Products in the category of forests and p-morphisms via Delannoy paths on Cartesian products

Pietro Codara

Dipartimento di Informatica, Università degli Studi di Milano

(joint work with Ottavio M. D'Antona, and Vincenzo Marra)

TACL 2015, Ischia (NA) — June 25, 2015

Basic notions.

A category of forests

- A forest is a finite poset F such that for every $x \in F$, $\downarrow x$ is a chain. A tree is a forest with a bottom element.

A category of forests

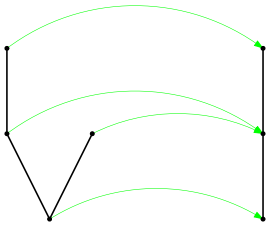
- A forest is a finite poset F such that for every $x \in F$, $\downarrow x$ is a chain. A tree is a forest with a bottom element.
- An order preserving map $f : F \longrightarrow G$ is a p-morphism (or is open) iff, for every $x \in F$,

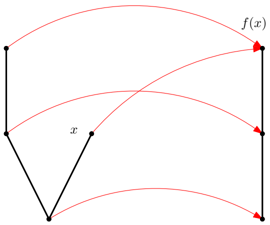
$$f(\downarrow x) = \downarrow f(x)\,.$$

A category of forests

- A forest is a finite poset F such that for every $x \in F$, $\downarrow x$ is a chain. A tree is a forest with a bottom element.
- An order preserving map $f : F \longrightarrow G$ is a p-morphism (or is open) iff, for every $x \in F$,

$$f(\downarrow x) = \downarrow f(x)\,.$$





In this talk.

In this talk

We show how to compute products in the category F of forests and p-morphisms.

In this talk

We show how to compute products in the category F of forests and p-morphisms.

- Dually, via Esakia duality, we show how to compute co-products of finitely presented Gödel algebras.

In this talk

We show how to compute products in the category F of forests and p-morphisms.

- Dually, via Esakia duality, we show how to compute co-products of finitely presented Gödel algebras.

Various techniques to perform this computation are known. Why should the one presented here be interesting?

In this talk

We show how to compute products in the category F of forests and p-morphisms.

- Dually, via Esakia duality, we show how to compute co-products of finitely presented Gödel algebras.

Various techniques to perform this computation are known. Why should the one presented here be interesting?

- In the category F products are not Cartesian.
- Our construction is “as Cartesian as possible”.

Product of forests,
known combinatorial methods.

Ordered partitions, and merged shuffles

[D'Antona, O.M., and Marra, V., *Computing coproducts of finitely presented Gödel algebras*, Ann. Pure Appl. Logic 142 (2006), 202–211]

Ordered partitions, and merged shuffles

[D'Antona, O.M., and Marra, V., *Computing coproducts of finitely presented Gödel algebras*, Ann. Pure Appl. Logic 142 (2006), 202–211]

- An ordered partition σ is a sequence of pairwise disjoint nonempty sets, called blocks. The union of the blocks of σ is the support of σ.

Ordered partitions, and merged shuffles

[D'Antona, O.M., and Marra, V., *Computing coproducts of finitely presented Gödel algebras*, Ann. Pure Appl. Logic 142 (2006), 202–211]

- An ordered partition σ is a sequence of pairwise disjoint nonempty sets, called blocks. The union of the blocks of σ is the support of σ.
- Let σ and τ be ordered partitions with disjoint supports. An ordered partition θ is a shuffle of σ and τ iff σ and τ are subsequences of θ, and $\mathbf{supp}\theta = \mathbf{supp}\sigma \cup \mathbf{supp}\tau$.

Ordered partitions, and merged shuffles

[D'Antona, O.M., and Marra, V., *Computing coproducts of finitely presented Gödel algebras*, Ann. Pure Appl. Logic 142 (2006), 202–211]

- An ordered partition σ is a sequence of pairwise disjoint nonempty sets, called blocks. The union of the blocks of σ is the support of σ.
- Let σ and τ be ordered partitions with disjoint supports. An ordered partition θ is a shuffle of σ and τ iff σ and τ are subsequences of θ, and $\operatorname{supp}\theta = \operatorname{supp}\sigma \cup \operatorname{supp}\tau$.
- A merged shuffle is obtained from a shuffle θ, by merging some consecutive pairs of blocks $A, B \in \theta$, with $A \in \sigma$, and $B \in \tau$.

Ordered partitions, and merged shuffles

[D'Antona, O.M., and Marra, V., *Computing coproducts of finitely presented Gödel algebras*, Ann. Pure Appl. Logic 142 (2006), 202–211]

- An ordered partition σ is a sequence of pairwise disjoint nonempty sets, called blocks. The union of the blocks of σ is the support of σ.
- Let σ and τ be ordered partitions with disjoint supports. An ordered partition θ is a shuffle of σ and τ iff σ and τ are subsequences of θ, and $\mathbf{supp}\theta = \mathbf{supp}\sigma \cup \mathbf{supp}\tau$.
- A merged shuffle is obtained from a shuffle θ, by merging some consecutive pairs of blocks $A, B \in \theta$, with $A \in \sigma$, and $B \in \tau$.

