Séminaire Prothéo
Polygraphic programs

Guillaume Bonfante – Yves Guiraud

INRIA – LORIA

Nancy – April 2, 2007

Polygraphic programs

Motivations

Static analysis of programs

Static analysis of programs

Data: the code of a (first-order functional) program

```
type integer = Z | S of integer ;;
let rec add = function
     | Z, x -> x
     | S(x), y -> S(add(x,y)) ;;
let rec mult = function
     | x, Z -> Z
     | x, S(y) -> add(x,mult(x,y)) ;;
```

Static analysis of programs

Data: the code of a (first-order functional) program

```
type integer = Z | S of integer ;;
let rec add = function
     | Z, x -> x
     | S(x), y -> S(add(x,y)) ;;
let rec mult = function
     | x, Z -> Z
     | x, S(y) -> add(x,mult(x,y)) ;;
```

Questions: properties such as

- termination,
- computational complexity,

i.e. memory (spatial complexity) and time (temporal complexity) required.

Polynomial interpretations of terms

Polynomial interpretations of terms

Idea: TRSs interpretations can bound complexity. [Hofbauer, Lautemann; Cichon, Lescanne]

Polynomial interpretations of terms

Idea: TRSs interpretations can bound complexity. [Hofbauer, Lautemann; Cichon, Lescanne]

1 – Translate the program into a TRS (Σ, R):

$$\Sigma: \quad \text{constructors } (0,\ s), \quad \text{functions } (+,\ \times),$$

$$R: \quad 0 + x \to x, \qquad s(x) + y \to s(x+y), \qquad (x \times 0) \to 0, \qquad x \times s(y) \to x + (x \times y).$$

Polynomial interpretations of terms

Idea: TRSs interpretations can bound complexity. [Hofbauer, Lautemann; Cichon, Lescanne]

1 – Translate the program into a TRS (Σ, R):

$$\Sigma : \quad \text{constructors } (0,\ s), \quad \text{functions } (+,\ \times),$$

$$R : \quad 0 + x \to x, \qquad s(x) + y \to s(x+y), \qquad (x \times 0) \to 0, \qquad x \times s(y) \to x + (x \times y).$$

2 – Build a map $(\!|\cdot|\!) : \Sigma \to \mathbb{N}[X, Y, \ldots]$, such as:

$$(\!|0|\!) = 1, \qquad (\!|s|\!) = X + 1, \qquad (\!|+|\!) = 2X + Y + 1, \qquad (\!|\times|\!) = (X+1)(Y+1).$$

Polynomial interpretations of terms

Idea: TRSs interpretations can bound complexity. [Hofbauer, Lautemann; Cichon, Lescanne]

1 – Translate the program into a TRS (Σ, R):

$$\Sigma: \quad \text{constructors } (0,\ s), \quad \text{functions } (+,\ \times),$$

$$R: \quad 0 + x \to x, \qquad s(x) + y \to s(x+y), \qquad (x \times 0) \to 0, \qquad x \times s(y) \to x + (x \times y).$$

2 – Build a map $(\!|\cdot|\!) : \Sigma \to \mathbb{N}[X, Y, \ldots]$, such as:

$$(\!|0|\!) = 1, \qquad (\!|s|\!) = X + 1, \qquad (\!|+|\!) = 2X + Y + 1, \qquad (\!|\times|\!) = (X+1)(Y+1).$$

Then $(\!|u|\!) > (\!|v|\!)$ when $u \to v$, plus some other conditions: $(\!|\cdot|\!)$ is a *polynomial interpretation*.

Polynomial interpretations of terms

Idea: TRSs interpretations can bound complexity. [Hofbauer, Lautemann; Cichon, Lescanne]

1 – Translate the program into a TRS (Σ, R):

$$\Sigma: \quad \text{constructors } (0,\ s), \quad \text{functions } (+,\ \times),$$

$$R: \quad 0 + x \to x, \qquad s(x) + y \to s(x+y), \qquad (x \times 0) \to 0, \qquad x \times s(y) \to x + (x \times y).$$

2 – Build a map $(\!|\cdot|\!) : \Sigma \to \mathbb{N}[X, Y, \ldots]$, such as:

$$(\!|0|\!) = 1, \qquad (\!|s|\!) = X + 1, \qquad (\!|+|\!) = 2X + Y + 1, \qquad (\!|\times|\!) = (X+1)(Y+1).$$

Then $(\!|u|\!) > (\!|v|\!)$ when $u \to v$, plus some other conditions: $(\!|\cdot|\!)$ is a *polynomial interpretation*.

Results: [Bonfante, Cichon, Marion, Touzet]

- Space: if u is a value, $(\!|u|\!)$ bounds the size of u.
- Time: if f is a function, then $(\!|f(\vec{u})|\!)$ bounds the number of steps to reach the result.
- The values of $(\!|\cdot|\!)$ on constructors give the complexity class of the functions (PTIME here).

Limitations of polynomial interpretations

Limitations of polynomial interpretations

Mixed bounds: $(\!|\cdot|\!)$ bounds space *and* time.

Example: $+$ requires X steps, not $2X+Y+1$.

Consequences: mixed information, overestimated bounds.

Limitations of polynomial interpretations

Mixed bounds: $(\!|\cdot|\!)$ bounds space *and* time.

Example: $+$ requires X steps, not $2X+Y+1$.

Consequences: mixed information, overestimated bounds.

Programs out of range: polynomial interpretations yield simplification orders.

Example: integer division $s(x)/y \to s((x-y)/y)$ gives $s(x)/s(x) \lhd s((x-s(x))/s(x))$.

Consequence: cannot give bounds for such programs.

Limitations of polynomial interpretations

Mixed bounds: $(\!|\cdot|\!)$ bounds space *and* time.

Example: $+$ requires X steps, not $2X+Y+1$.

Consequences: mixed information, overestimated bounds.

