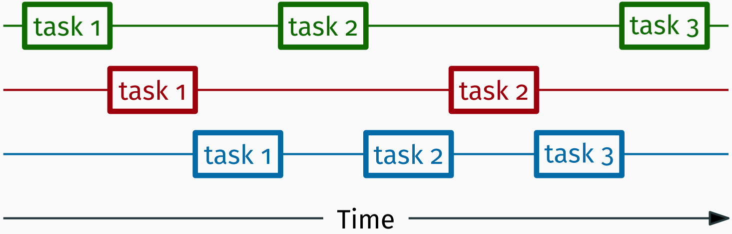
Statistical Analysis of Non-Deterministic Fork-Join Processes

Martin Pépin
Joint work with Antoine Genitrini & Frédéric Peschanski
Accepted for publication at ICTAC'20
October 13, 2020

Sorbonne Université — LIP6 — Paris

What is concurrency?



One computation unit shared by several processes:

→ Possible dependencies between processes

→ Scheduling

Why is it difficult?

You would like to check that **all possible schedulings** are correct.

Why is it difficult?

You would like to check that **all possible schedulings** are correct.

"**all schedulings**" → Combinatorics!

- Many possible schedulings: combinatorial explosion

Why is it difficult?

You would like to check that **all possible schedulings** are correct.

“**all schedulings**” → Combinatorics!

- Many possible schedulings: combinatorial explosion
- Can we (efficiently) count them?
- Can we (efficiently) sample among them?

Why is it difficult? (2)

Negative result (Brightwell & Winkler ’91)

Counting the linear extensions of a partial order is a *#-P* complete problem.

I.e. it is as hard as counting the number of solutions in SAT.

Why is it difficult? (2)

Negative result (Brightwell & Winkler '91)

Counting the linear extensions of a partial order is a *#-P* complete problem.

I.e. it is as hard as counting the number of solutions in SAT.

So we cannot count efficiently... in the **general** case.
But we can have some restrictions on the programs.

The long-term project

"Quantitative and algorithmic aspects of concurrency"

- Olivier Bodini, Matthieu Dien, Antoine Genitrini, MP, Frédéric Peschanski, ...
- Identify **fundamental** components of concurrency and **interpret** them as combinatorial objects
- Algorithmic solutions for the **counting** and **sampling** problems
- Analytical results (when possible)

Outline

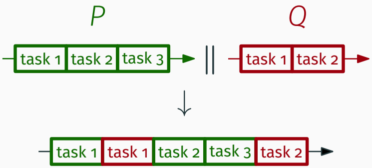
A class of concurrent programs

Algorithmic aspects

Conclusion and perspective

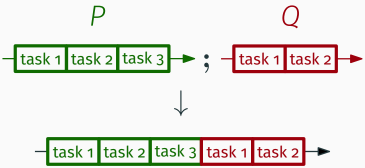
Fork-Join parallelism

Parallel composition



Execution = any interleaving of an execution of *P* and an execution of *Q*.

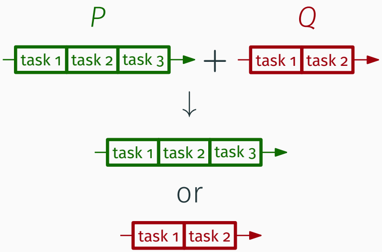
Sequential composition



Execution = an execution of *P* followed by an execution of *Q*.

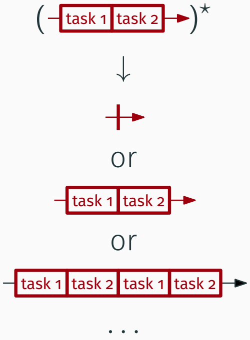
Non-determinism and loops

Non-deterministic choice



Execution = an execution of P or an execution of Q.

