Associativity for Binary Parallel Processes: a Quantitative Study

Olivier Bodini[1], Antoine Genitrini[2], Frederic Peschanski[2] and Nicolas Rolin[1]

[1]Université Paris 13 – LIPN

[2]UPMC Paris – LIP6

CALDAM 2015

Motivation

Model

A minimal formalization of concurrent processes: we study use a sub-algebra of Milner's Calculus of Communicating Systems [Mi80].

Goal

- combinatorial study of concurrent processes
- algorithms tailored for this processes

Process grammar

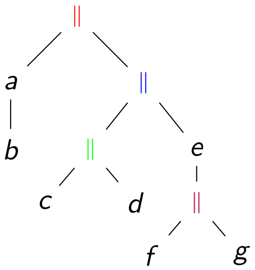
Specification for the processes:

- an atomic action, denoted $a, b, c, \ldots$ is a process,
- the prefixing $a.P$ of an action a followed by a process P is a process,
- the composition $P_1 \parallel P_2$ of exactly two processes P_1 and P_2, is a process.

Process trees

Reinterpretation as *process trees*.

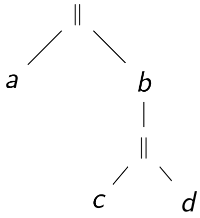
$$(a.b) \| [(c \| d) \| (e.(f \| g))].$$

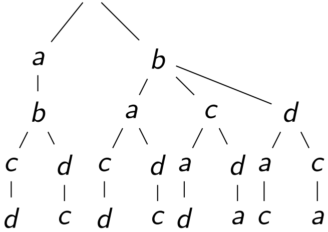


A run is a sequence of all the actions that satisfies precedence constraints.

Semantic trees

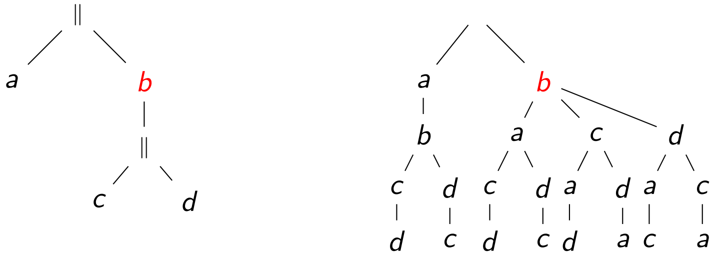
To express all the possible runs, we can also draw the *semantic tree*, in which every path to a leaf is a run. However the size grow exponentially with the size of the process tree, it is known as the combinatorial explosion [CGP99].





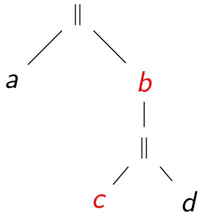
Semantic trees

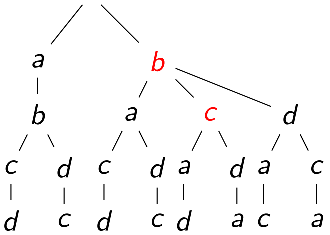
To express all the possible runs, we can also draw the *semantic tree*, in which every path to a leaf is a run. However the size grow exponentially with the size of the process tree, it is known as the combinatorial explosion [CGP99].



Semantic trees

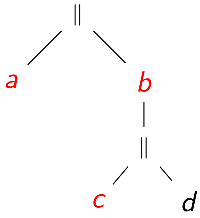
To express all the possible runs, we can also draw the *semantic tree*, in which every path to a leaf is a run. However the size grow exponentially with the size of the process tree, it is known as the combinatorial explosion [CGP99].

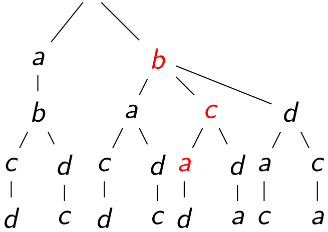




Semantic trees

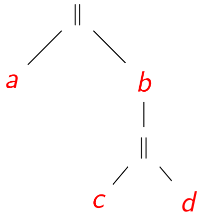
To express all the possible runs, we can also draw the *semantic tree*, in which every path to a leaf is a run. However the size grow exponentially with the size of the process tree, it is known as the combinatorial explosion [CGP99].

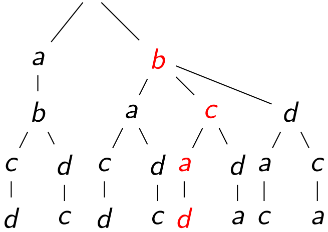




Semantic trees

To express all the possible runs, we can also draw the *semantic tree*, in which every path to a leaf is a run. However the size grow exponentially with the size of the process tree, it is known as the combinatorial explosion [CGP99].





