Transcendental Syntax

The dynamics of logic programs and tilings, applied to Linear Logic

Team LoVe – LIPN Université Sorbone Paris Nord

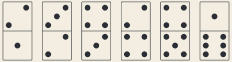
Boris ENG

Stellar Resolution

From tiles to logic programs

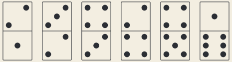
Wang tiles

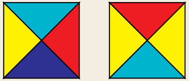
Hao Wang (1961)

Dominos. 

Wang tiles

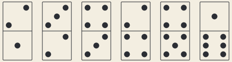
Hao Wang (1961)

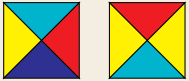
Dominos. 

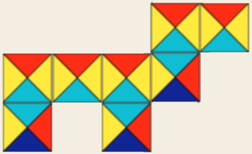
Wang set. 

Wang tiles

Hao Wang (1961)

Dominos. 

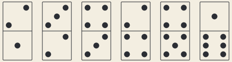
Wang set. 

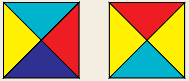


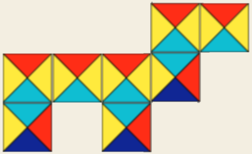
Tiling. $\alpha : \mathbf{Z}^2 \longrightarrow T$, adjacent tiles : sides of matching colours.

Wang tiles

Hao Wang (1961)

Dominos. 

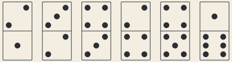
Wang set. 

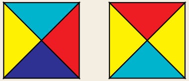
Tiling.  $\alpha : \mathbf{Z}^2 \longrightarrow T$, adjacent tiles : sides of matching colours.

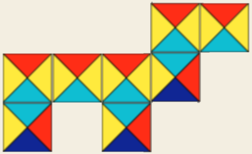
Turing-complete. by simulating space-time diagram.

Wang tiles

Hao Wang (1961)

Dominos. 

Wang set. 

Tiling.  $\alpha : \mathbf{Z}^2 \longrightarrow T$, adjacent tiles : sides of matching colours.

Turing-complete. by simulating space-time diagram.

Generalisations. different matchability (e.g DNA computing), higher dimensions (e.g $\mathbf{Z}^3$).

Flexible tiles

Nataša Jonoska (around 2000)

Coming from DNA computing.

Flexible tiles

Nataša Jonoska (around 2000)

Coming from DNA computing.

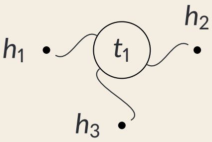
Set of borders H and an involution $\theta : H \longrightarrow H$ defining complementary.

Flexible tiles

Nataša Jonoska (around 2000)

Coming from DNA computing.

Set of borders H and an involution $\theta : H \to H$ defining complementary.

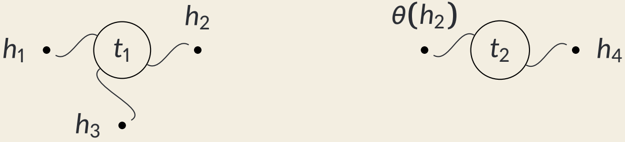


Flexible tiles

Nataša Jonoska (around 2000)

Coming from DNA computing.

Set of borders H and an involution $\theta : H \to H$ defining complementary.

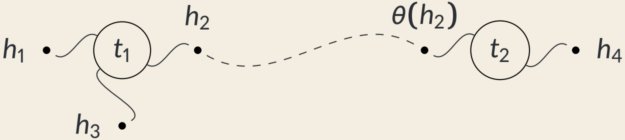


Flexible tiles

Nataša Jonoska (around 2000)

Coming from DNA computing.

Set of borders H and an involution $\theta : H \to H$ defining complementary.

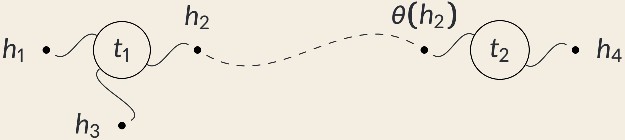


Flexible tiles

Nataša Jonoska (around 2000)

Coming from DNA computing.

Set of borders H and an involution $\theta : H \to H$ defining complementary.



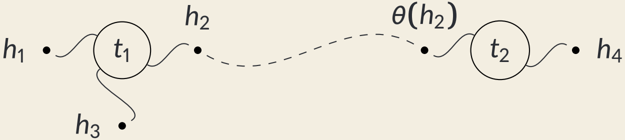
- No geometrical constraint (planarity).

Flexible tiles

Nataša Jonoska (around 2000)

Coming from DNA computing.

Set of borders H and an involution $\theta : H \longrightarrow H$ defining complementary.



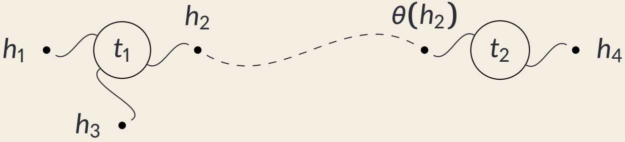
- No geometrical constraint (planarity).
- Simulates usual "rigid tiles" (on $\mathbf{Z}^n$).

Flexible tiles

Nataša Jonoska (around 2000)

Coming from DNA computing.

Set of borders H and an involution $\theta : H \to H$ defining complementary.



- No geometrical constraint (planarity).
- Simulates usual "rigid tiles" (on $\mathbf{Z}^n$).
- Related to `NTIME` classes.

  "*Complexity classes for self-assembling flexible tiles*" (Jonoska et al.).

Stellar Resolution (background)

Jean-Yves Girard (2013)

Flexible tiles on first-order terms.

Stellar Resolution (background)

Jean-Yves Girard (2013)

Flexible tiles on first-order terms.

↳ actually logic programming (first-order disjunctive clauses)

Stellar Resolution (background)

Jean-Yves Girard (2013)

Flexible tiles on first-order terms.

↳ actually logic programming (first-order disjunctive clauses)

But first, some elementary definitions :

First-order terms. $t, u ::= x \mid f(t_1, \ldots, t_n)$

Stellar Resolution (background)

Jean-Yves Girard (2013)

Flexible tiles on first-order terms.

↳ actually logic programming (first-order disjunctive clauses)

But first, some elementary definitions :

First-order terms. $t, u ::= x \mid f(t_1, \ldots, t_n)$

Unification. $t_1 \doteq t_2$: can we find $\theta : Vars \longmapsto Terms$ such that $\theta t_1 = \theta t_2$?

Stellar Resolution (background)

Jean-Yves Girard (2013)

Flexible tiles on first-order terms.

↳ actually logic programming (first-order disjunctive clauses)

But first, some elementary definitions :

First-order terms. $t, u ::= x \mid f(t_1, \ldots, t_n)$

Unification. $t_1 \doteq t_2$: can we find $\theta : \mathit{Vars} \mapsto \mathit{Terms}$ such that $\theta t_1 = \theta t_2$?