Loop



Execution = sequence of executions of Q

Non-deterministic Fork-Join programs (NFJ)

$$
\begin{array}{rcll}
P, Q & ::= & P \parallel Q & \text{(parallel composition)} \\
& \mid & P; Q & \text{(sequential composition)} \\
& \mid & P + Q & \text{(non-deterministic choice)} \\
& \mid & P^\star & \text{(loop)} \\
& \mid & a & \text{(atomic action)} \\
& \mid & 0 & \text{(empty program)}
\end{array}
$$

Combinatorial interpretation

Define the executions of P as a combinatorial class $[\![P]\!]$:

$$[\![0]\!] = \mathcal{E}$$

$$[\![a]\!] = \mathcal{Z}$$

Combinatorial interpretation

Define the executions of P as a combinatorial class $[\![P]\!]$:

$$
\begin{aligned}
[\![0]\!] &= \mathcal{E} \\
[\![a]\!] &= \mathcal{Z} \\
[\![P; Q]\!] &= [\![P]\!] \times [\![Q]\!] \\
[\![P \parallel Q]\!] &= [\![P]\!] \star [\![Q]\!]
\end{aligned}
$$

- Labelled and unlabelled operators in the same grammar;

Combinatorial interpretation

Define the executions of P as a combinatorial class $[\![P]\!]$:

$$
\begin{aligned}
[\![0]\!] &= \mathcal{E} \\
[\![a]\!] &= \mathcal{Z} \\
[\![P;Q]\!] &= [\![P]\!] \times [\![Q]\!] \\
[\![P \parallel Q]\!] &= [\![P]\!] \star [\![Q]\!] \\
[\![P+Q]\!] &= [\![P]\!] + [\![Q]\!]
\end{aligned}
$$

- Labelled and unlabelled operators in the same grammar;

Combinatorial interpretation

Define the executions of P as a combinatorial class $[\![P]\!]$:

$$
\begin{aligned}
[\![0]\!] &= \mathcal{E} \\
[\![a]\!] &= \mathcal{Z} \\
[\![P;Q]\!] &= [\![P]\!] \times [\![Q]\!] \\
[\![P \parallel Q]\!] &= [\![P]\!] \star [\![Q]\!] \\
[\![P+Q]\!] &= [\![P]\!] + [\![Q]\!] \\
[\![P^\star]\!] &= \textsc{Seq}\left([\![P]\!]\right)
\end{aligned}
$$

-
-

- Labelled and unlabelled operators in the same grammar;

Combinatorial interpretation

Define the executions of P as a combinatorial class $[\![P]\!]$:

$$
\begin{aligned}
[\![0]\!] &= \mathcal{E} \\
[\![a]\!] &= \mathcal{Z} \\
[\![P; Q]\!] &= [\![P]\!] \times [\![Q]\!] && \bullet \\
[\![P \parallel Q]\!] &= [\![P]\!] \star [\![Q]\!] && \bullet \\
[\![P + Q]\!] &= [\![P]\!] + [\![Q]\!] && \triangle \\
[\![P^\star]\!] &= \textsc{Seq}\left([\![P]\!]\right) && \triangle
\end{aligned}
$$

- Labelled and unlabelled operators in the same grammar;

⚠ $[\![P]\!]$, $[\![Q]\!]$ might contain the empty execution;

Combinatorial interpretation

Define the executions of P as a combinatorial class $[\![P]\!]$:

$$[\![0]\!] = \mathcal{E}$$

$$[\![a]\!] = \mathcal{Z}$$

$$[\![P;Q]\!] = [\![P]\!] \times [\![Q]\!]$$ ●

$$[\![P \parallel Q]\!] = [\![P]\!] \star [\![Q]\!]$$ ●

$$\cancel{[\![P+Q]\!] = [\![P]\!] + [\![Q]\!]}$$ ⚠

$$\cancel{[\![P^\star]\!] = \textsc{Seq}\left([\![P]\!]\right)}$$ ⚠

$$[\![P+Q]\!]_{\neq 0} = [\![P]\!]_{\neq 0} + [\![Q]\!]_{\neq 0}$$ ●

$$[\![P^\star]\!] = \textsc{Seq}\left([\![P]\!]_{\neq 0}\right)$$ ●

- ● Labelled and unlabelled operators in the same grammar;
- ⚠ $[\![P]\!]$, $[\![Q]\!]$ might contain the empty execution;
- ● $[\![P]\!]_{\neq 0} =$ non-empty executions of P.

