Étude quantitative et exploration statistique des processus concurrents

Equipe Algortithmes, Programmes et Résolution

Étude quantitative et Génération aléatoire

Idée

- (ré-)interpréter les objets syntaxiques et sémantiques en termes de structures combinatoires
- utiliser les outils de la combinatoire analytique (séries génératrices, etc.) pour les étudier quantitativement
- en déduire des algorithmes efficaces de génération aléatoire permettant d'analyser de façon probabiliste les structures sémantiques (généralement de taille exponentielle) sans les construire explicitement

- Outils d'analyse
  - Combinatoire analytique
  - Modélisation algorithmique
- Points forts de l'étude quantitative
  - Modélisation via des structures combinatoires complexes
  - Analyse en moyenne de classes de grands graphes
- Points forts de l'algorithmique
  - Génération aléatoire uniforme : méthode récursive, de Boltzmann ou ad hoc
  - Efficacité en temps et mémoire
- Notre contribution
  - Développements d'outils pour la Combinatoire Analytique
  - Développements logiciels (génériques et dédiés *ArboGen* : https://github.com/fredokun/arbogen)
  - Domaines applicatifs en perspective : model cheching statistique
- Interactions
  - ANR-MOST – MetACOnc (ANR-15-CE40-0014)
  - Spécialistes domaines : grands graphes, tests logiciels

Théorie de la concurrence

Langages et autres formalismes permettant de caractériser formellement les phénomènes de concurrence ⟹ parallélisme, non-déterminisme, synchronisation, etc.

Illustrations : diverses algèbres de processus simplifiées

Syntaxe un processus est construit via :

- des actions atomiques
- la concaténation
- le parallélisme
- le non-déterminisme
- la synchronisation

Sémantique On s'intéresse aux diverses exécutions d'un processus.

Observation : Explosion Combinatoire

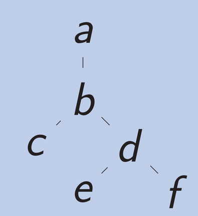
Modélisation contrées sur les contraintes de précédence entre les actions du processus : liens avec la théorie des ordres partiels.

Conséquence : Le comptage du nb d'executions est un problème #-P complet pour un processus arbitraire (Brightwell, Winkler (91)).

Parallélisme et Non-déterminisme

Un terme syntaxique et sa représentation arborescente :

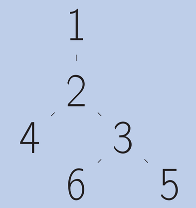
$$P = a.b.\,(c \parallel d.(e \parallel f))$$



La spécification combinatoire d'un processus est

$$\mathcal{P} = \mathcal{Z} \times \mathrm{Seq}(\mathcal{P}).$$

Une exécution correspond à un étiquetage croissant du processus : $\langle a, b, d, c, f, e\rangle$



La spécification combinatoire d'une exécution est

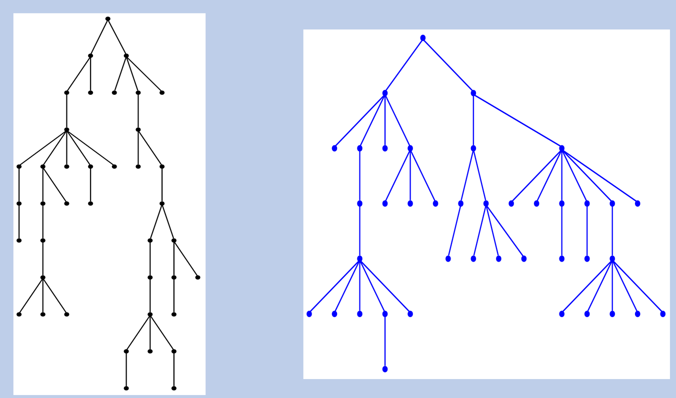
$$\mathcal{I} = \mathcal{Z}^{\square} \star \mathrm{Seq}(\mathcal{I}).$$