Matching. up-to-renaming $\alpha t_1 \doteq t_2$

Stellar Resolution (background)

Jean-Yves Girard (2013)

Flexible tiles on first-order terms.

↳ actually logic programming (first-order disjunctive clauses)

But first, some elementary definitions :

First-order terms. $t, u ::= x \mid f(t_1, \dots, t_n)$

Unification. $t_1 \doteq t_2$: can we find $\theta : \mathit{Vars} \mapsto \mathit{Terms}$ such that $\theta t_1 = \theta t_2$?

Matching. up-to-renaming $\alpha t_1 \doteq t_2$

↳ for $x \doteq f(x) \simeq_\alpha y \doteq f(x)$ we have $\theta = y \mapsto f(x)$

Stellar Resolution (static part)

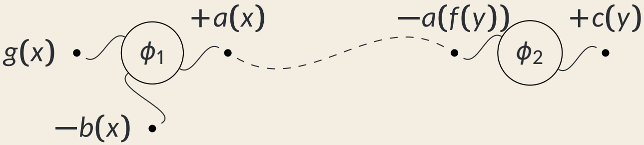
Stars and constellations

Borders are polarised first-order term with a head symbol called its colour.

Stellar Resolution (static part)

Stars and constellations

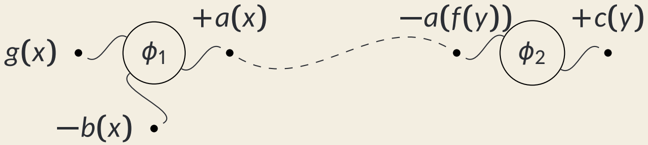
Borders are polarised first-order term with a head symbol called its colour.



Stellar Resolution (static part)

Stars and constellations

Borders are polarised first-order term with a head symbol called its colour.

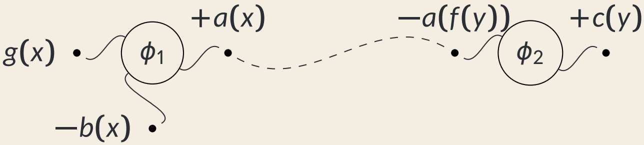


t and u are matchable with unifier $\theta = \{x \mapsto f(y)\}$.

Stellar Resolution (static part)

Stars and constellations

Borders are polarised first-order term with a head symbol called its colour.

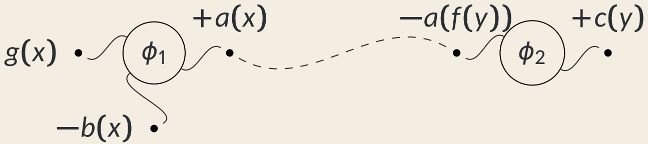


t and u are matchable with unifier $\theta = \{x \mapsto f(y)\}$.
Variables are local.

Stellar Resolution (static part)

Stars and constellations

Borders are polarised first-order term with a head symbol called its colour.



t and u are matchable with unifier $\theta = \{x \mapsto f(y)\}$.
Variables are local.

Tiles	Resolution	Stellar Resolution
	Atom $A = A(t), \neg A(t)$	Ray $r = +a(t), -a(t), t$
Tile	Clause $C = A_1 \vee \ldots \vee A_n$	Star $\phi = [r_1, \ldots, r_n]$
Tile set	Program $P = C_1 \wedge \ldots \wedge C_m$	Constellation $\Phi = \phi_1 + \ldots + \phi_m$
Tiling	Inference tree	Diagram

Stellar Resolution (dynamic part / fusion)

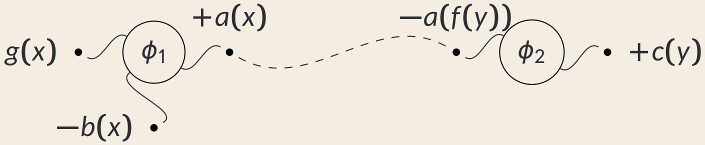
Jean-Yves Girard (2013)

Reducing diagrams by fusion of stars pairwise.

Stellar Resolution (dynamic part / fusion)

Jean-Yves Girard (2013)

Reducing diagrams by fusion of stars pairwise.

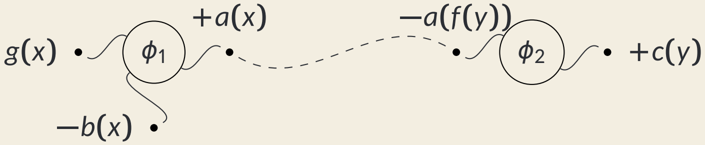


1. t and u are matchable with unifier $\theta = \{x \mapsto f(y)\}$.

Stellar Resolution (dynamic part / fusion)

Jean-Yves Girard (2013)

Reducing diagrams by fusion of stars pairwise.

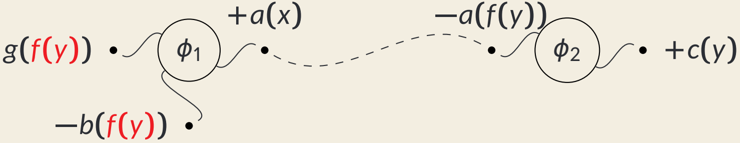


1. t and u are matchable with unifier $\theta = \{x \mapsto f(y)\}$.
2. propagation of θ.

Stellar Resolution (dynamic part / fusion)

Jean-Yves Girard (2013)

Reducing diagrams by fusion of stars pairwise.

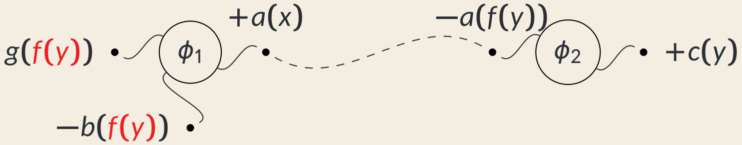


1. t and u are matchable with unifier $\theta = \{x \mapsto f(y)\}$.
2. propagation of θ.

Stellar Resolution (dynamic part / fusion)

Jean-Yves Girard (2013)

Reducing diagrams by fusion of stars pairwise.

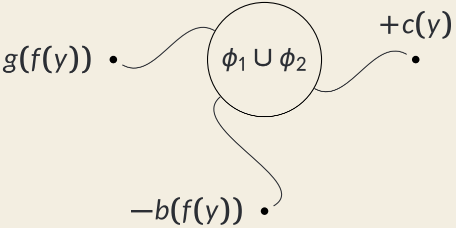


1. t and u are matchable with unifier $\theta = \{x \mapsto f(y)\}$.
2. propagation of θ.
3. destruction of connected rays + merging of stars.

Stellar Resolution (dynamic part / fusion)

Jean-Yves Girard (2013)

Reducing diagrams by fusion of stars pairwise.