Example. Let $\sigma = \{a|b\}$ and $\tau = \{x\}$. The merged shuffles of σ and τ are: $\{a|b|x\}$, $\{a|x|b\}$, $\{x|a|b\}$, $\{a|bx\}$, $\{ax|b\}$.

Trees of ordered partitions

Given ordered partitions $\sigma = \{A_1|\ldots|A_m\}$, and $\tau = \{B_1|\ldots|B_n\}$ with $m \leq n$ we write $\sigma \leq \tau$ iff $A_i = B_i$ for every $i \in \{1,\ldots,m\}$.

Trees of ordered partitions

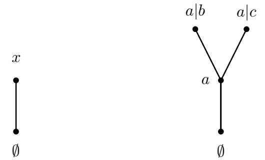
Given ordered partitions $\sigma = \{A_1|\ldots|A_m\}$, and $\tau = \{B_1|\ldots|B_n\}$ with $m \leq n$ we write $\sigma \leq \tau$ iff $A_i = B_i$ for every $i \in \{1, \ldots, m\}$.

One can label trees with ordered partitions...

Trees of ordered partitions

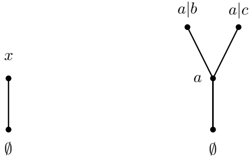
Given ordered partitions $\sigma = \{A_1|\ldots|A_m\}$, and $\tau = \{B_1|\ldots|B_n\}$ with $m \leq n$ we write $\sigma \leq \tau$ iff $A_i = B_i$ for every $i \in \{1,\ldots,m\}$.

One can label trees with ordered partitions...



Trees of ordered partitions

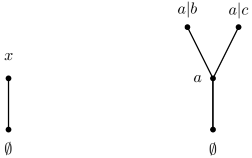
Given ordered partitions $\sigma = \{A_1|\ldots|A_m\}$, and $\tau = \{B_1|\ldots|B_n\}$ with $m \leq n$ we write $\sigma \leq \tau$ iff $A_i = B_i$ for every $i \in \{1,\ldots,m\}$. One can label trees with ordered partitions...



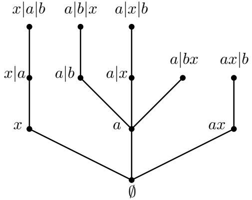
One can build a tree from a set of ordered partitions. The tree of merged shuffles of $\sigma = \{a|b\}$ and $\tau = \{x\}$ is...

Trees of ordered partitions

Given ordered partitions $\sigma = \{A_1|\ldots|A_m\}$, and $\tau = \{B_1|\ldots|B_n\}$ with $m \leq n$ we write $\sigma \leq \tau$ iff $A_i = B_i$ for every $i \in \{1, \ldots, m\}$. One can label trees with ordered partitions...



One can build a tree from a set of ordered partitions. The tree of merged shuffles of $\sigma = \{a|b\}$ and $\tau = \{x\}$ is...



Product of forests

- Let $F = \{T_1, \dots, T_r\}$ and $G = \{U_1, \dots, U_s\}$ be forests.
  $F \times_{\mathsf{F}} G = \{T_i \times_{\mathsf{F}} U_j\}$, $i \in \{1, \dots, r\}$, $j \in \{1, \dots, s\}$.

Product of forests

- Let $F = \{T_1, \ldots, T_r\}$ and $G = \{U_1, \ldots, U_s\}$ be forests. $F \times_{\mathsf{F}} G = \{T_i \times_{\mathsf{F}} U_j\}$, $i \in \{1, \ldots, r\}$, $j \in \{1, \ldots, s\}$.
- The problem of describing $F \times_{\mathsf{F}} G$ is reduced to that of describing its trees.

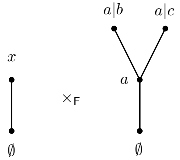
Product of forests

- Let $F = \{T_1, \ldots, T_r\}$ and $G = \{U_1, \ldots, U_s\}$ be forests.
  $F \times_\mathsf{F} G = \{T_i \times_\mathsf{F} U_j\}$, $i \in \{1, \ldots, r\}$, $j \in \{1, \ldots, s\}$.
- The problem of describing $F \times_\mathsf{F} G$ is reduced to that of describing its trees.

How to compute the product of trees?

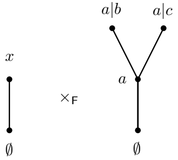
Product of trees

Computing the product of trees (an example).



Product of trees

Computing the product of trees (an example).