Class definition

A *combinatorial class* $\mathcal{C}$: is a set of objects, with a size function, denoted by $|\cdot| : \mathcal{C} \to \mathbb{N}$ and such that for every integer n, the subset $\mathcal{C}_n$ of objects of size n, is finite with cardinality C_n.

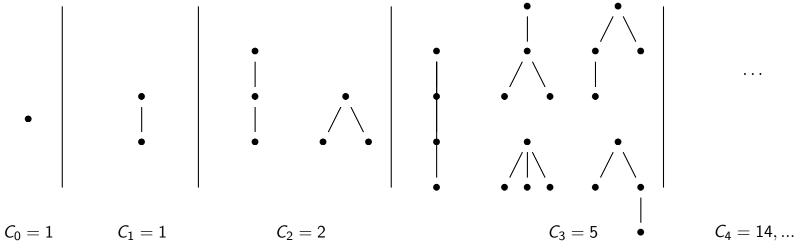


Figure : Catalan trees

Generating function

We define the *ordinary generating function* of a combinatorial class $\mathcal{C}$ to be:

$$C(z) = \sum_{n \geq 0} C_n z^n,$$

and the *exponential generating function* of a combinatorial class $\mathcal{C}$ to be:

$$C(z) = \sum_{n \geq 0} C_n \frac{z^n}{n!}.$$

example:

the first terms of the generating function $C(z)$ of the Catalan trees are:

$$C(z) = 1 + z + 2z^2 + 5z^3 + 14z^4 + ...$$

Symbolic method

We can translate specifications to generating functions automatically

Small dictionary

$$\begin{array}{rcl} \varepsilon & \rightarrow & 1 \\ \mathcal{Z} & \rightarrow & z \\ \mathcal{C} = \mathcal{A} + \mathcal{B} & \rightarrow & C(z) = A(z) + B(z) \\ \mathcal{C} = \mathcal{A} \times \mathcal{B} & \rightarrow & C(z) = A(z)B(z) \\ \mathcal{C} = Seq(\mathcal{A}) & \rightarrow & C(z) = \frac{1}{1-A(z)} \end{array}$$

example: Catalan trees can be specified as :

$$\mathcal{C} = \varepsilon + Seq(\mathcal{Z} \times \mathcal{C}),$$

Hence:

$$C(z) = 1 + \frac{1}{1 - zC(z)}.$$

Process tree

Our process have the following grammar :

$$P = a \mid a.P \mid P_1 \parallel P_2.$$

We transform it into a combinatorial specification

$$\mathcal{P} = \mathcal{Z} + \mathcal{Z} \times \mathcal{P} + \mathcal{P} \times \mathcal{P}.$$

Hence the generating function of class of process trees $P(z)$ verifies the following equation:

$$P(z) = z + zP(z) + P(z)^2.$$

Process tree

So the generating function is:

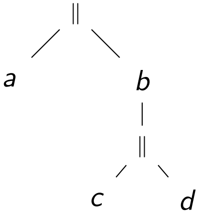
$$P(z) = \frac{1 - z - \sqrt{1 - 6z + z^2}}{2},$$

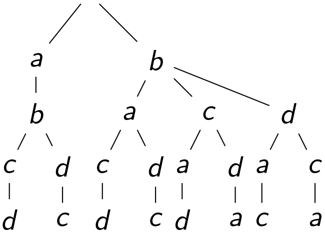
and we have an equivalent for the number of process trees:

$$P_n \sim_{n\to\infty} \sqrt{\frac{3\sqrt{2} - 4}{4\ \pi\ n^3}} \cdot \left(3 - 2\sqrt{2}\right)^{-n},$$

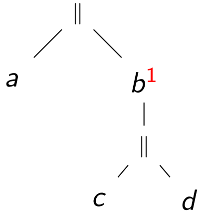
where $\left(3 - 2\sqrt{2}\right)^{-1} \approx 5.83$.
The first terms of the sequence are $1, 2, 6, 22, 90, \ldots$