1. t and u are matchable with unifier $\theta = \{x \mapsto f(y)\}$.
2. propagation of θ.
3. destruction of connected rays + merging of stars.

Stellar Resolution (dynamic part / execution)

Jean-Yves Girard (2013)

Fusion. diagram/tiling $\longmapsto$ star (non-empty)

Stellar Resolution (dynamic part / execution)

Jean-Yves Girard (2013)

Fusion. diagram/tiling $\longmapsto$ star (non-empty)

Execution. from a constellation (tile set) Φ :

ϕ_1

ϕ_2

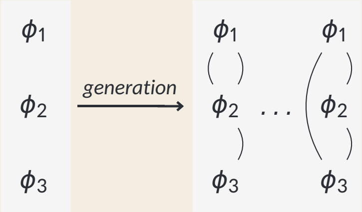
ϕ_3

Stellar Resolution (dynamic part / execution)

Jean-Yves Girard (2013)

Fusion. diagram/tiling $\mapsto$ star (non-empty)

Execution. from a constellation (tile set) Φ :

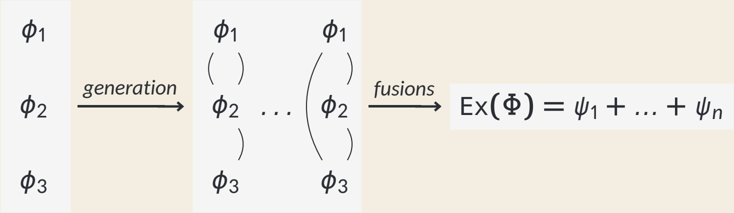


Stellar Resolution (dynamic part / execution)

Jean-Yves Girard (2013)

Fusion. diagram/tiling $\mapsto$ star (non-empty)

Execution. from a constellation (tile set) Φ :

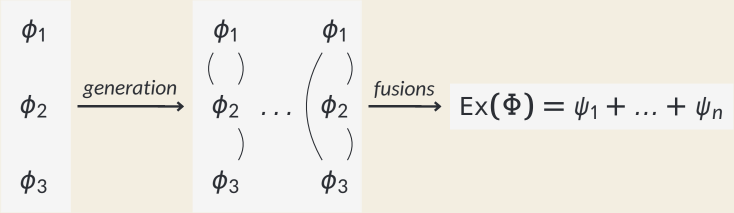


Stellar Resolution (dynamic part / execution)

Jean-Yves Girard (2013)

Fusion. diagram/tiling $\mapsto$ star (non-empty)

Execution. from a constellation (tile set) Φ :



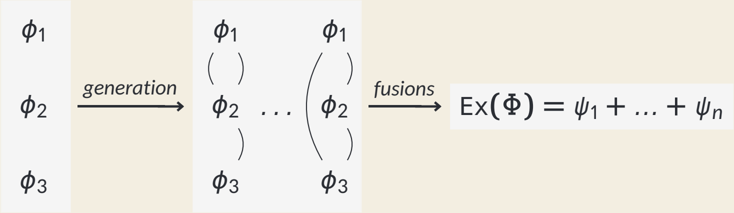
- We want the diagrams to be saturated (impossible to extend).

Stellar Resolution (dynamic part / execution)

Jean-Yves Girard (2013)

Fusion. diagram/tiling $\mapsto$ star (non-empty)

Execution. from a constellation (tile set) Φ :



- We want the diagrams to be saturated (impossible to extend).
- We also want them to be correct (no unification error).

Stellar Resolution

Few examples

Unary addition by **logic programming** :

$$[+add(0,y,y)] + [-add(x,y,z), +add(s(x),y,s(z))] + [-add(s^n(0),s^m(0),r),r]$$

Stellar Resolution

Few examples

Unary addition by **logic programming** :

$$[+add(0,y,y)] + [-add(x,y,z), +add(s(x),y,s(z))] + [-add(s^n(0),s^m(0),r),r]$$

$-add(0,y,y);$ — $+add(x,y,z);\ \ -add(s(x),y,s(z));$

$+add(x,y,z);\ \ -add(s(x),y,s(z));$ — $+add(s^2(0),s^2(0),r);\ \ r;$

Stellar Resolution

Few examples

Unary addition by **logic programming** :

$$[+add(0,y,y)] + [-add(x,y,z), +add(s(x),y,s(z))] + [-add(s^n(0), s^m(0), r), r]$$

$-add(0,y,y);$ — $+add(x,y,z);\ \ -add(s^2(x), y, s^2(z));$

$+add(s^2(0), s^2(0), r);\ \ r;$

Stellar Resolution

Few examples

Unary addition by **logic programming** :

$$[+add(0,y,y)] + [-add(x,y,z), +add(s(x),y,s(z))] + [-add(s^n(0),s^m(0),r),r]$$

$$\boxed{-add(s^2(0),y,s^2(y));} \;\text{———}\; \boxed{+add(s^2(0),s^2(0),r);\quad r;}$$

Stellar Resolution

Few examples

Unary addition by **logic programming** :

$$[+add(0, y, y)] + [-add(x, y, z), +add(s(x), y, s(z))] + [-add(s^n(0), s^m(0), r), r]$$

$s^4(0);$

Stellar Resolution

Few examples

Unary addition by **logic programming** :

$$[+add(0,y,y)] + [-add(x,y,z), +add(s(x),y,s(z))] + [-add(s^n(0), s^m(0), r), r]$$

$s^4(0)$;

All other diagrams fail, hence $\mathrm{Ex}(\Phi) = [s^4(0)]$.

Stellar Resolution

Few examples

Boolean circuits as hypergraph+dynamics : $X \vee \neg X$

Stellar Resolution

Few examples

Boolean circuits as hypergraph+dynamics : $X \vee \neg X$

$$\frac{-val(x), X(x)}{+c_1(x)} \;;$$

Stellar Resolution

Few examples

Boolean circuits as hypergraph+dynamics : $X \vee \neg X$

$$\frac{-val(x), X(x)}{+c_1(x)} \; ;$$

$$\frac{-c_1(x)}{+c_2(x), +c_3(x)} \; ;$$

Stellar Resolution

Few examples

Boolean circuits as hypergraph+dynamics : $X \vee \neg X$

$$\frac{-val(x), X(x)}{+c_1(x)} \; ;$$

$$\frac{-c_1(x)}{+c_2(x), +c_3(x)} \; ; \qquad \frac{-c_3(x), -not(x,r)}{+c_4(r)} \; ;$$