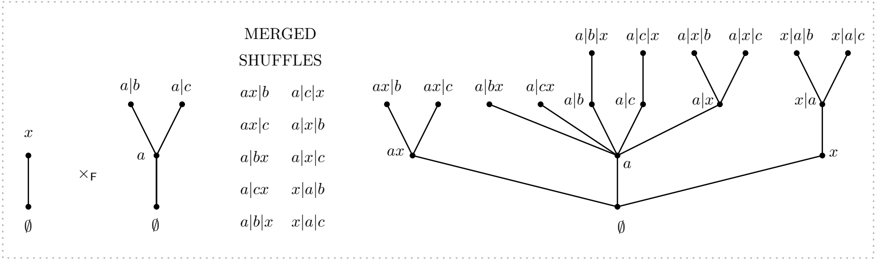


MERGED SHUFFLES

$ax\|b$	$a\|c\|x$
$ax\|c$	$a\|x\|b$
$a\|bx$	$a\|x\|c$
$a\|cx$	$x\|a\|b$
$a\|b\|x$	$x\|a\|c$

Product of trees

Computing the product of trees (an example).



Product of forests, a recursive construction

[Aguzzoli, S., Bova, S., and Gerla, B., *Free Algebra and Functional Representation for Fuzzy Logics*, in Handbook of Mathematical Fuzzy Logic - Vol. 2, P. Cintula, P. Hájek, C. Noguera, eds., Studies in Logic, Vol. 38, College Pubblications, London (2011), 713–791]

Product of forests, a recursive construction

[Aguzzoli, S., Bova, S., and Gerla, B., *Free Algebra and Functional Representation for Fuzzy Logics*, in Handbook of Mathematical Fuzzy Logic - Vol. 2, P. Cintula, P. Hájek, C. Noguera, eds., Studies in Logic, Vol. 38, College Pubblications, London (2011), 713–791]

Let F, G, and H be three forests.

Product of forests, a recursive construction

[Aguzzoli, S., Bova, S., and Gerla, B., *Free Algebra and Functional Representation for Fuzzy Logics*, in Handbook of Mathematical Fuzzy Logic - Vol. 2, P. Cintula, P. Hájek, C. Noguera, eds., Studies in Logic, Vol. 38, College Pubblications, London (2011), 713–791]

Let F, G, and H be three forests.

- If $|F| = 1$, then $F \times_{\mathsf{F}} G \cong G$.

Product of forests, a recursive construction

[Aguzzoli, S., Bova, S., and Gerla, B., *Free Algebra and Functional Representation for Fuzzy Logics*, in Handbook of Mathematical Fuzzy Logic - Vol. 2, P. Cintula, P. Hájek, C. Noguera, eds., Studies in Logic, Vol. 38, College Pubblications, London (2011), 713–791]

Let F, G, and H be three forests.

- If $|F| = 1$, then $F \times_{\mathsf{F}} G \cong G$.
- $(F + G) \times_{\mathsf{F}} H \cong (F \times_{\mathsf{F}} H) + (G \times_{\mathsf{F}} H)$.

Product of forests, a recursive construction

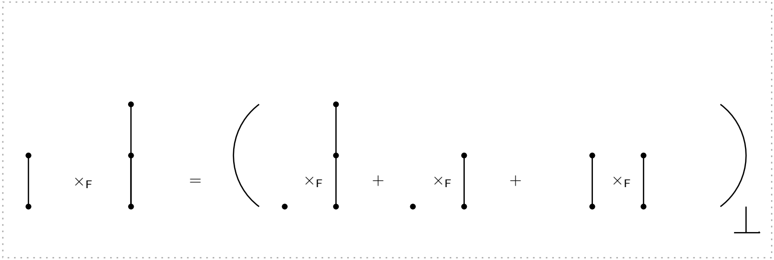
[Aguzzoli, S., Bova, S., and Gerla, B., *Free Algebra and Functional Representation for Fuzzy Logics*, in Handbook of Mathematical Fuzzy Logic - Vol. 2, P. Cintula, P. Hájek, C. Noguera, eds., Studies in Logic, Vol. 38, College Pubblications, London (2011), 713–791]

Let F, G, and H be three forests.

- If $|F| = 1$, then $F \times_{\mathsf{F}} G \cong G$.
- $(F + G) \times_{\mathsf{F}} H \cong (F \times_{\mathsf{F}} H) + (G \times_{\mathsf{F}} H)$.
- $F_{\perp} \times_{\mathsf{F}} G_{\perp} \cong ((F \times_{\mathsf{F}} G_{\perp}) + (F \times_{\mathsf{F}} G) + (F_{\perp} \times_{\mathsf{F}} G))_{\perp}$.

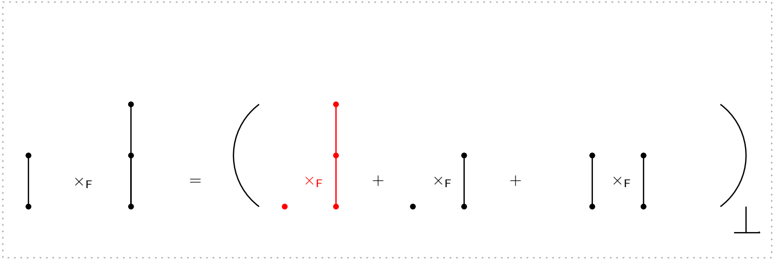
Product of trees, a recursive formula

Computing the product of trees (an example).