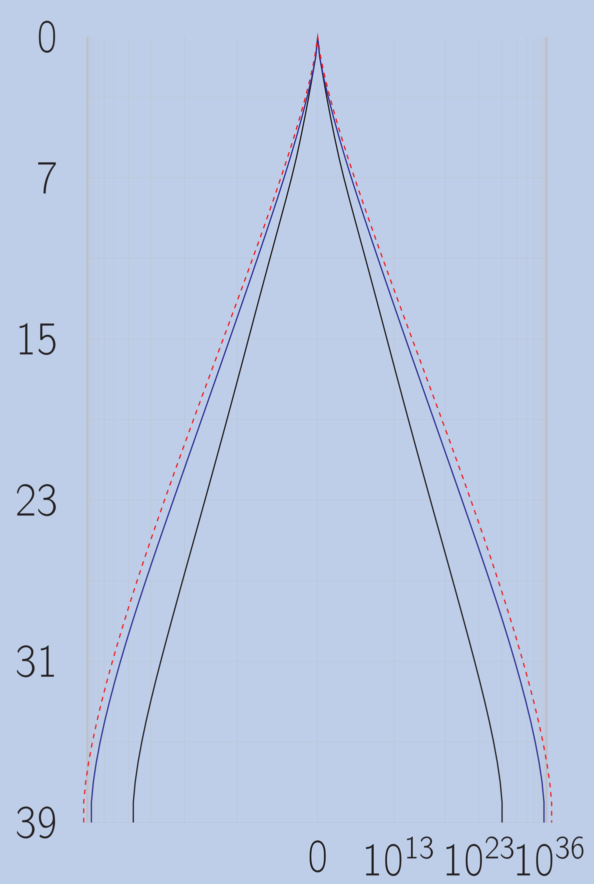
L'opérateur $\cdot^{\square}\star$ encode la contrainte de croissance.

Le nombre moyen d'exécutions d'un processus de taille n :

$$\frac{n!}{2^{n-1}} \sim_{n\to\infty} 2\sqrt{2\pi n}\left(\frac{n}{2e}\right)^n$$

Voici deux arbres typiques de taille 40, suivis de la représentation par arbre préfixe de leurs exécutions.



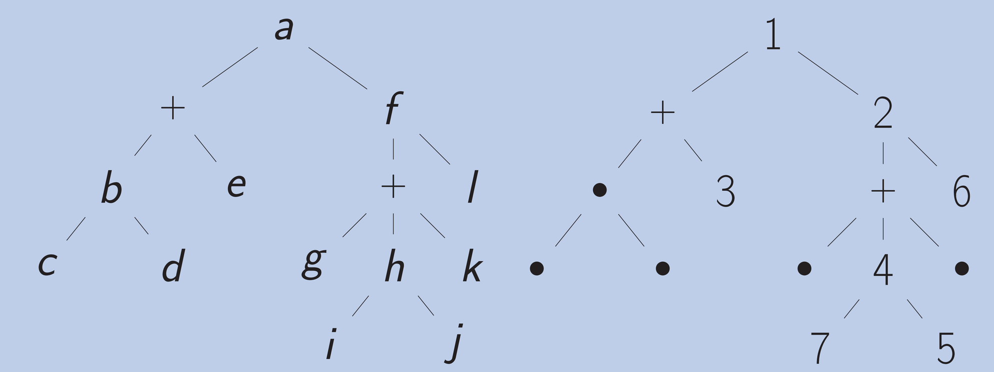


Enumeration and Random Generation of Concurrent Computations : O. Bodini, A. Genitrini et F. Peschanski ; Anaysis of Algorithms, 2012.

Associativity for Binary Parallel Processes: a Quantitative Study : O. Bodini, A. Genitrini, N. Rolin et F. Peschanski ; Conference on Algorithms and Discrete Applied Mathematics, 2015.

A quantitative study of pure parallel processes : O. Bodini, A. Genitrini et F. Peschanski ; Electronic Journal of Combinatorics, 2016.