Stellar Resolution

Few examples

Boolean circuits as hypergraph+dynamics : $X \vee \neg X$

$$\frac{-val(x), X(x)}{+c_1(x)};$$

$$\frac{-c_1(x)}{+c_2(x), +c_3(x)}; \qquad \frac{-c_3(x), -not(x,r)}{+c_4(r)};$$

$$\frac{-c_2(x) \quad -c_3(y) \quad -or(x,y,r)}{+c_5(r)};$$

Stellar Resolution

Few examples

Boolean circuits as hypergraph+dynamics : $X \vee \neg X$

$$\frac{-val(x), X(x)}{+c_1(x)} \;;$$

$$\frac{-c_1(x)}{+c_2(x), +c_3(x)} \;; \qquad \frac{-c_3(x), -not(x,r)}{+c_4(r)} \;;$$

$$\frac{-c_2(x) \qquad -c_3(y) \qquad -or(x,y,r)}{+c_5(r)} \;;$$

Stellar Resolution

Few examples

Boolean circuits as hypergraph+dynamics : $X \vee \neg X$

$$\frac{-val(x), X(x)}{+c_1(x)} ;$$

$$\frac{-c_1(x)}{+c_2(x), +c_3(x)} ;$$

$$\frac{-c_3(x), -not(x,r)}{+c_4(r)} ;$$

$$\frac{-c_2(x) \quad -c_3(y) \quad -or(x,y,r)}{+c_5(r)} ;$$

$$\frac{-c_5(r)}{R(r)} ;$$

Stellar Resolution

Few examples

Boolean circuits as hypergraph+dynamics : $X \vee \neg X$

$$\frac{-val(x), X(x)}{+c_1(x)} ;$$

$$\frac{-c_1(x)}{+c_2(x), +c_3(x)} ; \qquad \frac{-c_3(x), -not(x,r)}{+c_4(r)} ;$$

$$\frac{-c_2(x) \qquad -c_3(y) \qquad -or(x,y,r)}{+c_5(r)} ;$$

$$\frac{-c_5(r)}{R(r)} ;$$

Stellar Resolution

Few examples

Boolean circuits as hypergraph+dynamics : $X \vee \neg X$

$$\frac{-val(x), X(x)}{+c_1(x)} \;;$$

$$\frac{-c_1(x)}{+c_2(x), +c_3(x)} \;; \qquad \frac{-c_3(x), -not(x,r)}{+c_4(r)} \;;$$

$$\frac{-c_2(x) \quad -c_3(y) \quad -or(x,y,r)}{+c_5(r)} \;;$$

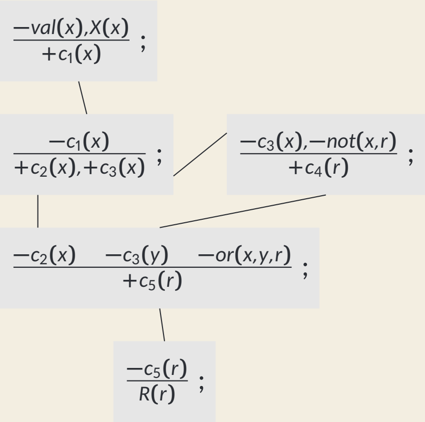
$$\frac{-c_5(r)}{R(r)} \;;$$

$$[+val(0)] + [+val(1)] + [+not(1,0)] + [+not(0,1)] + [+or(0,0,0)] + [+or(0,1,1)] + [+or(1,0,1)] + [+or(1,1,1)]$$

Stellar Resolution

Few examples

Boolean circuits as hypergraph+dynamics : $X \vee \neg X$



$$[+val(0)] + [+val(1)] + [+not(1,0)] + [+not(0,1)] + [+or(0,0,0)] + [+or(0,1,1)] + [+or(1,0,1)] + [+or(1,1,1)]$$

$$\mathrm{Ex}(\Phi) = [X(0), R(1)] + [X(1), R(1)]$$

Stellar Resolution

Few examples

Boolean circuits as hypergraph+dynamics : $X \vee \neg X$

$$\frac{-val(x), X(x)}{+c_1(x)};$$

$$\frac{-c_1(x)}{+c_2(x), +c_3(x)}; \qquad \frac{-c_3(x), -not(x,r)}{+c_4(r)};$$

$$\frac{-c_2(x) \quad -c_3(y) \quad -or(x,y,r)}{+c_5(r)};$$

$$\frac{-c_5(r)}{R(r)};$$

$$[+val(0)] + [+val(1)] + [+not(1,0)] + [+not(0,1)] + [+or(0,0,0)] + [+or(0,1,1)] + [+or(1,0,1)] + [+or(1,1,1)]$$

$$\mathrm{Ex}(\Phi) = [X(0), R(1)] + [X(1), R(1)]$$

Extensible to arithmetic circuits

Transcendental Syntax

Proofs as tilings

Transcendental Syntax

Motivations

Explain (linear) logic from its computational behaviour.

Transcendental Syntax

Motivations

Explain (linear) logic from its computational behaviour. By finite means !

Transcendental Syntax

Motivations

Explain (linear) logic from its computational behaviour. By finite means !

Answers/Analytic adequate computational space : stellar resolution.

Transcendental Syntax

Motivations

Explain (linear) logic from its computational behaviour. By finite means !

Answers/Analytic adequate computational space : stellar resolution.

↳ independant/local interaction

Transcendental Syntax

Motivations

Explain (linear) logic from its computational behaviour. By finite means !

Answers/Analytic adequate computational space : stellar resolution.

↳ independant/local interaction

↳ "large enough"

Transcendental Syntax

Motivations

Explain (linear) logic from its computational behaviour. By finite means !

Answers/Analytic adequate computational space : stellar resolution.

↳ independant/local interaction

↳ "large enough"

Questions/Synthetic emergence of logical space : correctness, formulas, use.

Transcendental Syntax

Motivations

Explain (linear) logic from its computational behaviour. By finite means !

Answers/Analytic adequate computational space : stellar resolution.

↳ independant/local interaction

↳ "large enough"

Questions/Synthetic emergence of logical space : correctness, formulas, use.

↳ what is a "good" interaction ? (subjective)

Transcendental Syntax

Motivations

Explain (linear) logic from its computational behaviour. By finite means !

Answers/Analytic adequate computational space : stellar resolution.

↳ independant/local interaction

↳ "large enough"

Questions/Synthetic emergence of logical space : correctness, formulas, use.

↳ what is a "good" interaction ? (subjective)

"*This can only be a reconstruction, which means that we roughly know what we are aiming at.*" (Transcendental Syntax I).