Programs out of range: polynomial interpretations yield simplification orders.

Example: integer division $\quad s(x)/y \rightarrow s((x-y)/y) \quad$ gives $\quad s(x)/s(x) \lhd s((x-s(x))/s(x))$.

Consequence: cannot give bounds for such programs.

Algorithms out of range: TRSs can only compute *products* of functions with one output each.

Counter-example: computing *at the same time* $(x_1, x_3, \ldots)$ and $(x_2, x_4, \ldots)$ from $(x_1, x_2, x_3, \ldots)$.

Consequence: TRSs cannot describe faithfully "divide-and-conquer" algorithms.

Limitations of polynomial interpretations

Mixed bounds: $(\!|\cdot|\!)$ bounds space *and* time.

Example: $+$ requires X steps, not $2X+Y+1$.

Consequences: mixed information, overestimated bounds.

Programs out of range: polynomial interpretations yield simplification orders.

Example: integer division $\quad s(x)/y \to s((x-y)/y) \quad$ gives $\quad s(x)/s(x) \lhd s((x-s(x))/s(x))$.

Consequence: cannot give bounds for such programs.

Algorithms out of range: TRSs can only compute *products* of functions with one output each.

Counter-example: computing *at the same time* $(x_1, x_3, \ldots)$ and $(x_2, x_4, \ldots)$ from $(x_1, x_2, x_3, \ldots)$.

Consequence: TRSs cannot describe faithfully "divide-and-conquer" algorithms.

A solution: provide an alternative framework for programs.

Polygraphic programs

Polygraphic programs

Examples of polygraphic programs

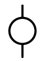
Examples of polygraphic programs

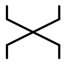
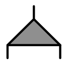
Addition and multiplication:

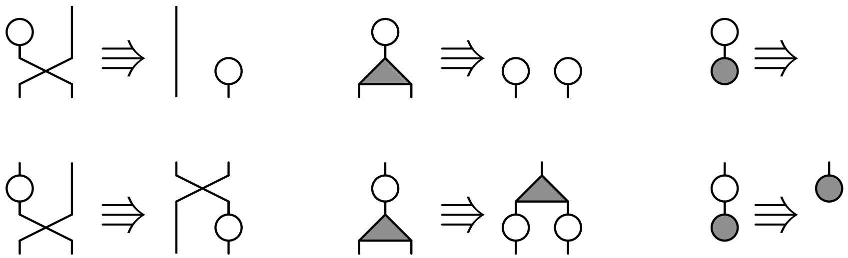
Computes addition and multiplication over natural numbers $\langle\ ,\ \rangle$.

Examples of polygraphic programs

Addition and multiplication:

Computes addition  and multiplication  over natural numbers ⟨, ⟩.

Structure operations permutation , duplication  and erasure  are explicit:

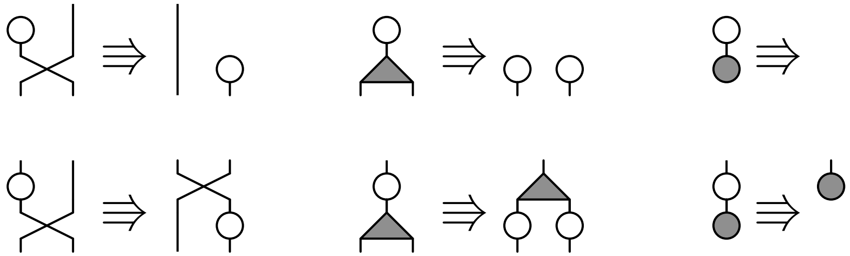


Examples of polygraphic programs

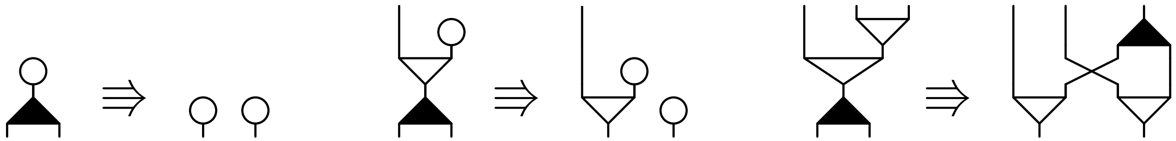
Addition and multiplication:

Computes addition and multiplication over natural numbers ⟨ , ⟩.

Structure operations permutation , duplication and erasure are explicit:



Functions with many outputs:



A list splitting function on lists ⟨ , ⟩: both sublists are computed at the same time.

Polygraphic programs

Polygraphic programs

A **polygraphic program** consists of:

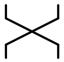
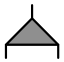
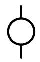
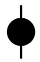
Polygraphic programs

A **polygraphic program** consists of:

- **1-cells** (wires) with 1 composition $\star_0$ to build 1-paths (parallel wires).

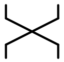
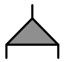
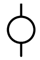
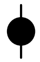
Polygraphic programs

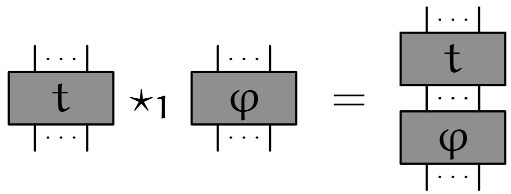
A **polygraphic program** consists of:

- **1-cells** (wires) with 1 composition $\star_0$ to build 1-paths (parallel wires).
- **2-cells** (gates) with 2 compositions $\star_0$ (horizontal) and $\star_1$ (vertical) to build 2-paths (circuits). Three kinds:
  - Structure: ,  and  with all possible 1-cells.
  - Constructors (with one output) such as , , , etc.
  - Functions (any possible shape) such as , , , etc.