Un terme syntaxique : Une exécution :



La spécification combinatoire d'une exécution est encodée dans la classe $\mathcal{B}$ telle que :

$$\left[\begin{array}{lcl} \mathcal{A} &=& \mathcal{A}_{|||} + \mathcal{A}_{+} \\ \mathcal{A}_{|||} &=& \mathcal{Z} \times \mathrm{Seq}\,(\mathcal{A}) \\ \mathcal{A}_{+} &=& \mathcal{A}_{||} \times \mathcal{A}_{||} \times \mathrm{Seq}(\mathcal{A}_{|||}) \end{array}\right.$$

$$\left[\begin{array}{lcl} \mathcal{B} &=& \mathcal{B}_{||} + \mathcal{B}_{+} \\ \mathcal{B}_{|||} &=& \mathcal{W}^{\square w} \star \mathcal{Z} \times \mathrm{Seq}(\mathcal{B}) \\ \mathcal{B}_{+} &=& (\mathcal{B}_{|||} \times \mathcal{A}_{|||} \times \mathrm{Seq}(\mathcal{A}_{|||})) \\ && + (\mathcal{A}_{|||} \times \mathrm{Seq}(\mathcal{A}_{|||}) \\ && \times \mathcal{B}_{|||} \times \mathrm{Seq}(\mathcal{A}_{|||})) \end{array}\right.$$

Le nombre moyen d'exécutions d'un processus de taille n :

$$\Theta\left(n!\,(6-4\sqrt{2})^n\right),\ \text{où } 6-4\sqrt{2} \approx 0.34315\ldots$$

Cette quantité moyenne étant exponentiellement plus petite que dans le cas des processus purement parallèle, l'opérateur de choix non-déterministe engendre un réel effet de coupure dans le nombre d'exécutions des processus.

Génération aléatoire uniforme d'une exécution, pour un processus fixé, en 2 étapes :

- Choisir un seul descendant des nœuds + (avec la distribution adéquate)

le nombre de possibilités est exponentiel !

- Choisir un étiquetage croissant (formule des équerres)

Dans ce but, on adapte la méthode génération récursive en considérant chaque processus comme la classe combinatoire de ses exécutions :

$$\mathcal{Z}^{\square} \star \left(\left(y_b \mathcal{Z}^{\square} \star \mathcal{Z}^2 + y_e \mathcal{Z}\right) \times \mathcal{Z}^{\square} \star \left(\left(y_g \mathcal{Z} + y_h \mathcal{Z}^{\square} \star \mathcal{Z}^2 + y_k \mathcal{Z}\right) \times \mathcal{Z}\right)\right).$$

La traduction via la méthode symbolique donne :

$$\int_0^z \left[y_b \cdot \int_0^t u^2 du + y_e \cdot t\right] \cdot \left[\int_0^t \left(y_g \cdot u + y_h \cdot \int_0^u v^2 dv + y_k \cdot u\right) \cdot u\, du\right] dt.$$

On obtient ainsi la fonction génératrice (exponentielle) énumérant les exécutions :

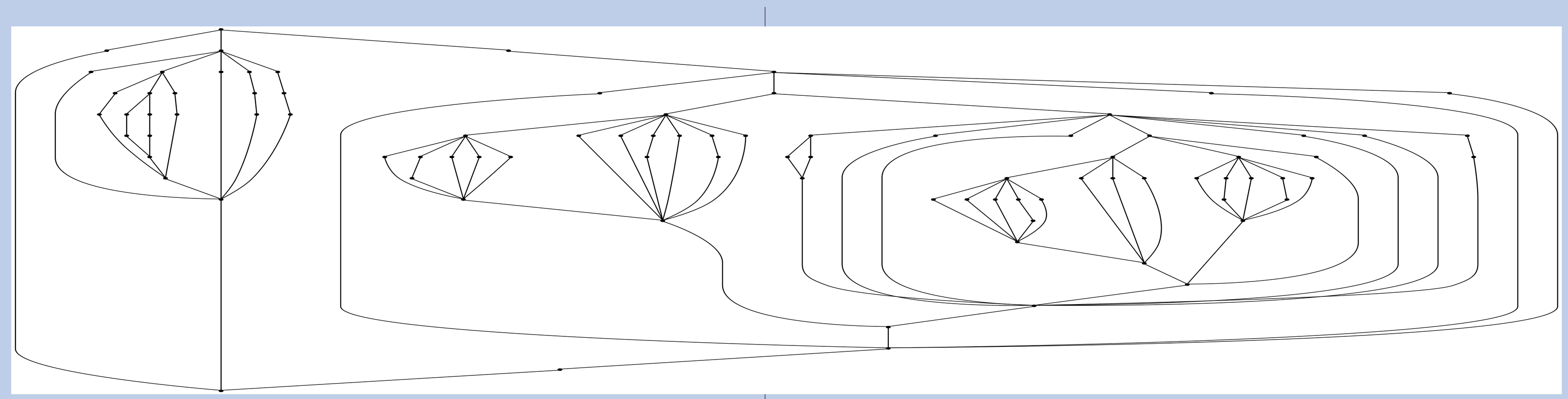
$$\mathcal{P}(z) = \tfrac{896\,z^9}{9!}\cdot y_b y_h + \tfrac{16\,z^7}{7!}\cdot(5y_b(y_g+y_k) + 3y_e y_h) + \tfrac{8\,z^5}{5!}\cdot y_e(y_g+y_k).$$