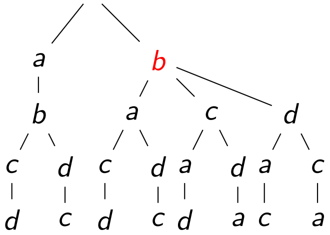
Increasing tree



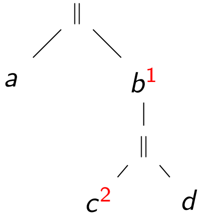


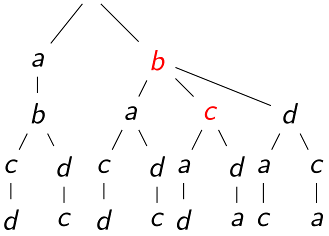
Increasing tree



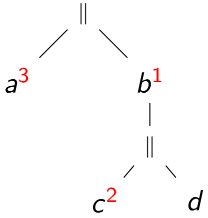


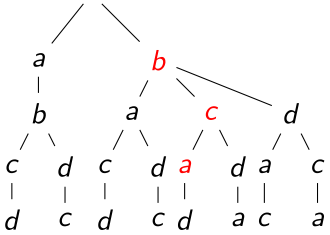
Increasing tree



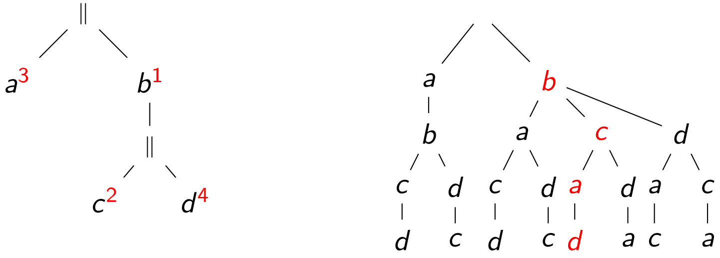


Increasing tree





Increasing tree



Boxed product

$$\mathcal{C} = \mathcal{A} \star^{\Box} \mathcal{B},$$

means that $\mathcal{C}$ is the product of $\mathcal{A}$ and $\mathcal{B}$, and that the element with the smallest label is in $\mathcal{A}$

Its translation to equation is:

$$\mathcal{C} = \mathcal{A} \star^{\Box} \mathcal{B} \rightarrow C(z) = \int_{v=0}^{v=z} \frac{dA}{dv}(v) B(v) dv.$$

Increasing labeled process

Hence the class of increasing labeled process $\mathcal{G}$ verifies the following specification:

$$\mathcal{G} = \mathcal{Z} \star^{\Box} (\mathcal{G} + 1) + \mathcal{G} \times \mathcal{G},$$

its generating function is:

$$G(z) = -1 - \frac{3}{2} \cdot \text{LambertW}\left(-\frac{2}{3}\exp\left(\frac{z-2}{3}\right)\right),$$

where the LambertW-function satisfies:

$$\text{LambertW}(z) \cdot \exp(\text{LambertW}(z)) = z.$$

However we have an equivalent for the number of increasing labeled process:

$$\bar{G}_n \sim_{n\to\infty} 3 \cdot \sqrt{\frac{\ln\frac{3}{2} - \frac{1}{3}}{6\sqrt{2} - 8}} \cdot \left(\frac{3 - 2\sqrt{2}}{3\left(\ln\frac{3}{2} - \frac{1}{3}\right)}\right)^n \cdot n!,$$

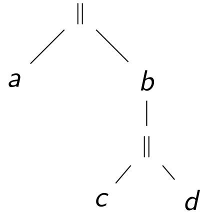
where $\left(\frac{3-2\sqrt{2}}{3\left(\ln\frac{3}{2}-\frac{1}{3}\right)}\right) \approx 0.79$.

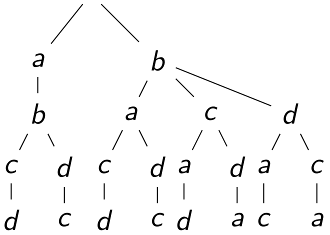
Size of the semantic tree

By using bivariate generating function, we can prove that

$$\bar{L}_n \sim_{n\to\infty} e \cdot \bar{G}_n,$$

where $\bar{L}_n$ is the mean total size of the semantic trees.





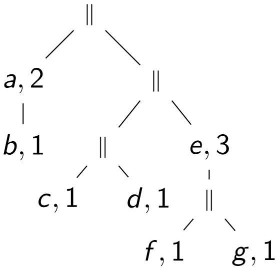
Hook length formula

The number of computations G_P on a given tree P is:

$$G_P = \frac{|P|!}{\prod_{P_\alpha \text{ prefixed subtree}} |P_\alpha|},$$

A prefixed subtree is a subtree with an atomic action as root.