Polygraphic programs

A **polygraphic program** consists of:

- **1-cells** (wires) with 1 composition $\star_0$ to build 1-paths (parallel wires).
- **2-cells** (gates) with 2 compositions $\star_0$ (horizontal) and $\star_1$ (vertical) to build 2-paths (circuits). Three kinds:
  - Structure: ,  and  with all possible 1-cells.
  - Constructors (with one output) such as , , , etc.
  - Functions (any possible shape) such as , , , etc.
- **3-cells** (rules) with 3 compositions $\star_0$, $\star_1$ (parallel) and $\star_2$ (sequential) to build 3-paths (computations). Two kinds:
  - Structure: compute permutations, duplications and erasures on constructors.
  - Computation with source like this (φ function and t term):



Computational power of polygraphic programs

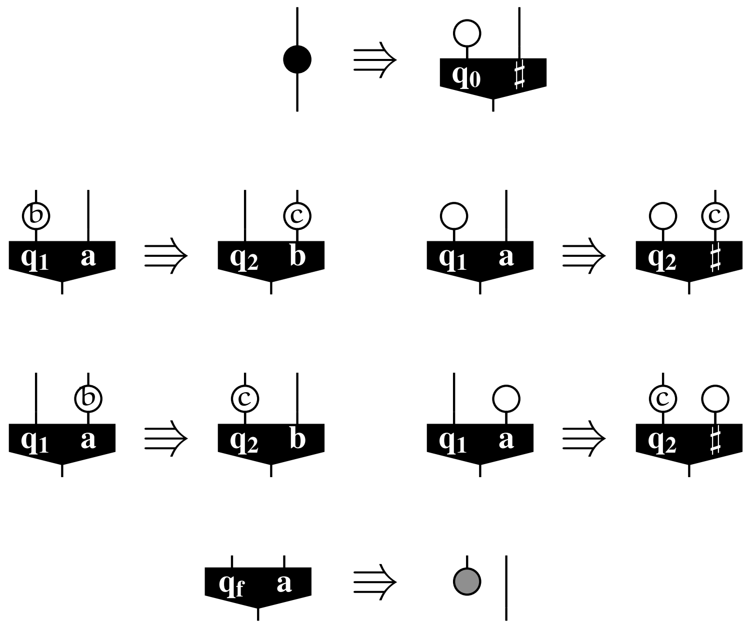
Computational power of polygraphic programs

Theorem: polygraphic programs form a Turing-complete model of computation.

Computational power of polygraphic programs

Theorem: polygraphic programs form a Turing-complete model of computation.

Proof: build a polygraphic Turing machine:



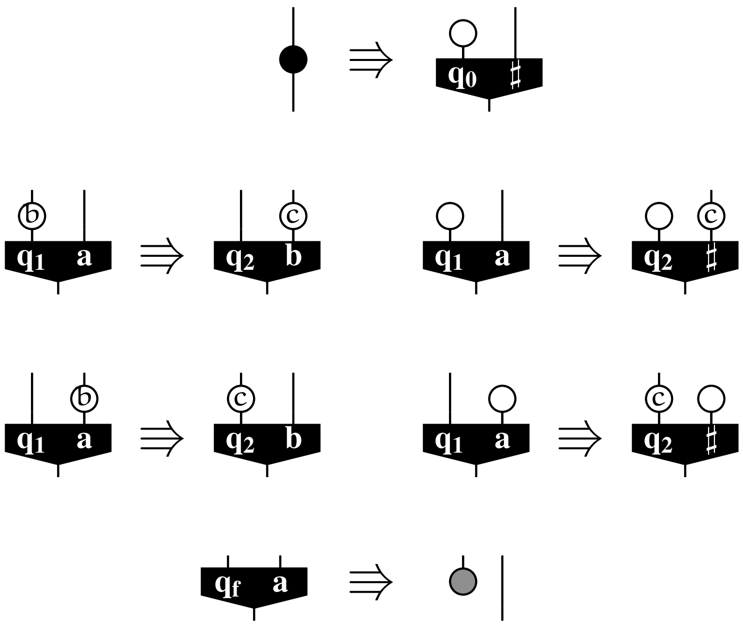
if $\delta(q_1, a) = (q_2, c, L)$

if $\delta(q_1, a) = (q_2, c, R)$

Computational power of polygraphic programs

Theorem: polygraphic programs form a Turing-complete model of computation.

Proof: build a polygraphic Turing machine:



if $\delta(q_1, a) = (q_2, c, L)$

if $\delta(q_1, a) = (q_2, c, R)$

Constructors: $\langle \circ, ⓐ, \ldots \rangle$ for words.

Functions: $\bullet$ (computed one) and $q\ a$ (state q, reading a, inputs are left and right parts of the tape).

Polygraphic programs

Polygraphic interpretations, termination and complexity bounds

Simplified definition

Simplified definition

Problem: find numeric interpretations that prove termination *and* give complexity bounds.

Simplified definition

Problem: find numeric interpretations that prove termination *and* give complexity bounds.

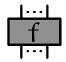
Electrical point of view: circuits are crossed by currents and gates produce heat.

Simplified definition

Problem: find numeric interpretations that prove termination *and* give complexity bounds.

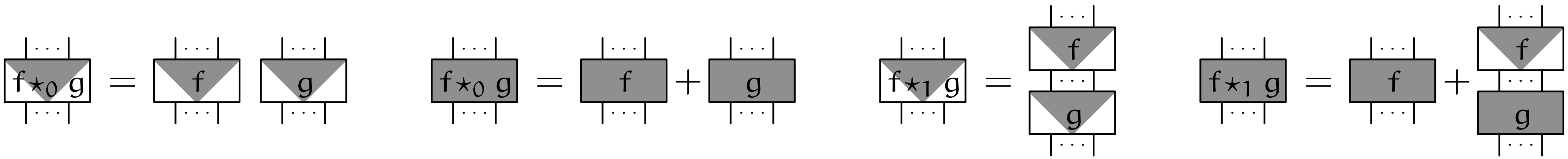
Electrical point of view: circuits are crossed by currents and gates produce heat.