Transcendental Syntax

MLL proof-structures

An alternative representation of proofs as hypergraphs :

Transcendental Syntax

MLL proof-structures

An alternative representation of proofs as hypergraphs :

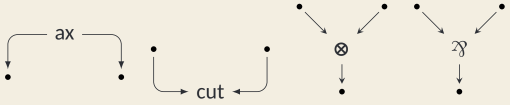
Formulas/Vertices. $A, B := X \mid A \otimes B \mid A \parr B$

Transcendental Syntax

MLL proof-structures

An alternative representation of proofs as hypergraphs :

Formulas/Vertices. $A, B := X \mid A \otimes B \mid A ⅋ B$



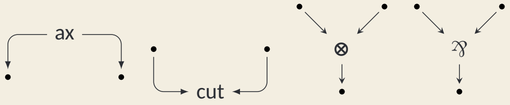
Rules/Hyperedges. Axiom Cut Tensor Par

Transcendental Syntax

MLL proof-structures

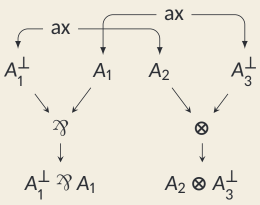
An alternative representation of proofs as hypergraphs :

Formulas/Vertices. $A, B := X \mid A \otimes B \mid A \parr B$



Axiom Cut Tensor Par

Rules/Hyperedges.



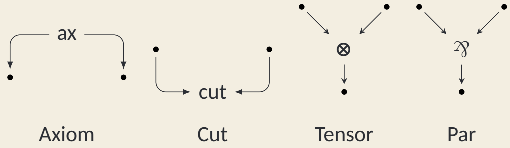
Transcendental Syntax

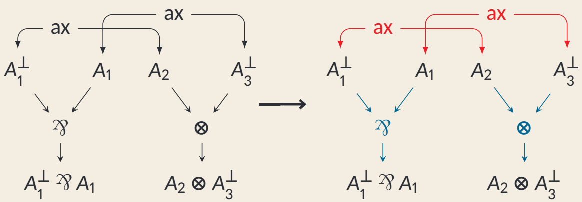
MLL proof-structures

An alternative representation of proofs as hypergraphs :

Formulas/Vertices. $A, B := X \mid A \otimes B \mid A \parr B$

Rules/Hyperedges.





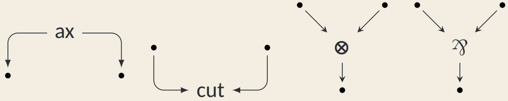
Transcendental Syntax

MLL proof-structures

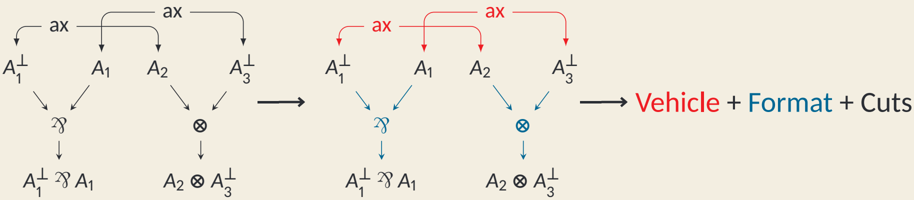
An alternative representation of proofs as hypergraphs :

Formulas/Vertices. $A, B := X \mid A \otimes B \mid A \parr B$

Rules/Hyperedges.



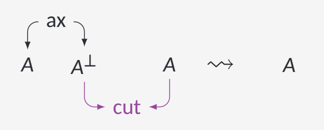
Axiom Cut Tensor Par

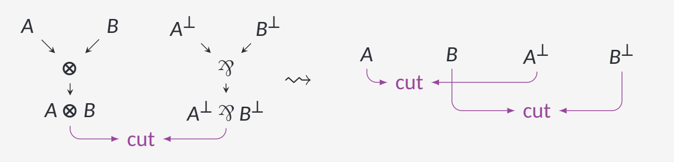


Transcendental Syntax

The computational content of proof-structures

Cut-elimination procedure :

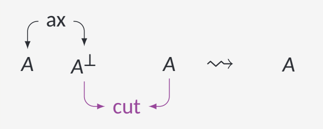


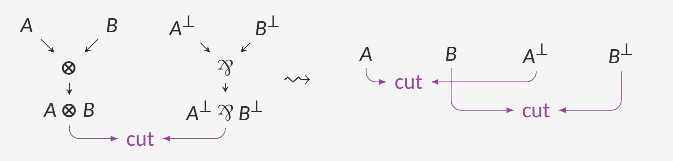


Transcendental Syntax

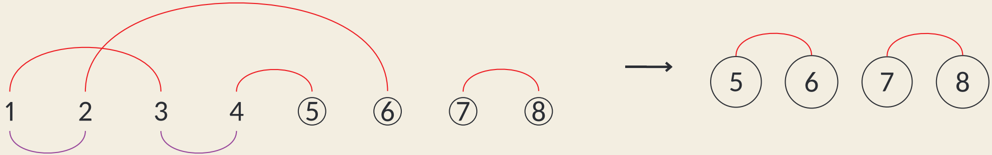
The computational content of proof-structures

Cut-elimination procedure :





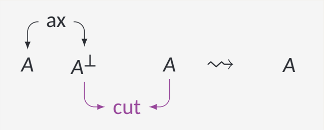
Geometry of Interaction :

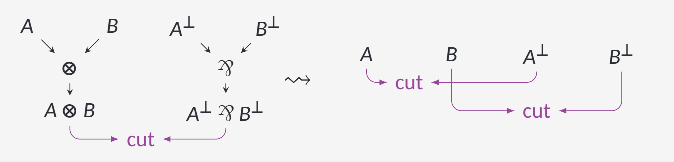


Transcendental Syntax

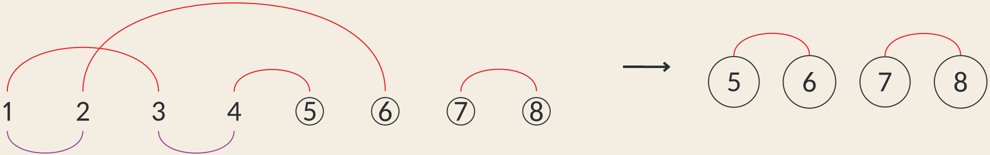
The computational content of proof-structures

Cut-elimination procedure :





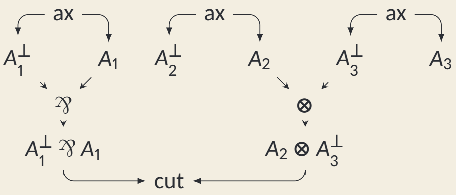
Geometry of Interaction :

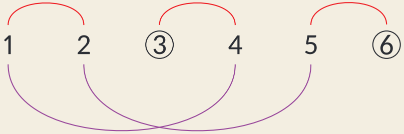


Basically a computation of maximal paths in a graph.