Example



Thus for this example, the hook length formula gives:

$$G_P = \frac{7!}{2 \cdot 1 \cdot 1 \cdot 1 \cdot 3 \cdot 1 \cdot 1} = 840.$$

Uniform generatation of runs

Algorithm

Data : T: a weighted process tree of size n

Result : σ: a run (a list of nodes)

$\sigma := \langle\rangle$

$\mu := \{\!\{ a^{\text{weight}(a)} \}\!\}$ # initialize a multiset

for *i from* $n-1$ *to 1 by -1* **do**

invariant: the total weight of μ is i

$\alpha :=$sample(μ) # sample an action according to its cardinality

$\sigma := \sigma \cup \langle\alpha\rangle$ # append the sampled action

$\mu.weight(\alpha) := 0$ # α cannot be sampled anymore

$\mu := \mu \cup \{\!\{ \gamma^{\text{weight}(\gamma)} \mid \gamma \text{ a child of } \alpha \}\!\}$

insert the children of α in μ

return σ

Directly from the Hook Length Formula

Example of prefix generation

Example

$$(a.b) \,||\, [(c \,||\, d) \,||\, (e.(f \,||\, g))].$$

$\sigma := \langle\rangle$
$\mu := \{\!\{a, a, c, d, e, e, e\}\!\}$

Example of prefix generation

Example

$$(a.b) \,||\, [(c \,||\, d) \,||\, (e.(f \,||\, g))].$$

$\sigma := \langle\rangle$
$\mu := \{\!\{a, a, c, d, e, e, e\}\!\}$
$\alpha := e$ # happends with probability 3/7
$\sigma := \langle e \rangle$
$\mu := \{\!\{a, a, c, d, f, g\}\!\}$

Example of prefix generation

Example

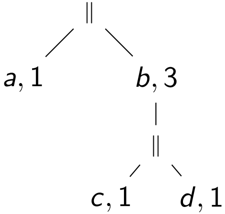
$$(a.b) \,||\, [(c \,||\, d) \,||\, (e.(f \,||\, g))].$$

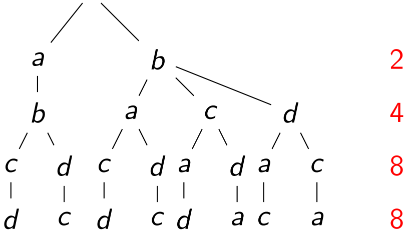
$\sigma := \langle\rangle$
$\mu := \{\!\{a, a, c, d, e, e, e\}\!\}$
$\alpha := e$ # happends with probability 3/7
$\sigma := \langle e\rangle$
$\mu := \{\!\{a, a, c, d, f, g\}\!\}$
$\alpha := c$ # happends with probability 1/6
$\sigma := \langle e, c\rangle$
$\mu := \{\!\{a, a, d, f, g\}\!\}$
...

The worst case complexity is $O(n \log n)$

Profile

Calculate the number of nodes for each level of the semantic tree





Profile

Here our process is seen as a specification for run prefixes:

$$P = a \,||\, (b.(c \,||\, d)),$$

we derive the combinatorial specification

$$P = \mathcal{Z} \times (\mathcal{Z} \star^{\square} (\mathcal{Z} \times \mathcal{Z})),$$

as we want profiles we add empty trees

$$P = (\mathcal{Z} + \epsilon) \times (\mathcal{Z} \star^{\square} ((\mathcal{Z} + \epsilon) \times (\mathcal{Z} + \epsilon)) + \epsilon),$$

and use our automatic method to deduce an equation:

$$P(z) = (z+1) \left(\int_{t=0}^{t=z} (t+1)(t+1)dt + 1 \right),$$

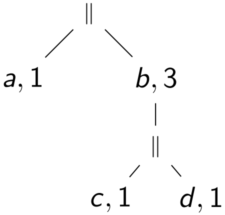
Profile

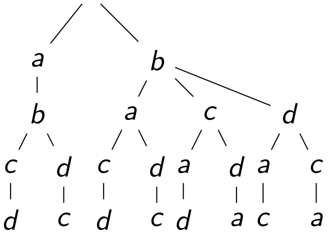
$$P(z) = 1 + 2\,z + 2\,z^2 + \frac{4}{3}z^3 + \frac{1}{3}z^4$$

As it is an exponential generating function, we need to multiply z^i by $i!$ to have the number of element.
Hence the profile of P is $1, 2, 4, 8, 8$.

The profile can be obtained in linear time.