A **polygraphic interpretation** (p.i.) sends a circuit $f : m \Rightarrow n$ onto monotone maps:

$f_* = (f_*^1, \ldots, f_*^n) =$  $: \mathbb{N}^m \to \mathbb{N}^n$ and $[f] =$  $: \mathbb{N}^m \to \mathbb{N}$,

such that:

- If x is a 1-cell: $x_* = \mathrm{Id}_{\mathbb{N}}$ and $[x] = 0$.
- If f and g are 2-paths:



Simplified definition

Problem: find numeric interpretations that prove termination *and* give complexity bounds.

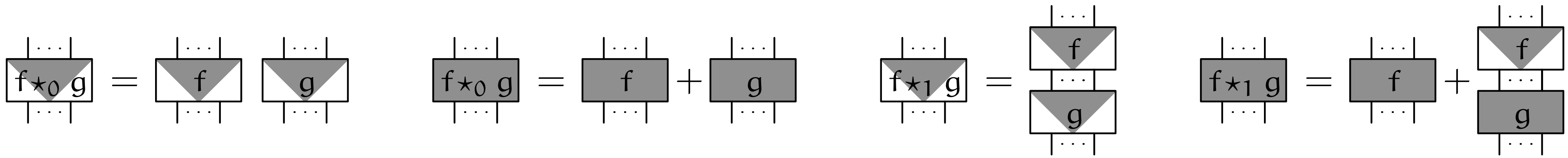
Electrical point of view: circuits are crossed by currents and gates produce heat.

A **polygraphic interpretation** (p.i.) sends a circuit $f : m \Rightarrow n$ onto monotone maps:

$$f_* = (f_*^1, \ldots, f_*^n) = \boxed{f} : \mathbb{N}^m \to \mathbb{N}^n \qquad \text{and} \qquad [f] = \boxed{f} : \mathbb{N}^m \to \mathbb{N},$$

such that:

- If x is a 1-cell: $x_* = \mathrm{Id}_{\mathbb{N}}$ and $[x] = 0$.
- If f and g are 2-paths:



Lemma: the maps $(\cdot)_*$ and $[\cdot]$ are entirely defined by their values on 2-cells.

Simplified definition

Problem: find numeric interpretations that prove termination *and* give complexity bounds.

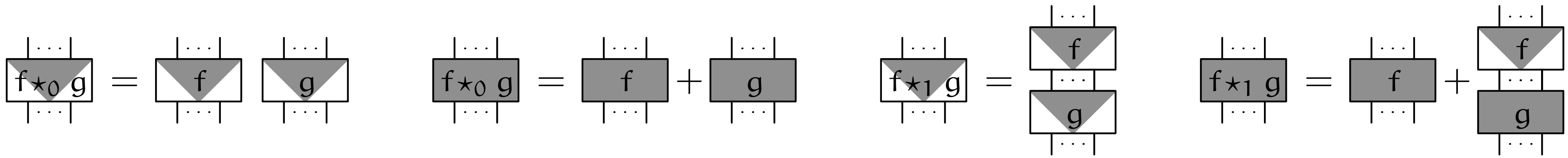
Electrical point of view: circuits are crossed by currents and gates produce heat.

A **polygraphic interpretation** (p.i.) sends a circuit $f : m \Rrightarrow n$ onto monotone maps:

$$f_* = (f_*^1, \ldots, f_*^n) = \boxed{f} : \mathbb{N}^m \to \mathbb{N}^n \qquad \text{and} \qquad [f] = \boxed{f} : \mathbb{N}^m \to \mathbb{N},$$

such that:

- If x is a 1-cell: $x_* = \mathrm{Id}_{\mathbb{N}}$ and $[x] = 0$.
- If f and g are 2-paths:



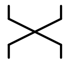
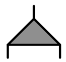
Lemma: the maps $(\cdot)_*$ and $[\cdot]$ are entirely defined by their values on 2-cells.

Order relation: $f \succ g$ when $f_* \geq g_*$ and $[f] > [g]$.

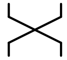
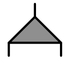
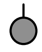
Compatibility with a 3-cell $\alpha : f \Rrightarrow g$ when $f \succ g$.

Examples of polygraphic interpretations

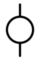
Examples of polygraphic interpretations

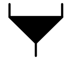
Standard interpretation of structure 2-cells: $_* = (Y, X)$, $_* = (X, X)$, $_* = *$, heats are 0.

Examples of polygraphic interpretations

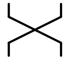
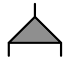
Standard interpretation of structure 2-cells: $_* = (Y, X)$, $_* = (X, X)$, $_* = *$, heats are 0.

Addition and multiplication:

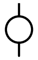
Constructors: $_* = 1$, $_* = X + 1$, heats are 0.

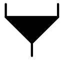
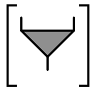
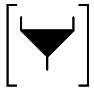
Functions: $_* = X + Y$, $_* = XY$, $\big[$$\big] = X$, $\big[$$\big] = XY + Y$.

Examples of polygraphic interpretations

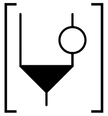
Standard interpretation of structure 2-cells:  $_* = (Y, X)$,  $_* = (X, X)$,  $_* = *$, heats are 0.

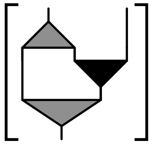
Addition and multiplication:

Constructors:  $_* = 1$,  $_* = X + 1$, heats are 0.

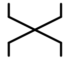
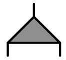
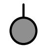
Functions:  $_* = X + Y$,  $_* = XY$,  $= X$,  $= XY + Y$.

Example:

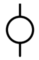
 (X, Y)

 (X, Y)

Examples of polygraphic interpretations

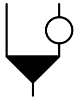
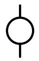
Standard interpretation of structure 2-cells:  $_* = (Y, X)$,  $_* = (X, X)$,  $_* = *$, heats are 0.