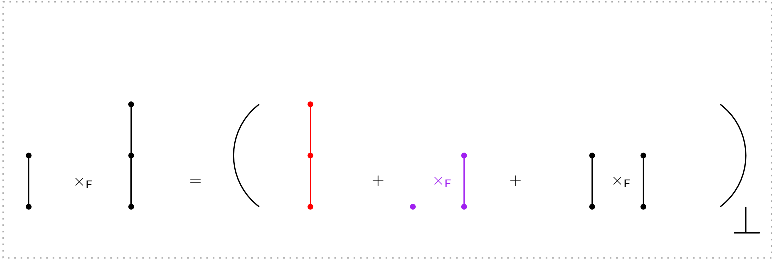
Product of trees, a recursive formula

Computing the product of trees (an example).



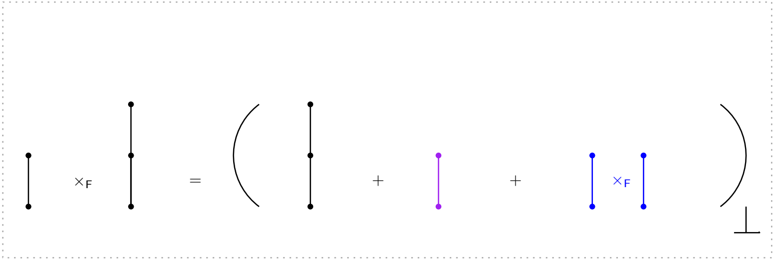
Product of trees, a recursive formula

Computing the product of trees (an example).



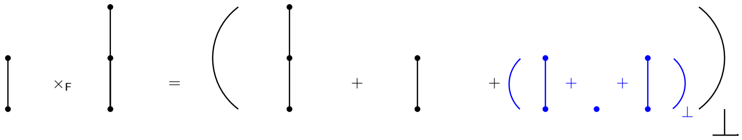
Product of trees, a recursive formula

Computing the product of trees (an example).



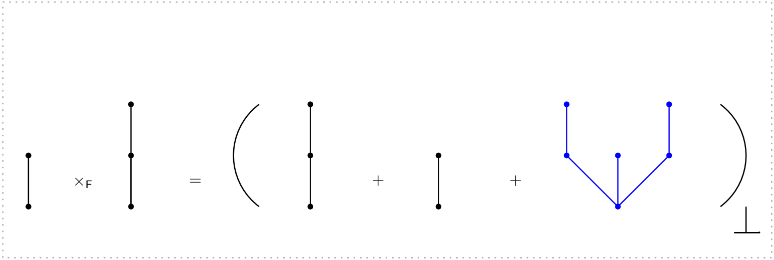
Product of trees, a recursive formula

Computing the product of trees (an example).



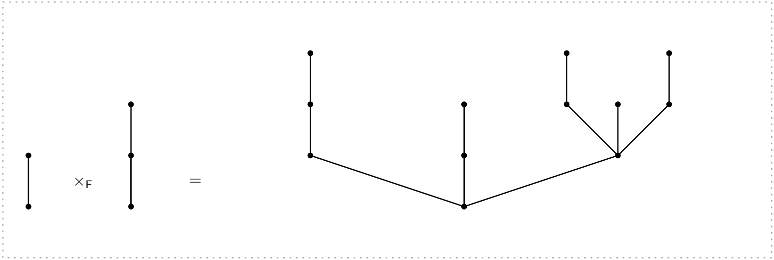
Product of trees, a recursive formula

Computing the product of trees (an example).



Product of trees, a recursive formula

Computing the product of trees (an example).



Products of forests via Delannoy paths on Cartesian products.

Classical Delannoy paths, and paths on posets

- A Delannoy path is a path on the first integer quadrant $\mathbb{N}^2 \subseteq \mathbb{Z}^2$ that starts from the origin and only uses northward, eastward, and north-eastward steps.

Classical Delannoy paths, and paths on posets

- A Delannoy path is a path on the first integer quadrant $\mathbb{N}^2 \subseteq \mathbb{Z}^2$ that starts from the origin and only uses northward, eastward, and north-eastward steps.
- A (finite) path on a poset P is a non-empty sequence $\langle p_1, p_2, \ldots, p_h \rangle$ of elements of P such that $p_i < p_j$ whenever $i < j$. (A path on P is therefore the same thing as a chain of P.)

Classical Delannoy paths, and paths on posets

- A Delannoy path is a path on the first integer quadrant $\mathbb{N}^2 \subseteq \mathbb{Z}^2$ that starts from the origin and only uses northward, eastward, and north-eastward steps.
- A (finite) path on a poset P is a non-empty sequence $\langle p_1, p_2, \ldots, p_h \rangle$ of elements of P such that $p_i < p_j$ whenever $i < j$. (A path on P is therefore the same thing as a chain of P.)
- For each $i \in \{1, \ldots, n-1\}$, the pair p_i, p_{i+1} is called a step of the path.