Transcendental Syntax

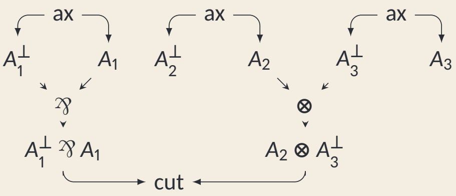
Simulation of cut-elimination

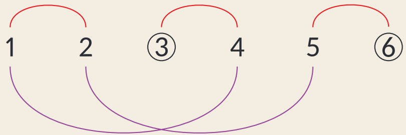




Transcendental Syntax

Simulation of cut-elimination





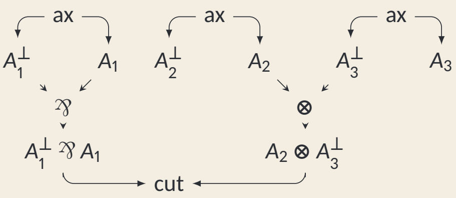
$+c.p_{A_1^\perp ⅋ A_1}(\mathtt{l}x);\ +c.p_{A_1^\perp ⅋ A_1}(\mathtt{r}x);$

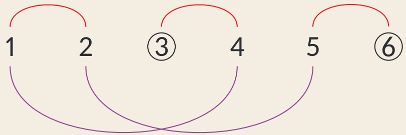
$+c.p_{A_2^\perp}(x);\ +c.p_{A_2 \otimes A_3^\perp}(\mathtt{l}x);$

$+c.p_{A_2 \otimes A_3^\perp}(\mathtt{r}x);\ +c.p_{A_3}(x);$

Transcendental Syntax

Simulation of cut-elimination





$+c.p_{A_1^\perp \parr A_1}(\mathtt{l}x);\quad +c.p_{A_1^\perp \parr A_1}(\mathtt{r}x)\ ;$

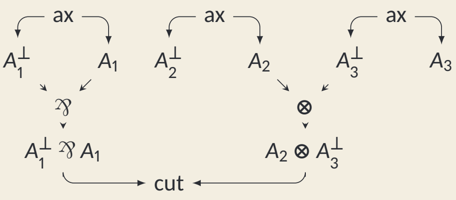
$+c.p_{A_2^\perp}(x);\quad +c.p_{A_2 \otimes A_3^\perp}(\mathtt{l}x)\ ;$

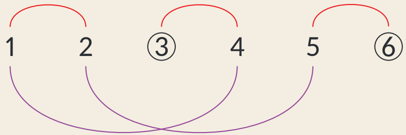
$+c.p_{A_2 \otimes A_3^\perp}(\mathtt{r}x);\quad +c.p_{A_3}(x)\ ;$

$-c.p_{A_1^\perp \parr A_1}(x);\quad -c.p_{A_2 \otimes A_3^\perp}(x)\ ;$

Transcendental Syntax

Simulation of cut-elimination



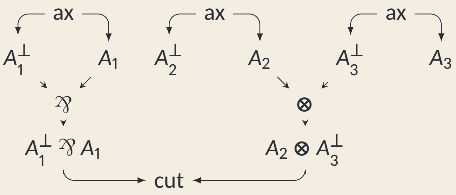


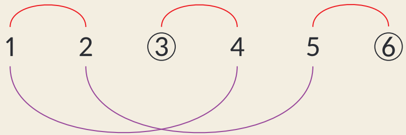
$+c.p_{A_1^\perp \parr A_1}(\mathtt{l}x);\ +c.p_{A_1^\perp \parr A_1}(\mathtt{r}x)\ ;$ $\quad$ $+c.p_{A_2^\perp}(x);\ +c.p_{A_2 \otimes A_3^\perp}(\mathtt{l}x)\ ;$ $\quad$ $+c.p_{A_2 \otimes A_3^\perp}(\mathtt{r}x);\ +c.p_{A_3}(x)\ ;$

$-c.p_{A_1^\perp \parr A_1}(\mathtt{l}x);\ -c.p_{A_2 \otimes A_3^\perp}(\mathtt{l}x)\ ;$ $\quad$ $-c.p_{A_1^\perp \parr A_1}(\mathtt{r}x);\ -c.p_{A_2 \otimes A_3^\perp}(\mathtt{r}x)\ ;$

Transcendental Syntax

Simulation of cut-elimination





$+c.p_{A_2^\perp}(x)$; $+c.p_{A_3}(x)$;

Transcendental Syntax

The logical content of proof-structures

Only some proof-structures are "logically correct".

Transcendental Syntax

The logical content of proof-structures

Only some proof-structures are "logically correct".

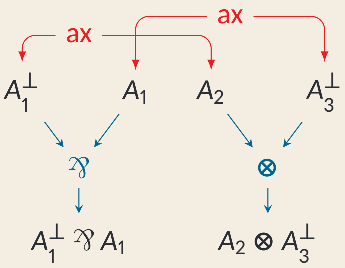
Danos-Regnier **correctness criterion** : testing the vehicle against several Tests $\in$ Format.

Transcendental Syntax

The logical content of proof-structures

Only some proof-structures are "logically correct".

Danos-Regnier **correctness criterion** : testing the vehicle against several Tests ∈ Format.

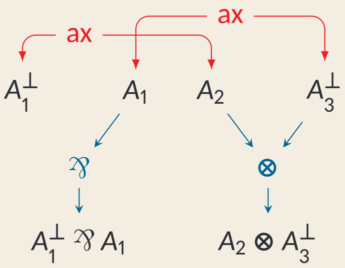


Transcendental Syntax

The logical content of proof-structures

Only some proof-structures are "logically correct".

Danos-Regnier **correctness criterion** : testing the vehicle against several Tests ∈ Format.

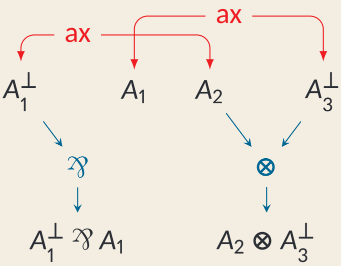


Transcendental Syntax

The logical content of proof-structures

Only some proof-structures are "logically correct".

Danos-Regnier **correctness criterion** : testing the vehicle against several Tests $\in$ Format.



Transcendental Syntax

Simulation of correctness

Stars $\equiv$ Oriented hyperedges

$+t.p_{A\otimes B}(\mathtt{l}x);\ \ +t.p_{A^\perp \parr B^\perp}(\mathtt{l}x)\ ;$

$+t.p_{A\otimes B}(\mathtt{r}x);\ \ +t.p_{A^\perp \parr B^\perp}(\mathtt{r}x)\ ;$