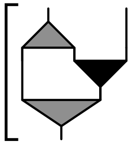
Addition and multiplication:

Constructors:  $_* = 1$,  $_* = X + 1$, heats are 0.

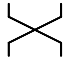
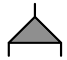
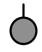
Functions:  $_* = X + Y$,  $_* = XY$, $\big[$  $\big] = X$, $\big[$  $\big] = XY + Y$.

Example:

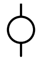
$\Big[$  $\Big] (X, Y) = \Big[$  $\Big] \Big(X,$  $_* (Y) \Big)$

$\Big[$  $\Big] (X, Y)$

Examples of polygraphic interpretations

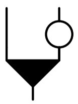
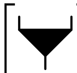
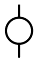
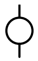
Standard interpretation of structure 2-cells: $_*$ $= (Y, X)$, $_*$ $= (X, X)$, $_*$ $= *$, heats are 0.

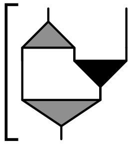
Addition and multiplication:

Constructors: $_*$ $= 1$, $_*$ $= X + 1$, heats are 0.

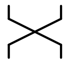
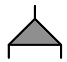
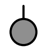
Functions: $_*$ $= X + Y$, $_*$ $= XY$, [] $= X$, [] $= XY + Y$.

Example:

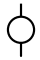
[] $(X, Y) =$ [] $\big(X,$ $_*$ $(Y)\big) +$ [] (Y)

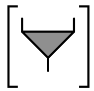
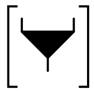
[] (X, Y)

Examples of polygraphic interpretations

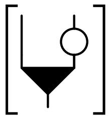
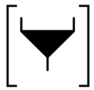
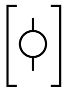
Standard interpretation of structure 2-cells:  $_* = (Y, X)$,  $_* = (X, X)$,  $_* = *$, heats are 0.

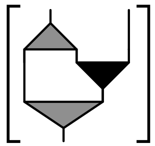
Addition and multiplication:

Constructors:  $_* = 1$,  $_* = X + 1$, heats are 0.

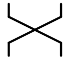
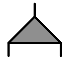
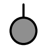
Functions:  $_* = X + Y$,  $_* = XY$,  $= X$,  $= XY + Y$.

Example:

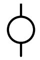
 $(X, Y) =$  $(X, Y+1) +$  (Y)

 (X, Y)

Examples of polygraphic interpretations

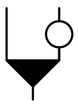
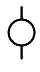
Standard interpretation of structure 2-cells:  $_* = (Y, X)$,  $_* = (X, X)$,  $_* = *$, heats are 0.

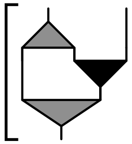
Addition and multiplication:

Constructors:  $_* = 1$,  $_* = X + 1$, heats are 0.

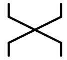
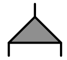
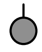
Functions:  $_* = X + Y$,  $_* = XY$, [] $= X$, [] $= XY + Y$.

Example:

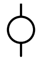
[] $(X, Y) = X(Y + 1) + (Y + 1) +$ [] (Y)

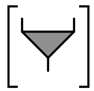
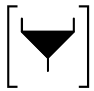
[] (X, Y)

Examples of polygraphic interpretations

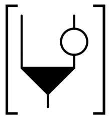
Standard interpretation of structure 2-cells:  $_* = (Y, X)$,  $_* = (X, X)$,  $_* = *$, heats are 0.

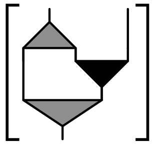
Addition and multiplication:

Constructors:  $_* = 1$,  $_* = X + 1$, heats are 0.

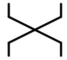
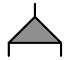
Functions:  $_* = X + Y$,  $_* = XY$,  $= X$,  $= XY + Y$.

Example:

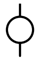
 $(X, Y) = X(Y + 1) + (Y + 1) + 0$

 (X, Y)

Examples of polygraphic interpretations

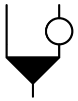
Standard interpretation of structure 2-cells: $\;$  $_* = (Y, X)$,  $_* = (X, X)$,  $_* = *$, heats are 0.

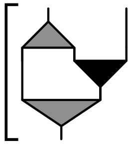
Addition and multiplication:

Constructors:  $_* = 1$,  $_* = X + 1$, heats are 0.

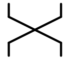
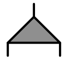
Functions:  $_* = X + Y$,  $_* = XY$, [] $= X$, [] $= XY + Y$.

Example:

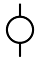
[] $(X, Y) = XY + X + Y + 1$,

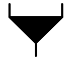
[] (X, Y)

Examples of polygraphic interpretations

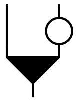
Standard interpretation of structure 2-cells:  $_* = (Y, X)$,  $_* = (X, X)$,  $_* = *$, heats are 0.

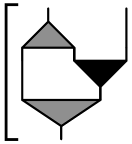
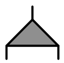
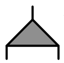
Addition and multiplication:

Constructors:  $_* = 1$,  $_* = X + 1$, heats are 0.

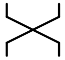
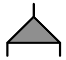
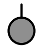
Functions:  $_* = X + Y$,  $_* = XY$, $[$  $] = X$, $[$  $] = XY + Y$.

Example:

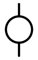
$[$  $](X, Y) = XY + X + Y + 1$,

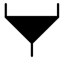
$[$  $](X, Y) = [$  $]\Big($  $^1_*(X),$  $_*\Big($  $^2_*(X), Y\Big)\Big)$

Examples of polygraphic interpretations

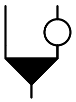
Standard interpretation of structure 2-cells:  $_* = (Y, X)$,  $_* = (X, X)$,  $_* = *$, heats are 0.