Classical Delannoy paths, and paths on posets

- A Delannoy path is a path on the first integer quadrant $\mathbb{N}^2 \subseteq \mathbb{Z}^2$ that starts from the origin and only uses northward, eastward, and north-eastward steps.
- A (finite) path on a poset P is a non-empty sequence $\langle p_1, p_2, \ldots, p_h \rangle$ of elements of P such that $p_i < p_j$ whenever $i < j$. (A path on P is therefore the same thing as a chain of P.)
- For each $i \in \{1, \ldots, n-1\}$, the pair p_i, p_{i+1} is called a step of the path.
- Given a poset P, and two elements $p, q \in P$, we write $p \lessdot q$ to indicate that q covers p in P, that is, $p < q$ and for every $s \in P$, if $p \leq s \leq q$, then either $s = p$ or $s = q$.

Delannoy paths on Cartesian products of posets

Definition

Let $P_1, \ldots, P_n$ be posets, and let $P = P_1 \times \cdots \times P_n$ be their (Cartesian) product. Let $\langle p_1, \ldots, p_h \rangle$ be a path on P.

The step from $p_i = (p_{i,1}, \ldots, p_{i,n})$ to $p_{i+1} = (p_{i+1,1}, \ldots, p_{i+1,n})$ is a Delannoy step, written $p_i \prec p_{i+1}$, if and only if there exists $k \in \{1, \ldots, n\}$ such that $p_{i,k} \neq p_{i+1,k}$, and for each $j \in \{1, \ldots, n\}$, $p_{i,j} \trianglelefteq p_{i+1,j}$.

The path $\langle p_1, \ldots, p_h \rangle$ on P is a Delannoy path if and only if p_1 is a minimal element of P, and for each $i \in \{1, \ldots, n-1\}$, $p_i \prec p_{i+1}$.

Delannoy paths on Cartesian products of posets

- A Delannoy path on P is thus a sequence of Delannoy steps starting from a minimal element of P.

Delannoy paths on Cartesian products of posets

- A Delannoy path on P is thus a sequence of Delannoy steps starting from a minimal element of P.
- Delannoy paths on a poset $P = P_1 \times \cdots \times P_n$ can be partially ordered by $\langle q_1, \ldots, q_m \rangle \leq \langle p_1, \ldots, p_h \rangle$ if and only if $m \leq h$ and $q_i = p_i$ for each $i \in \{1, \ldots, m\}$.

Delannoy paths on Cartesian products of posets

- A Delannoy path on P is thus a sequence of Delannoy steps starting from a minimal element of P.
- Delannoy paths on a poset $P = P_1 \times \cdots \times P_n$ can be partially ordered by $\langle q_1, \ldots, q_m \rangle \leq \langle p_1, \ldots, p_h \rangle$ if and only if $m \leq h$ and $q_i = p_i$ for each $i \in \{1, \ldots, m\}$.
- We denote by $\mathcal{D}(P_1, \ldots, P_n)$ the poset of all Delannoy paths on P.

Delannoy paths on Cartesian products of posets

- A Delannoy path on P is thus a sequence of Delannoy steps starting from a minimal element of P.
- Delannoy paths on a poset $P = P_1 \times \cdots \times P_n$ can be partially ordered by $\langle q_1, \ldots, q_m \rangle \leq \langle p_1, \ldots, p_h \rangle$ if and only if $m \leq h$ and $q_i = p_i$ for each $i \in \{1, \ldots, m\}$.
- We denote by $\mathcal{D}(P_1, \ldots, P_n)$ the poset of all Delannoy paths on P.
- Clearly, $\mathcal{D}(P_1, \ldots, P_n)$ is a forest.

Product in F via Delannoy paths

Theorem

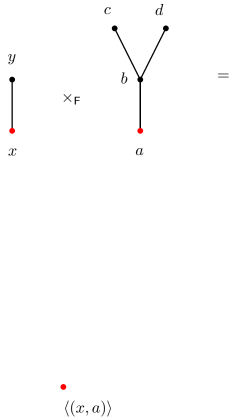
Let F and G be forests. Then $\mathcal{D}(F, G)$ is the product $F \times_{\mathsf{F}} G$ in the category F:

$$F \xleftarrow{\pi_F} F \times_{\mathsf{F}} G \xrightarrow{\pi_G} G\,.$$

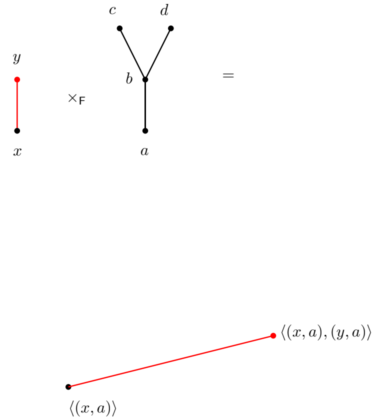
Let $d \in F \times_{\mathsf{F}} G$, with $d = \langle (f_1, g_1), \ldots, (f_n, g_n) \rangle$. The projection functions $\pi_F : F \times_{\mathsf{F}} G \to F$ and $\pi_G : F \times_{\mathsf{F}} G \to G$ are defined by