Théorème

Soit P un processus de taille n construit avec les opérateurs de concaténation, de parallélisme et de non-déterminisme.
Le polynôme $\mathcal{P}(z)$ est construit en $O(n^2)$ opérations arithmétiques, et en complexité $O(n^2)$ en espace.
Puis, chaque exécution est générée uniformément en $O(n \log n)$ complexité en temps.

The Combinatorics of Non-determinism : O. Bodini, A. Genitrini et F. Peschanski ; Conference on Foundations of Software Technology and Theoretical Computer Science, 2013.

Parallélisme et Synchronisation



Processus diamant d'arité arbitraire

La spécification combinatoire d'un diamant binaire est

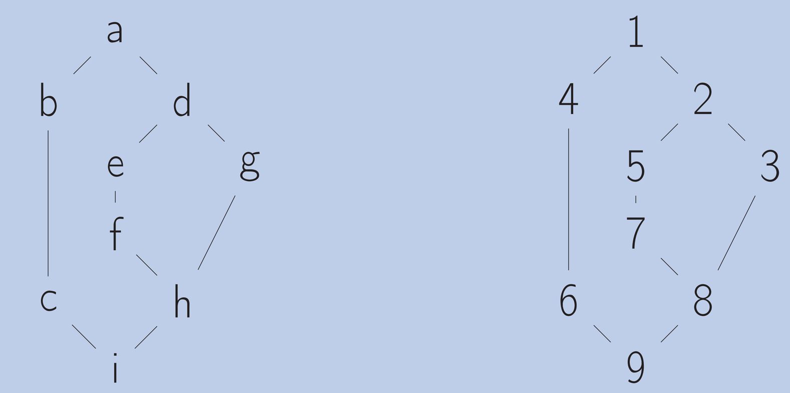
$$\mathcal{D} = \mathcal{Z} + \mathcal{Z} \times (\epsilon + \mathcal{D} \times \mathcal{D}) \times \mathcal{Z}.$$

Ainsi, la spécification des exécutions est

$$\mathcal{S} = \mathcal{Z} + \mathcal{Z}^{\square} \star (\epsilon + \mathcal{S} + \mathcal{S} \star \mathcal{S}) \star \mathcal{Z}^{\blacksquare}.$$

Les opérateurs $\cdot^{\square}\star$ et $\star\cdot^{\blacksquare}$ permettent d'encoder le fait que les éléments source et puits aient respectivement la plus petite et la plus grande étiquette.

Un terme syntaxique : Une exécution :

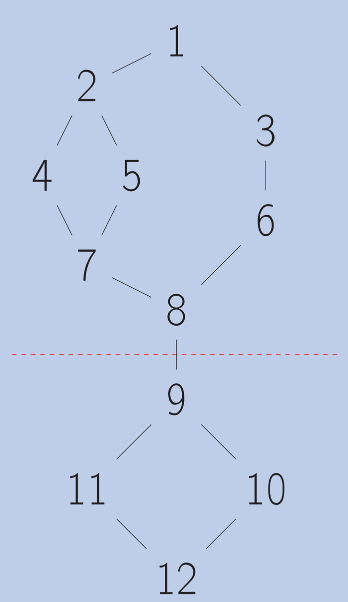


Cette dernière spécification, encodant les exécutions des processus se traduit en une équation aux dérivées partielles d'ordre 2, et se résout via les fonctions $\wp$ de Weierstrass.