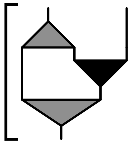
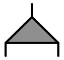
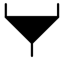
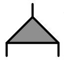
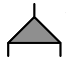
Addition and multiplication:

Constructors:  $_* = 1$,  $_* = X+1$, heats are 0.

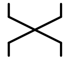
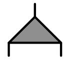
Functions:  $_* = X+Y$,  $_* = XY$, $\big[$  $\big] = X$, $\big[$  $\big] = XY+Y$.

Example:

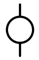
$\big[$  $\big](X, Y) = XY + X + Y + 1$,

$\big[$  $\big](X, Y) = \big[$  $\big]\Big($  $^1_*(X),$  $_*\Big($  $^2_*(X), Y\Big)\Big) + \big[$  $\big]\Big($  $^2_*(X), Y\Big)$

Examples of polygraphic interpretations

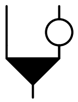
Standard interpretation of structure 2-cells:  $_* = (Y, X)$,  $_* = (X, X)$,  $_* = *$, heats are 0.

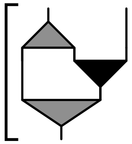
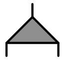
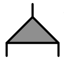
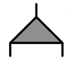
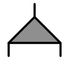
Addition and multiplication:

Constructors:  $_* = 1$,  $_* = X + 1$, heats are 0.

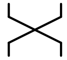
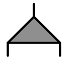
Functions:  $_* = X + Y$,  $_* = XY$, $[$  $] = X$, $[$  $] = XY + Y$.

Example:

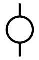
$[$  $](X, Y) = XY + X + Y + 1,$

$[$  $](X, Y) = [$  $]\big($  $^1_*(X),$  $_*\big($  $^2_*(X), Y\big)\big) + [$  $]\big($  $^2_*(X), Y\big) + [$  $](X)$

Examples of polygraphic interpretations

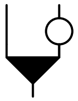
Standard interpretation of structure 2-cells:  $_* = (Y, X)$,  $_* = (X, X)$,  $_* = *$, heats are 0.

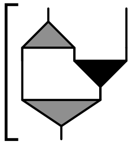
Addition and multiplication:

Constructors:  $_* = 1$,  $_* = X + 1$, heats are 0.

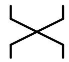
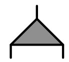
Functions:  $_* = X + Y$,  $_* = XY$, $\big[$  $\big] = X$, $\big[$  $\big] = XY + Y$.

Example:

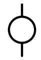
$\big[$  $\big]$ $(X, Y) = XY + X + Y + 1,$

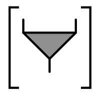
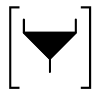
$\big[$  $\big]$ $(X, Y) = \big[$  $\big]$ $\Big(X,$  $_*(X, Y)\Big) + \big[$  $\big]$ $(X, Y) + 0$

Examples of polygraphic interpretations

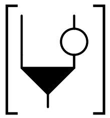
Standard interpretation of structure 2-cells:  $_* = (Y, X)$,  $_* = (X, X)$,  $_* = *$, heats are 0.

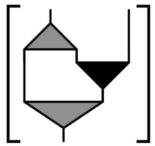
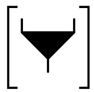
Addition and multiplication:

Constructors:  $_* = 1$,  $_* = X + 1$, heats are 0.

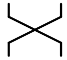
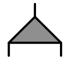
Functions:  $_* = X + Y$,  $_* = XY$,  $= X$,  $= XY + Y$.

Example: $\left\{ \right.$

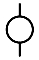
 $(X, Y) = XY + X + Y + 1,$

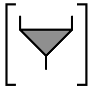
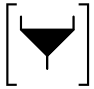
 $(X, Y) = X +$  (X, Y)

Examples of polygraphic interpretations

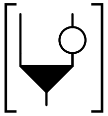
Standard interpretation of structure 2-cells: $_* = (Y, X)$, $_* = (X, X)$, $_* = *$, heats are 0.

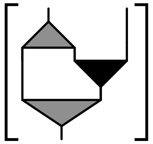
Addition and multiplication:

Constructors: $_* = 1$, $_* = X + 1$, heats are 0.

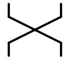
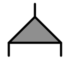
Functions: $_* = X + Y$, $_* = XY$,  $= X$,  $= XY + Y$.

Example:

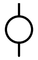
 $(X, Y) = XY + X + Y + 1,$

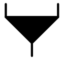
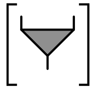
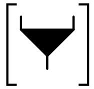
 $(X, Y) = X + (X + 1)Y$

Examples of polygraphic interpretations

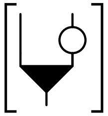
Standard interpretation of structure 2-cells:  $_* = (Y, X)$,  $_* = (X, X)$,  $_* = *$, heats are 0.

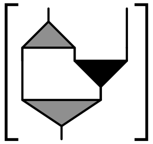
Addition and multiplication:

Constructors:  $_* = 1$,  $_* = X + 1$, heats are 0.

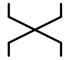
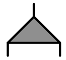
Functions:  $_* = X + Y$,  $_* = XY$,  $= X$,  $= XY + Y$.

Example:

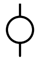
 $(X, Y) = XY + X + Y + 1,$

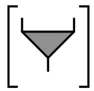
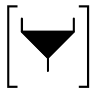
 $(X, Y) = XY + X + Y.$

Examples of polygraphic interpretations

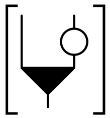
Standard interpretation of structure 2-cells:  $_* = (Y, X)$,  $_* = (X, X)$,  $_* = *$, heats are 0.

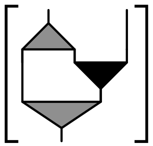
Addition and multiplication:

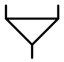
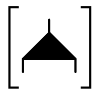
Constructors:  $_* = 1$,  $_* = X + 1$, heats are 0.