Transcendental Syntax

Simulation of correctness

Stars ≡ Oriented hyperedges

$+t.p_{A\otimes B}(lx);\ \ +t.p_{A^\perp ⅋ B^\perp}(lx)\ ;$

$+t.p_{A\otimes B}(rx);\ \ +t.p_{A^\perp ⅋ B^\perp}(rx)\ ;$

$[\frac{-t.p_{A\otimes B}(lx)}{+c.q_A(x)}]\ ;$

$[\frac{-t.p_{A\otimes B}(rx)}{+c.q_{A^\perp}(x)}]\ ;$

$[\frac{-t.p_{A^\perp ⅋ B^\perp}(lx)}{+c.q_B(x)}]\ ;$

$[\frac{-t.p_{A^\perp ⅋ B^\perp}(rx)}{+c.q_{B^\perp}(x)}]\ ;$

$[\frac{-c.q_A(x)\quad -c.q_B(x)}{+c.q_{A\otimes B}(x)}]\ ;$

$\frac{-c.q_{A^\perp}(x)}{}\ ;$

$\frac{-c.q_{B^\perp}(x)}{+c.q_{A^\perp ⅋ B^\perp}(x)}\ ;$

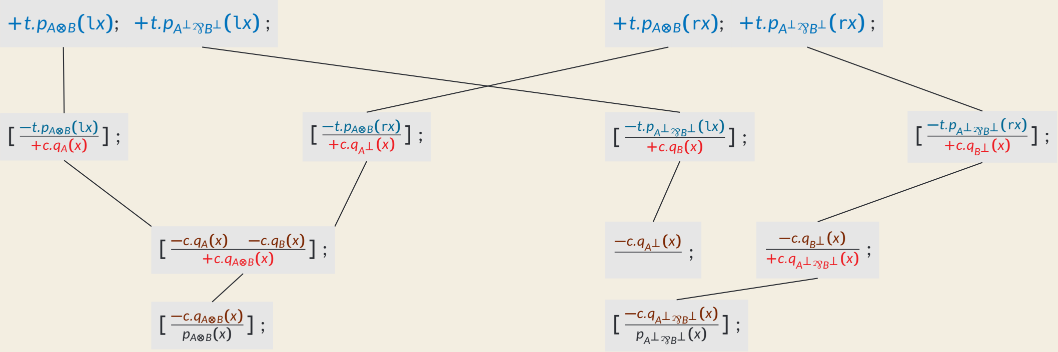
$[\frac{-c.q_{A\otimes B}(x)}{p_{A\otimes B}(x)}]\ ;$

$[\frac{-c.q_{A^\perp ⅋ B^\perp}(x)}{p_{A^\perp ⅋ B^\perp}(x)}]\ ;$

Transcendental Syntax

Simulation of correctness

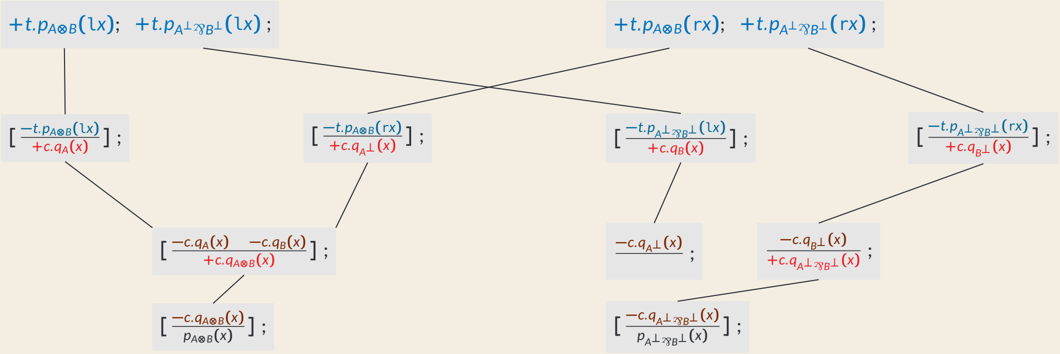
Stars $\equiv$ Oriented hyperedges



Transcendental Syntax

Simulation of correctness

Stars ≡ Oriented hyperedges



Property of the tiling : **logically correct** iff for all test, the normal form is a single star.

Transcendental Syntax

Typing and formulas

Use of techniques from "linear" realisability.

Transcendental Syntax

Typing and formulas

Use of techniques from "linear" realisability.

Pre-types description of a behaviour $\mathbf{A} = \{\Phi_i\}_{i \in I}$.

Transcendental Syntax

Typing and formulas

Use of techniques from "linear" realisability.

Pre-types description of a behaviour $\mathbf{A} = \{\Phi_i\}_{i \in I}$.

Orthogonality Choose a definition of "good interaction" $\Phi \perp \Phi'$.

Transcendental Syntax

Typing and formulas

Use of techniques from "linear" realisability.

Pre-types description of a behaviour $\mathbf{A} = \{\Phi_i\}_{i \in I}$.

Orthogonality Choose a definition of "good interaction" $\Phi \perp \Phi'$.

Dual pre-type $\mathbf{A}^{\perp}$ set of "good partners" $\{\Phi \mid \forall \Phi_A \in \mathbf{A}, \Phi \perp \Phi_A\}$.

Transcendental Syntax

Typing and formulas

Use of techniques from "linear" realisability.

Pre-types description of a behaviour $\mathbf{A} = \{\Phi_i\}_{i \in I}$.

Orthogonality Choose a definition of "good interaction" $\Phi \perp \Phi'$.

Dual pre-type $\mathbf{A}^{\perp}$ set of "good partners" $\{\Phi \mid \forall \Phi_A \in \mathbf{A}, \Phi \perp \Phi_A\}$.

Types $\mathbf{A} = \mathbf{A}^{\perp\perp}$ closed interaction.

Transcendental Syntax

Typing and formulas

Use of techniques from "linear" realisability.

Pre-types description of a behaviour $\mathbf{A} = \{\Phi_i\}_{i \in I}$.

Orthogonality Choose a definition of "good interaction" $\Phi \perp \Phi'$.

Dual pre-type $\mathbf{A}^\perp$ set of "good partners" $\{\Phi \mid \forall \Phi_A \in \mathbf{A}, \Phi \perp \Phi_A\}$.

Types $\mathbf{A} = \mathbf{A}^{\perp\perp}$ closed interaction.

Tensor $\mathbf{A} \otimes \mathbf{B} := \{\Phi_A \uplus \Phi_B \mid \Phi_A \in \mathbf{A}, \Phi_B \in \mathbf{B}\}^{\perp\perp}$.

Transcendental Syntax

Typing and formulas

Use of techniques from "linear" realisability.

Pre-types description of a behaviour $\mathbf{A} = \{\Phi_i\}_{i \in I}$.

Orthogonality Choose a definition of "good interaction" $\Phi \perp \Phi'$.

Dual pre-type $\mathbf{A}^\perp$ set of "good partners" $\{\Phi \mid \forall \Phi_A \in \mathbf{A}, \Phi \perp \Phi_A\}$.

Types $\mathbf{A} = \mathbf{A}^{\perp\perp}$ closed interaction.

Tensor $\mathbf{A} \otimes \mathbf{B} := \{\Phi_A \uplus \Phi_B \mid \Phi_A \in \mathbf{A}, \Phi_B \in \mathbf{B}\}^{\perp\perp}$.

Other constructions $\mathbf{A} \parr \mathbf{B} := (\mathbf{A}^\perp \otimes \mathbf{B}^\perp)^\perp$, $\quad \mathbf{A} \multimap \mathbf{B} := \mathbf{A}^\perp \parr \mathbf{B}$.