$$\pi_F(d) = f_n, \text{ and } \ \pi_G(d) = g_n\,.$$

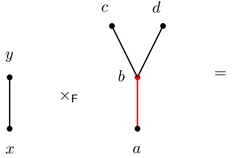
Computing the product in F, an example

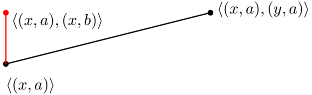


Computing the product in F, an example

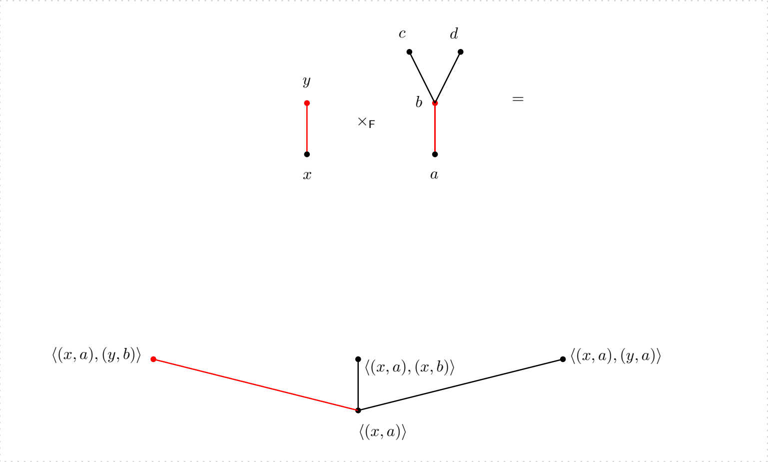


Computing the product in F, an example

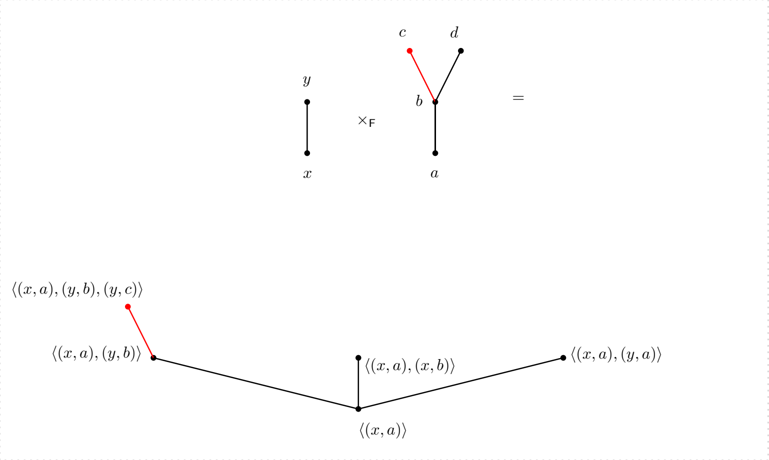




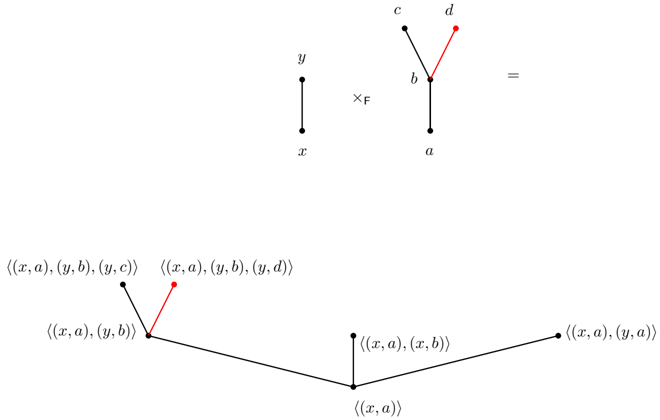
Computing the product in F, an example



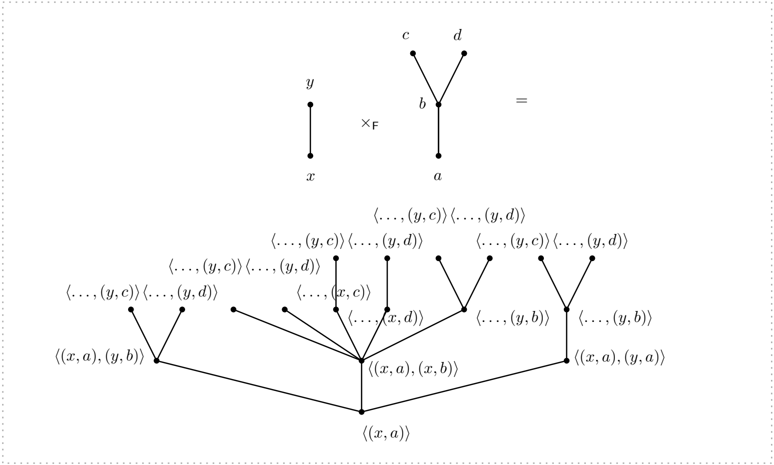
Computing the product in F, an example



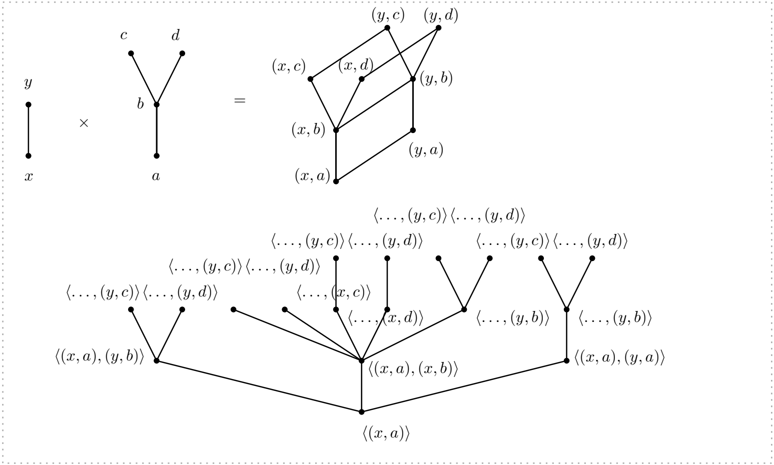
Computing the product in F, an example



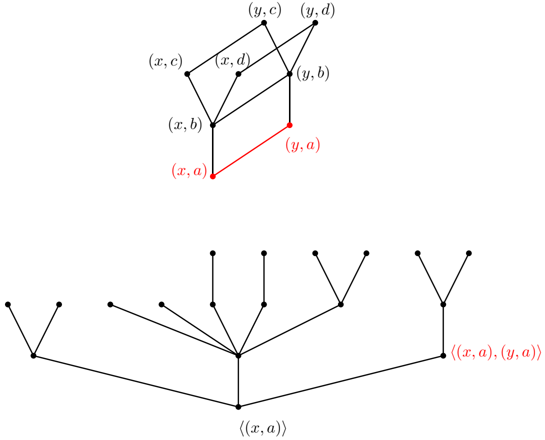
Computing the product in F, an example



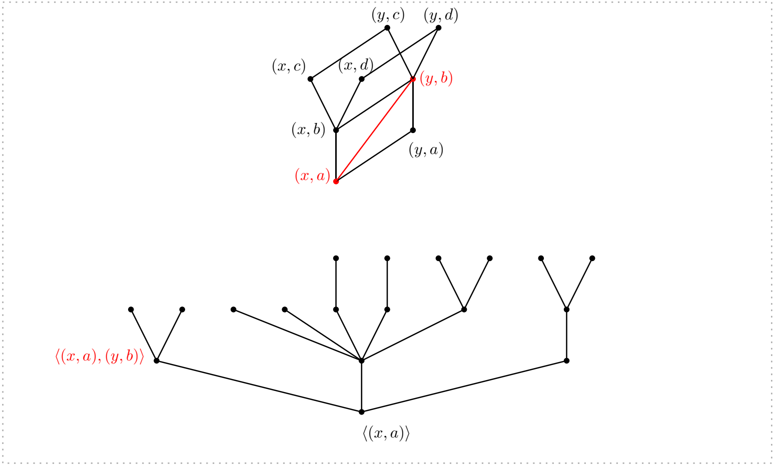