Functions:  $_* = X + Y$,  $_* = XY$,  $= X$,  $= XY + Y$.

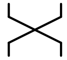
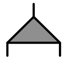
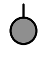
Example:

 $(X, Y) = XY + X + Y + 1,$

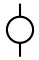
 $(X, Y) = XY + X + Y.$

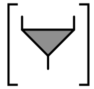
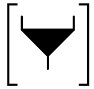
List splitting:  $_* = 1$,  $_* = X + Y + 1$,  $_* = \left(\left\lceil \frac{X}{2} \right\rceil, \left\lfloor \frac{X}{2} \right\rfloor\right)$,  $= X$.

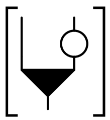
Examples of polygraphic interpretations

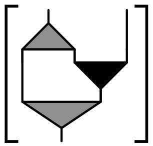
Standard interpretation of structure 2-cells: $_* = (Y, X)$, $_* = (X, X)$, $_* = *$, heats are 0.

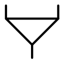
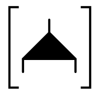
Addition and multiplication:

Constructors: $_* = 1$, $_* = X + 1$, heats are 0.

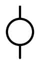
Functions: $_* = X + Y$, $_* = XY$,  $= X$,  $= XY + Y$.

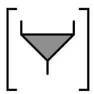
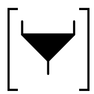
Example: $(X, Y) = XY + X + Y + 1$,

$(X, Y) = XY + X + Y$.

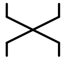
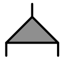
List splitting: $_* = 1$, $_* = X + Y + 1$, $_* = \left(\left\lceil \frac{X}{2} \right\rceil, \left\lfloor \frac{X}{2} \right\rfloor\right)$,  $= X$.

Integer division: minus is  and division is 

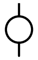
Constructors: $_* = 1$, $_* = X + 1$, heats are 0.

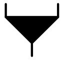
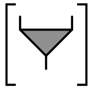
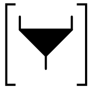
Functions: $_* = X$, $_* = X$,  $= Y + 1$,  $= XY + 2X$.

Examples of polygraphic interpretations

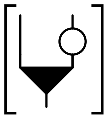
Standard interpretation of structure 2-cells:  $_* = (Y,X)$,  $_* = (X,X)$,  $_* = *$, heats are 0.

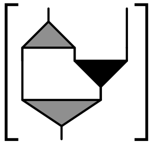
Addition and multiplication:

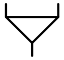
Constructors:  $_* = 1$,  $_* = X+1$, heats are 0.

Functions:  $_* = X+Y$,  $_* = XY$,  $= X$,  $= XY+Y$.

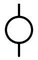
Example:

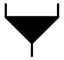
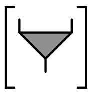
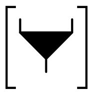
 $(X,Y) = XY+X+Y+1$,

 $(X,Y) = XY+X+Y$.

List splitting:  $_* = 1$,  $_* = X+Y+1$,  $_* = \left(\left\lceil \frac{X}{2} \right\rceil, \left\lfloor \frac{X}{2} \right\rfloor\right)$,  $= X$.

Integer division: minus is  and division is 

Constructors:  $_* = 1$,  $_* = X+1$, heats are 0.

Functions:  $_* = X$,  $_* = X$,  $= Y+1$,  $= XY+2X$.

Compatible with all computation rules: order $\succ$ is not a simplification order.

Termination results

Termination results

Theorem: if a polygraphic program admits a p.i. compatible with its 3-cells, then it terminates.

Termination results

Theorem: if a polygraphic program admits a p.i. compatible with its 3-cells, then it terminates.

A polygraphic program is **simple** when there is a p.i. "such as in the examples" and compatible with its *computation* 3-cells.

Termination results

Theorem: if a polygraphic program admits a p.i. compatible with its 3-cells, then it terminates.

A polygraphic program is **simple** when there is a p.i. "such as in the examples" and compatible with its *computation* 3-cells.

Theorem: simple polygraphic programs terminate.

Termination results

Theorem: if a polygraphic program admits a p.i. compatible with its 3-cells, then it terminates.

A polygraphic program is **simple** when there is a p.i. "such as in the examples" and compatible with its *computation* 3-cells.

Theorem: simple polygraphic programs terminate.

Proof:

– p.i. statisfies $f \succeq g$ for each structure 3-cell $\alpha : f \Rrightarrow g$;

– hence: program terminates iff structure 3-cells terminate;

– structure 3-cells terminate (with *structure heat map*).

Spatial complexity of simple programs

Spatial complexity of simple programs

Question: can we bound spatial complexity, *i.e.* the size $||u||$ of any intermediate value u produced by a computation?

Spatial complexity of simple programs

Question: can we bound spatial complexity, *i.e.* the size $||\mathfrak{u}||$ of any intermediate value $\mathfrak{u}$ produced by a computation?

Answer: yes, with currents maps.

Spatial complexity of simple programs

Question: can we bound spatial complexity, *i.e.* the size $||u||$ of any intermediate value u produced by a computation?

Answer: yes, with currents maps.

Lemma: for every value u, we have $||u|| \leq u_* \leq a||u||$, where a is a global positive constant, given by the current maps of the constructors.

Spatial complexity of simple programs

Question: can we bound spatial complexity, *i.e.* the size $\|u\|$ of any intermediate value u produced by a computation?

Answer: yes, with currents maps.

Lemma: for every value u, we have $\|u\| \leq u_* \leq a\|u\|$, where a is a global positive constant, given by the current maps of the constructors.

Proposition: for every function φ with n inputs and every family $u = (u_1, \ldots, u_n)$ of values:

$$\|\varphi(u)\| \leq P_\varphi(\|u_1\|, \ldots, \|u_n\|), \quad \text{where} \quad P_\varphi = \sum_{j=1}^{n} \varphi_*^j(aX_1, \ldots, aX_n).$$

The inequality also holds if $\varphi(u)$ is replaced by any intermediate value.