Transcendental Syntax

Typing and formulas

Use of techniques from "linear" realisability.

Pre-types description of a behaviour $\mathbf{A} = \{\Phi_i\}_{i \in I}$.

Orthogonality Choose a definition of "good interaction" $\Phi \perp \Phi'$.

Dual pre-type $\mathbf{A}^\perp$ set of "good partners" $\{\Phi \mid \forall \Phi_A \in \mathbf{A}, \Phi \perp \Phi_A\}$.

Types $\mathbf{A} = \mathbf{A}^{\perp\perp}$ closed interaction.

Tensor $\mathbf{A} \otimes \mathbf{B} := \{\Phi_A \uplus \Phi_B \mid \Phi_A \in \mathbf{A}, \Phi_B \in \mathbf{B}\}^{\perp\perp}$.

Other constructions $\mathbf{A} \parr \mathbf{B} := (\mathbf{A}^\perp \otimes \mathbf{B}^\perp)^\perp$, $\mathbf{A} \multimap \mathbf{B} := \mathbf{A}^\perp \parr \mathbf{B}$.

$\Phi_1 \perp \Phi_2 \iff |Ex(\Phi_1 \uplus \Phi_2)| = 1$: captures MLL formulas.

Transcendental Syntax

Conclusion

- Logic programs and geometric tiling meet

Transcendental Syntax

Conclusion

- Logic programs and geometric tiling meet
- Can be extended to full linear logic and second order

Transcendental Syntax

Conclusion

- Logic programs and geometric tiling meet
- Can be extended to full linear logic and second order

  ↳ exponentials : work in progress

Transcendental Syntax

Conclusion

- Logic programs and geometric tiling meet
- Can be extended to full linear logic and second order
  ↳ exponentials : work in progress
- Reconstruction of first-order logic possible

Transcendental Syntax

Conclusion

- Logic programs and geometric tiling meet
- Can be extended to full linear logic and second order
  ↳ exponentials : work in progress
- Reconstruction of first-order logic possible
  ↳ terms/individuals as multiplicative propositions

Transcendental Syntax

Conclusion

- Logic programs and geometric tiling meet
- Can be extended to full linear logic and second order

  ↳ exponentials : work in progress
- Reconstruction of first-order logic possible

  ↳ terms/individuals as multiplicative propositions

  ↳ equality as linear equivalence (not predicate !)

Transcendental Syntax

Conclusion

- Logic programs and geometric tiling meet
- Can be extended to full linear logic and second order
  ↳ exponentials : work in progress
- Reconstruction of first-order logic possible
  ↳ terms/individuals as multiplicative propositions
  ↳ equality as linear equivalence (not predicate !)
- Logic programs and functional programs, unified ?

Future and related works

(actually a call for help)

Hypergraphings

Hypergraphs with dynamics

Computation with hypergraphs :

Model	**Vertices**	**Hyperedges**
Boolean circuits	addresses	gates
Proof-nets	addresses	rules
Constellations	terms	stars
Automata	states	transitions

Hypergraphings

Hypergraphs with dynamics

Computation with hypergraphs :

Model	**Vertices**	**Hyperedges**
Boolean circuits	addresses	gates
Proof-nets	addresses	rules
Constellations	terms	stars
Automata	states	transitions

↳ interaction of hypergraphs + execution.

Hypergraphings

Hypergraphs with dynamics

Computation with hypergraphs :

Model	**Vertices**	**Hyperedges**
Boolean circuits	addresses	gates
Proof-nets	addresses	rules
Constellations	terms	stars
Automata	states	transitions

↳ interaction of hypergraphs + execution.

↳ generalisation of Seiller’s *Graphings*.

Hypergraphings

Hypergraphs with dynamics

Computation with hypergraphs :

Model	**Vertices**	**Hyperedges**
Boolean circuits	addresses	gates
Proof-nets	addresses	rules
Constellations	terms	stars
Automata	states	transitions

↳ interaction of hypergraphs + execution.

↳ generalisation of Seiller's *Graphings*.

↳ categorical framework ? Operads, string diagrams, hypergraph categories, frobenius algebras, ...

Stellar Resolution and Automata Theory

Dependency graph $\mathfrak{D}(\Phi)$: relations of matchability within a constellation Φ.

$$\begin{tikzcd} -add(0,y,y); \arrow[r, dash] & +add(x,y,z);\ \ -add(s(x),y,s(z)); \arrow[loop above, dash] \arrow[r, dash] & +add(s^n(0),s^m(0),r);\ \ r; \end{tikzcd}$$

Stellar Resolution and Automata Theory

Dependency graph $\mathfrak{D}(\Phi)$: relations of matchability within a constellation Φ.

$-add(0, y, y);$ — $+add(x, y, z);\quad -add(s(x), y, s(z));$ — $+add(s^n(0), s^m(0), r);\quad r;$

Diagram (formally) : graph homomorphism $\delta : G \to \mathfrak{D}(\Phi)$.

$-add(0, y, y);$ — $+add(x, y, z);\quad -add(s(x), y, s(z));$

$+add(x, y, z);\quad -add(s(x), y, s(z));$ — $+add(s^2(0), s^2(0), r);\quad r;$

Run in finite automata : path $\longmapsto$ state graph

Stellar Resolution and Automata Theory

Dependency graph $\mathfrak{D}(\Phi)$: relations of matchability within a constellation Φ.

$$-add(0, y, y); \quad — \quad +add(x, y, z); \;\; -add(s(x), y, s(z)); \quad — \quad +add(s^n(0), s^m(0), r); \;\; r;$$

Diagram (formally) : graph homomorphism $\delta : G \to \mathfrak{D}(\Phi)$.

$$-add(0, y, y); \quad — \quad +add(x, y, z); \;\; -add(s(x), y, s(z));$$

$$+add(x, y, z); \;\; -add(s(x), y, s(z)); \quad — \quad +add(s^2(0), s^2(0), r); \;\; r;$$

Run in finite automata : path $\longmapsto$ state graph

↳ we are interested in reaching finale state.

Stellar Resolution and Automata Theory

Dependency graph $\mathfrak{D}(\Phi)$: relations of matchability within a constellation Φ.

$-add(0, y, y);$ — $+add(x, y, z);\ \ -add(s(x), y, s(z));$ — $+add(s^n(0), s^m(0), r);\ \ r;$

Diagram (formally) : graph homomorphism $\delta : G \to \mathfrak{D}(\Phi)$.

$-add(0, y, y);$ — $+add(x, y, z);\ \ -add(s(x), y, s(z));$

$+add(x, y, z);\ \ -add(s(x), y, s(z));$ — $+add(s^2(0), s^2(0), r);\ \ r;$

Run in finite automata : path $\longmapsto$ state graph

↳ we are interested in reaching finale state.

↳ automaton for stellar execution ?