Computing the product in F, an example



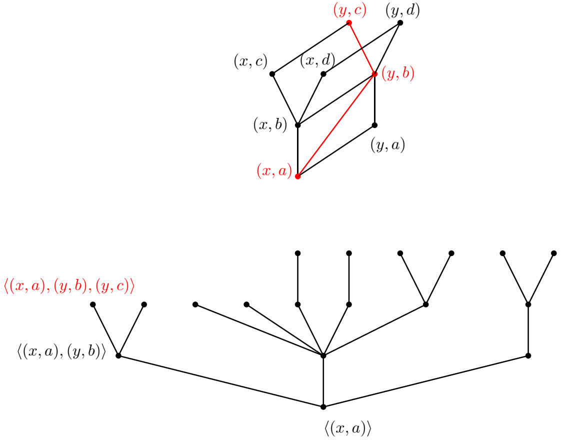
Computing the product in F, an example



Computing the product in F, an example



Computing the product in F, an example



Enumeration.

Delannoy numbers

The Delannoy number $D_{n,m}$ counts the number of Delannoy paths from $(0,0)$ to (n,m). Delannoy numbers satisfy the following recurrence relation.

$$D_{n,m} = D_{n-1,m} + D_{n,m-1} + D_{n-1,m-1}$$

Delannoy numbers

The Delannoy number $D_{n,m}$ counts the number of Delannoy paths from $(0,0)$ to (n,m). Delannoy numbers satisfy the following recurrence relation.

$$D_{n,m} = D_{n-1,m} + D_{n,m-1} + D_{n-1,m-1}$$

The following table shows some values of Delannoy numbers.