Uniform sampling of run of prefixes





Uniform run of run of prefixes

$$P = a \,||\, (b.(c \,||\, d)),$$

This time we decorate all atomic element with his value:

$$P = \mathcal{Z}_a \times (\mathcal{Z}_b \star^{\square} (\mathcal{Z}_c \times \mathcal{Z}_d)).,$$

as we still want profiles we add empty trees

$$P = (\mathcal{Z}_a + \epsilon) \times (\mathcal{Z}_b \star^{\square} ((\mathcal{Z}_c + \epsilon) \times (\mathcal{Z}_d + \epsilon)) + \epsilon),$$

and use the automatic method to deduce an equation:

$$P(z) = (y_a z + 1) \left(\int_{t=0}^{t=z} y_b (y_c t + 1)(y_d t + 1) dt + 1 \right).$$

Uniform run of prefixes

So

$$P(z) = \frac{1}{3} y_a y_b y_c y_d z^4 + \frac{1}{6}\left(3\, y_a y_b y_c + 3\, y_a y_b y_d + 2\, y_b y_c y_d\right) z^3$$

$$+\frac{1}{2}\left(2 y_a y_b + \ y_b y_c + \ y_b y_d\right) z^2 + (y_a + y_b) z + 1$$

Uniform run of prefixes

Goal: sample uniformly a run amongst the runs of size ℓ

Algorithm in 3 steps.

Uniform run of prefixes: 1st step

choose a set of actions according to its distribution at size ℓ

example: $\ell = 3$ and

$$P(z) = \frac{1}{3}\, y_a y_b y_c y_d z^4 + \frac{1}{6}\left(3\, y_a y_b y_c + 3\, y_a y_b y_d + 2\, y_b y_c y_d\right) z^3$$

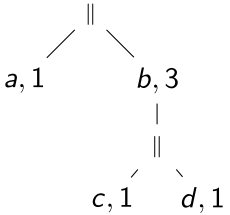
$$+\frac{1}{2}\left(2 y_a y_b + \; y_b y_c + \; y_b y_d\right) z^2 + (y_a + y_b) z + 1$$

We choose *abc* with probabilty $\frac{3}{8}$, *abd* with probabilty $\frac{3}{8}$ and *bcd* with probabilty $\frac{1}{4}$.

Uniform run of prefixes: 2nd step

Remove all the unused actions from the tree and remove the $\|$ operator on nodes that are no longer binary, it gives you a new process tree sample uniformly a run from that new tree.

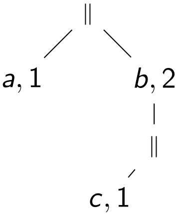
example: with the triplet *abc*:



Uniform run of prefixes: 2nd step

Remove all the unused actions from the tree and remove the $\|$ operator on nodes that are no longer binary, it gives you a new process tree sample uniformly a run from that new tree.

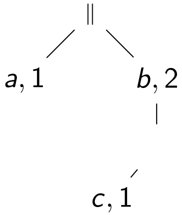
example: with the triplet *abc*:



Uniform run of prefixes: 2nd step

Remove all the unused actions from the tree and remove the ∥ operator on nodes that are no longer binary, it gives you a new process tree sample uniformly a run from that new tree.

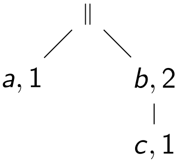
example: with the triplet *abc*:



Uniform run of prefixes: 2nd step

Remove all the unused actions from the tree and remove the $\|$ operator on nodes that are no longer binary, it gives you a new process tree sample uniformly a run from that new tree.

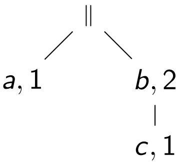
example: with the triplet *abc*:



Uniform run of prefixes: 3rd step

sample an uniform run from the tree of the 2nd step

example:



Conclusion

We have:

- An asymptotic analysis of the semantic tree: mean number of nodes and number of leaves
- An efficient algorithm to generate runs uniformly
- An efficient algorithm to compute the profile of a semantic tree
- An efficient algorithm to generate run prefixes uniformly

Thanks for your attention

Backup slide

Data : T: a weighted process tree of size n

Result : σ: a run (a list of nodes)

$\sigma := \langle\rangle$

$\mu := \{\!\{ a^{\mathrm{weight}(a)} \}\!\}$ # initialize a multiset

for *i from* $n-1$ *to 1 by -1* **do**

invariant: the total weight of μ is i

$\alpha :=$sample(μ) # sample an action according to its cardinality

$\sigma := \sigma \cup \langle\alpha\rangle$ # append the sampled action

$\mu.weight(\alpha) := 0$ # α cannot be sampled anymore

$\mu := \mu \cup \{\!\{ \gamma^{\mathrm{weight}(\gamma)} \mid \gamma \text{ a child of } \alpha \}\!\}$

insert the children of α in μ

return σ