Le nombre moyen d'exécutions d'un processus diamant binaire de taille n :

$$\Theta\left((n+1)!\ \alpha^n\right),\ \text{où } \alpha \approx 0.13689\ldots$$

Pour ajouter de l'expressivité, nous mettons des processus diamants en série. La méthode symbolique n'est pas assez riche pour spécifier la contrainte de croissance.
Nous introduisons un nouvel opérateur $\star$: $S_2 = S \star S$, qui encode le fait que toutes les étiquettes du premier diamant sont plus petites que les étiquettes du deuxième.



Increasing Diamonds : O. Bodini, M. Dien, X. Fontaine, A. Genitrini et H.-K. Hwang ; Latin American Theoretical INformatics Symposium, 2016.

The Ordered and Colored Products in Analytic Combinatorics: Application to the Quantitative Study of Synchronizations in Concurrent Processes : O. Bodini, M. Dien, A. Genitrini et F. Peschanski ; Meeting on Analytic Algorithmics and Combinatorics, 2017.

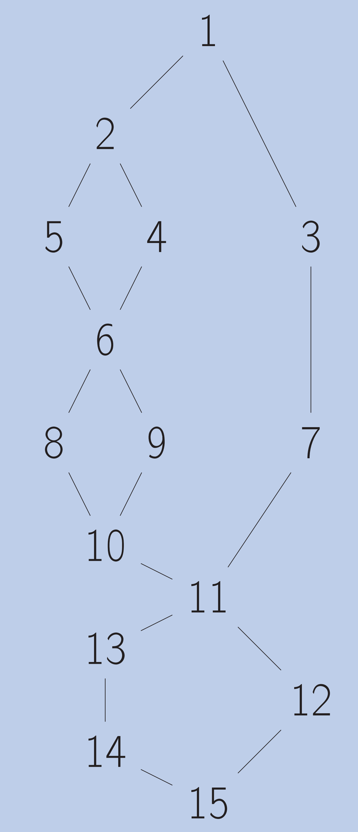
Avec ce nouvel opérateur, on peut spécifier les processus Fork/Join

$$\mathcal{F} = \mathcal{Z} + \mathcal{Z} \times \mathcal{F} + \mathcal{Z} \times (\mathcal{F} \times \mathcal{F}) \times \mathcal{F}.$$

La spécification des exécutions est

$$\mathcal{T} = \mathcal{Z} + \mathcal{Z}^{\square} \star \mathcal{T} + \mathcal{Z}^{\square} \star (\mathcal{T} \star \mathcal{T}) \star \mathcal{T}.$$

En adaptant les spécifications, en faisant intervenir des actions muettes, on est en mesure d'encoder tout processus série-parallèle.



On obtient un algorithme bottom-up de génération uniforme d'exécécutions d'un processus Fork/Join.

```
function RandExec(P)
    if P = ○ then return [ ]
    else if P = •x then return [x]
    else if P = •x . T then return cons(x, RandExec(Q))
    else if P = □ . (L ∥ R) . T then
        h := Shuffle(RandExec(L), RandExec(R))
        t := RandExec(T)
        if □ = •x then return concat(cons(x, h), t)
        else return concat(h, t)
```

En ajustant la fonction Shuffle, on obtient un algorithme entropique en nombre de bits aléatoires consommés.

Entropic Uniform Sampling of Linear Extensions in Series-Parallel Posets : O. Bodini, M. Dien, A. Genitrini et F. Peschanski ; International Computer Science Symposium in Russia, 2017.

Travaux en cours et Perspectives

- Étude combinatoire de structures de graphes dirigés acycliques étiquetés de façon croissante
- Étude de classes de processus au-delà du cadre série-parallèle : processus cordaux, cycle-free, ...
- Développement d'un prototype de model-checker statistique proposant un langage d'entrée inspiré de Promela de l'outil Spin
- Compression des arbres préfixe des exécutions