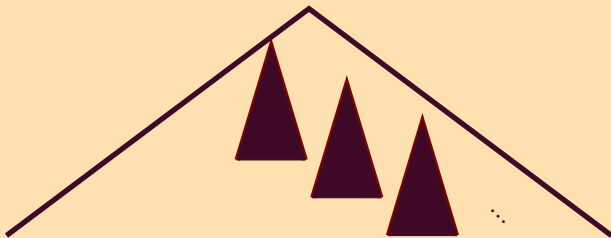
The Antichain Case

Condition (A) is sufficient by **Composition Theorem**



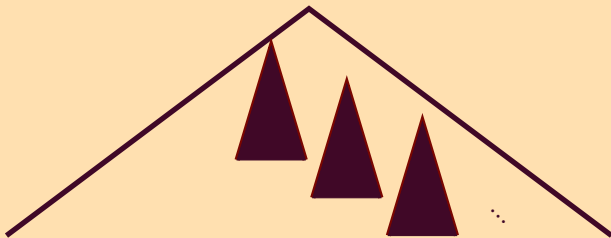
The Antichain Case

Condition (A) is sufficient by **Composition Theorem**

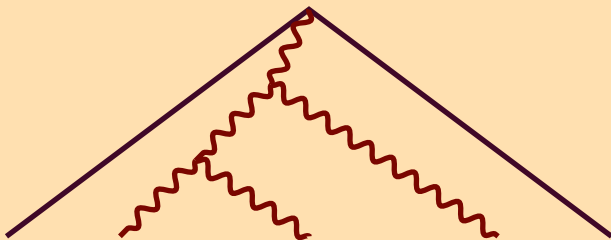


The Antichain Case

Condition (A) is sufficient by **Composition Theorem**



If not (A): each X has only finitely many D-paths



Path Case

D-path B for uncountably many X



Path Case

D-path B for uncountably many X



Decompose along B

$$\mathfrak{T} \models \varphi(X, \overline{Y}) \iff (B, <) \models \vartheta(P_1, \dots, P_l),$$

where $P_i = \{w \in B \mid (\mathcal{T}_{w \setminus B}, \{w\}) \models \tau_i\}$.

Path Case

D-path B for uncountably many X



Decompose along B

$$\mathfrak{T} \models \varphi(X, \bar{Y}) \iff (B, <) \models \vartheta(P_1, \dots, P_l),$$

where $P_i = \{w \in B \mid (\mathcal{T}_{w \setminus B}, \{w\}) \models \tau_i\}$.

Reduce to the question over $(\omega, <)$

There are **uncountably many** X with D-path B if

(Ba) For one X and infinitely many $w \in B$ the tree $\mathfrak{T}_{w \setminus B}$ is a **D-tree**, or

(Bc) For uncountably many \bar{P} the formula $\vartheta(\bar{P}) \wedge \mathbf{realisable}(\bar{P})$ holds on B

Uncountably many branches

(C): The set of infinite D-paths is uncountable

Uncountably many branches

(C): The set of infinite D-paths is uncountable

(C'): The set of infinite D-paths contains a perfect subset

- First: interpretation in the binary tree
- Second: apply the previous theorem

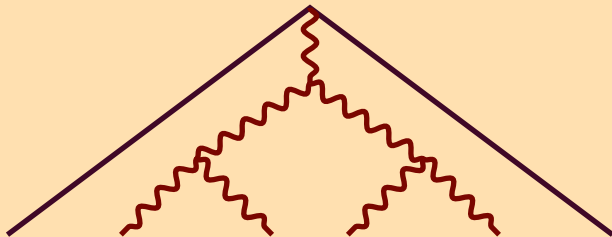
Uncountably many branches

(C): The set of infinite D-paths is uncountable

(C'): The set of infinite D-paths contains a perfect subset

- First: interpretation in the binary tree
- Second: apply the previous theorem

Perfect tree \mathfrak{T} : for each w there exist incomparable $u, v > w$



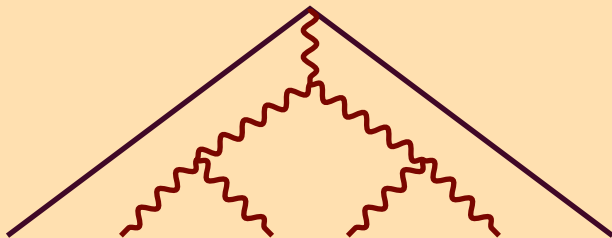
Uncountably many branches

(C): The set of infinite D-paths is uncountable

(C'): The set of infinite D-paths contains a perfect subset

- First: interpretation in the binary tree
- Second: apply the previous theorem

Perfect tree \mathfrak{T} : for each w there exist incomparable $u, v > w$



(C''): The set of infinite D-paths induces a perfect tree

Outlook

Decomposition method and automata theory

- allow to **understand** properties of regular structures
- are used to prove **decidability** of model-checking problems
- correspond to **parallel algorithms** in logic
- are an essential ingredient to **understand computer intelligence**

Challenges

- applications in **complexity theory** and **parallel algorithms**
- interactions with **combinatorics** and **Ramsey theory**
- extensions to **quantitative** and **probabilistic setting**

Outlook

Decomposition method and automata theory

- allow to **understand** properties of regular structures
- are used to prove **decidability** of model-checking problems
- correspond to **parallel algorithms** in logic
- are an essential ingredient to **understand computer intelligence**

Challenges

- applications in **complexity theory** and **parallel algorithms**
- interactions with **combinatorics** and **Ramsey theory**
- extensions to **quantitative** and **probabilistic setting**

Thank You