1	1	1	1	1	1	1	1
1	3	5	7	9	11	13	15
1	5	13	25	41	61	85	113
1	7	25	63	129	231	377	575
1	9	41	129	321	681	1289	2241
1	11	61	231	681	1683	3653	7183

A Formula for the number of elements of the products

Let T, U be trees.

$$|T \times_{\mathsf{F}} U| = \sum_{i \geq 0} \sum_{j \geq 0} t_i u_j D_{i,j} \ ,$$

where t_i is the number of elements at level i of T, and u_j is the number of elements at level j of U.

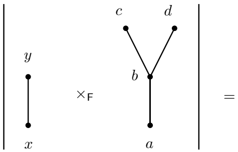
A Formula for the number of elements of the products

Let T, U be trees.

$$|T \times_{\mathsf{F}} U| = \sum_{i \geq 0} \sum_{j \geq 0} t_i u_j D_{i,j} \ ,$$

where t_i is the number of elements at level i of T, and u_j is the number of elements at level j of U.

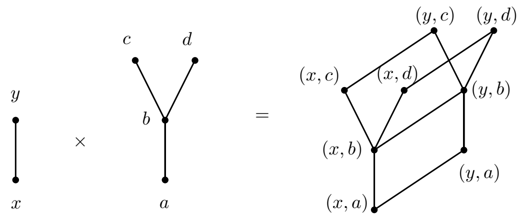
Example.



$= 1 \cdot 1 \cdot D_{0,0} + 1 \cdot 1 \cdot D_{0,1} + 1 \cdot 2 \cdot D_{0,2} + 1 \cdot 1 \cdot D_{1,0} + 1 \cdot 1 \cdot D_{1,1} + 1 \cdot 2 \cdot D_{1,2} =$
$= 1 + 1 + 2 + 1 + 3 + 10 = 18.$

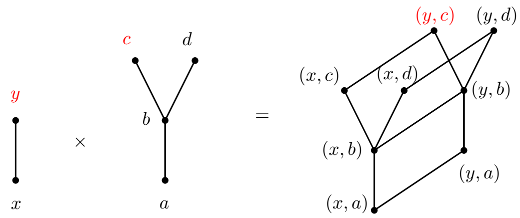
A Formula for the number of elements of the products

$$|T \times_{\mathsf{F}} U| = \sum_{i \geq 0} \sum_{j \geq 0} t_i u_j D_{i,j}$$



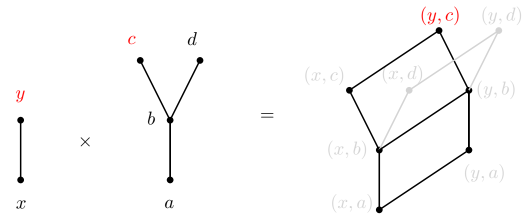
A Formula for the number of elements of the products

$$|T \times_{\mathsf{F}} U| = \sum_{i \geq 0} \sum_{j \geq 0} t_i u_j D_{i,j}$$



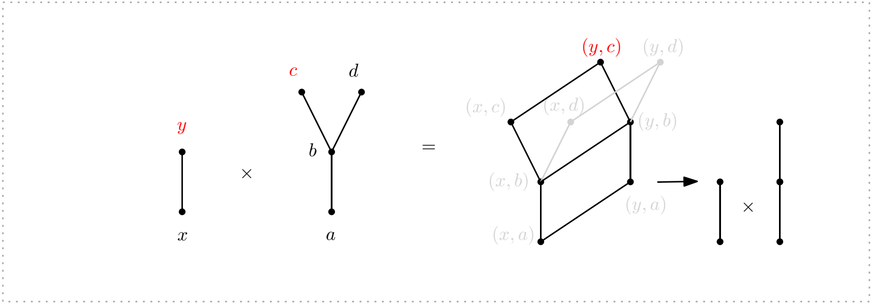
A Formula for the number of elements of the products

$$|T \times_{\mathsf{F}} U| = \sum_{i \geq 0} \sum_{j \geq 0} t_i u_j D_{i,j}$$



A Formula for the number of elements of the products

$$|T \times_{\mathsf{F}} U| = \sum_{i \geq 0} \sum_{j \geq 0} t_i u_j D_{i,j}$$



References

- D'Antona, O.M., and Marra, V.: Computing coproducts of finitely presented Gödel algebras. Ann. Pure Appl. Logic 142, 202–211 (2006)
- Aguzzoli, S., Bova, S., and Gerla, B.: Free Algebra and Functional Representation for Fuzzy Logics. in Handbook of Mathematical Fuzzy Logic - Vol. 2, P. Cintula, P. Hájek, C. Noguera, eds., Studies in Logic, Vol. 38, College Pubblications, London, 713–791 (2011)
- Codara, P., D'Antona, O.M., and Marra, V.: Propositional Gödel Logic and Delannoy Paths. IEEE International Fuzzy Systems Conference (FUZZ-IEEE) 2007, 1228–1232 (2007)

Thank you for your attention.