Spatial complexity of simple programs

Question: can we bound spatial complexity, *i.e.* the size $||u||$ of any intermediate value u produced by a computation?

Answer: yes, with currents maps.

Lemma: for every value u, we have $||u|| \leq u_* \leq a\,||u||$, where a is a global positive constant, given by the current maps of the constructors.

Proposition: for every function φ with n inputs and every family $u = (u_1, \ldots, u_n)$ of values:

$$||\varphi(u)|| \leq P_\varphi(||u_1||, \ldots, ||u_n||), \quad \text{where} \quad P_\varphi = \sum_{j=1}^{n} \varphi_*^j(aX_1, \ldots, aX_n).$$

The inequality also holds if $\varphi(u)$ is replaced by any intermediate value.

Examples:

- Addition and multiplication: $P_{\triangledown} = X + Y$ and $P_{\blacktriangledown} = XY$ $\quad$ ($\sim 2X + Y$ and $\sim XY + X + Y$ with terms).
- Division: $P_{\triangledown} = X$ and $P_{\blacktriangledown} = X$ $\quad$ (no bound with terms).
- List splitting: $P_{\blacktriangle} = \blacktriangle_*^1 + \blacktriangle_*^2 = \lceil \frac{X}{2} \rceil + \lfloor \frac{X}{2} \rfloor = X$ $\quad$ ($\sim 2X$ with terms).

Temporal complexity of simple programs

Temporal complexity of simple programs

Idea: the heat maps bound temporal complexity,

i.e. the number $|||\mathbb{F}|||$ of *computation* 3-cells in any normalizing 3-path $\mathbb{F}$.

Temporal complexity of simple programs

Idea: the heat maps bound temporal complexity,
i.e. the number $|||F|||$ of *computation* 3-cells in any normalizing 3-path F.

Proposition: for every function φ, every family $u = (u_1, \ldots, u_n)$ of values and every normalizing path $F : u \star_1 \varphi \Rrightarrow \varphi(u)$, we have $|||F||| \leq Q_\varphi(\|u_1\|, \ldots, \|u_n\|)$, where $Q_\varphi = [\varphi](aX_1, \ldots, aX_n)$.

Temporal complexity of simple programs

Idea: the heat maps bound temporal complexity,
i.e. the number $|||F|||$ of *computation* 3-cells in any normalizing 3-path F.

Proposition: for every function φ, every family $u = (u_1, \ldots, u_n)$ of values and every normalizing path $F : u \star_1 \varphi \Rrightarrow \varphi(u)$, we have $|||F||| \leq Q_{\varphi}(\|u_1\|, \ldots, \|u_n\|)$, where $Q_{\varphi} = [\varphi](aX_1, \ldots, aX_n)$.

Examples:

- Addition and multiplication: $Q_{\triangledown} = X$ and $Q_{\blacktriangledown} = XY + Y$ $\quad$ ($\sim 2X + Y$ and $\sim XY + X + Y$ with terms).
- Division: $Q_{\triangledown} = Y + 1$ and $Q_{\blacktriangledown} = XY + 2X$ $\quad$ (no bound with terms).
- List splitting: $Q_{\blacktriangle} = X$ $\quad$ ($\sim 2X$ with terms).

Temporal complexity of simple programs

Idea: the heat maps bound temporal complexity,
i.e. the number $|||F|||$ of *computation* 3-cells in any normalizing 3-path F.

Proposition: for every function φ, every family $u = (u_1, \dots, u_n)$ of values and every normalizing path $F : u \star_1 \varphi \Rrightarrow \varphi(u)$, we have $|||F||| \leq Q_\varphi(\|u_1\|, \dots, \|u_n\|)$, where $Q_\varphi = [\varphi](aX_1, \dots, aX_n)$.

Examples:

- Addition and multiplication: $Q_{\triangledown} = X$ and $Q_{\blacktriangledown} = XY + Y$ $\quad$ ($\sim 2X + Y$ and $\sim XY + X + Y$ with terms).
- Division: $Q_{\triangledown} = Y + 1$ and $Q_{\blacktriangledown} = XY + 2X$ $\quad$ (no bound with terms).
- List splitting: $Q_{\blacktriangle} = X$ $\quad$ ($\sim 2X$ with terms).

Theorem: simple programs compute exactly PTIME functions.

Temporal complexity of simple programs

Idea: the heat maps bound temporal complexity,
i.e. the number $|||F|||$ of *computation* 3-cells in any normalizing 3-path F.

Proposition: for every function φ, every family $u = (u_1, \ldots, u_n)$ of values and every normalizing path $F : u \star_1 \varphi \Rrightarrow \varphi(u)$, we have $|||F||| \leq Q_\varphi(\|u_1\|, \ldots, \|u_n\|)$, where $Q_\varphi = [\varphi](aX_1, \ldots, aX_n)$.

Examples:

- Addition and multiplication: $Q_{\triangledown} = X$ and $Q_{\blacktriangledown} = XY + Y$ $\quad$ ($\sim 2X + Y$ and $\sim XY + X + Y$ with terms).
- Division: $Q_{\triangledown} = Y + 1$ and $Q_{\blacktriangledown} = XY + 2X$ $\quad$ (no bound with terms).
- List splitting: $Q_{\blacktriangle} = X$ $\quad$ ($\sim 2X$ with terms).

Theorem: simple programs compute exactly PTIME functions.

Proof:

– use above proposition to bound computation 3-cells by a polynomial;

– build a polynomial bound for number of structure 3-cells;

– prove that each rewriting step takes polynomial time;

– build a polygraphic Turing machine with a clock.

Séminaire Prothéo
Polygraphic programs

Guillaume Bonfante – Yves Guiraud

INRIA – LORIA

Nancy – April 2, 2007