Outline

A class of concurrent programs

Algorithmic aspects

Conclusion and perspective

Counting executions

Algorithm: $P \xrightarrow{\text{prev. slide}} [\![P]\!] \xrightarrow{\text{symbolic method}} GF \xrightarrow{[z^n]} \text{count}$

Counting executions

Algorithm: $P \xrightarrow{\text{prev. slide}} [\![P]\!] \xrightarrow{\text{symbolic method}} GF \xrightarrow{[z^n]} \text{count}$

$$
\begin{aligned}
\textsc{count}(0) &= 1 \\
\textsc{count}(a) &= z \\
\textsc{count}(P \parallel Q) &= p(z) \circledcirc q(z) \\
\textsc{count}(P; Q) &= p(z) \cdot q(z) \\
\textsc{count}(P + Q) &= p(z) + q(z) - p(0)q(0) \\
\textsc{count}(P^\star) &= (1 - (p(z) - p(0)))^{-1}
\end{aligned}
$$

All operation are taken $\bmod z^{n+1}$

$$
\begin{aligned}
p(z) &= \textsc{count}(P) \\
q(z) &= \textsc{count}(Q)
\end{aligned}
$$

Counting executions

Algorithm: $P \xrightarrow{\text{prev. slide}} [\![P]\!] \xrightarrow{\text{symbolic method}} GF \xrightarrow{[z^n]} \text{count}$

$$
\begin{aligned}
\textsc{count}(0) &= 1 \\
\textsc{count}(a) &= z \\
\textsc{count}(P \parallel Q) &= p(z) \circledcirc q(z) \bullet \\
\textsc{count}(P; Q) &= p(z) \cdot q(z) \\
\textsc{count}(P + Q) &= p(z) + q(z) - p(0)q(0) \\
\textsc{count}(P^\star) &= (1 - (p(z) - p(0)))^{-1}
\end{aligned}
$$

All operation are taken $\bmod z^{n+1}$

$$
\begin{aligned}
p(z) &= \textsc{count}(P) \\
q(z) &= \textsc{count}(Q)
\end{aligned}
$$

- $p(z) \circledcirc q(z) = \mathcal{L}\left(\mathcal{B}\left(p(z)\right) \cdot \mathcal{B}\left(q(z)\right)\right)$
  where $\mathcal{L}\left(\sum_n \frac{a_n}{n!} z^n\right) = \sum_n a_n z^n$ and $\mathcal{B}\left(\sum_n a_n z^n\right) = \sum_n \frac{a_n}{n!} z^n$

Counting executions — complexity

Theorem

The counting algorithm performs $O(|P|M(n))$ arithmetic operations on big integers.

The coefficients of the polynomial have $O(n \ln n)$ bits.

- $|P|$ is the syntactic size of P.
- $M(n)$ is the cost of the multiplication of two polynomials of degree n.

Counting executions — complexity

Theorem

The counting algorithm performs $O(|P|M(n))$ arithmetic operations on big integers.

The coefficients of the polynomial have $O(n \ln n)$ bits.

- $|P|$ is the syntactic size of P.
- $M(n)$ is the cost of the multiplication of two polynomials of degree n.

$\implies$ $O(|P|M(n)M(n \ln n))$ bit-complexity.

Random sampling of executions

Algorithm: $P \xrightarrow{\text{prev. slides}} [\![P]\!] \xrightarrow[\text{[}FZC'93\text{]}]{\text{recursive method}}$ uniform execution

Random sampling of executions

Algorithm: $P \xrightarrow{\text{prev. slides}} [\![P]\!] \xrightarrow[\text{[FZC'93]}]{\text{recursive method}}$ uniform execution

$$\textsc{sample}((a+b)^\star \parallel (c + (d;e) + (f;g)), 3)$$

Rule:

Random sampling of executions

Algorithm: $P \xrightarrow{\text{prev. slides}} [\![P]\!] \xrightarrow[\text{[}FZC'93\text{]}]{\text{recursive method}}$ uniform execution

$$\textsc{sample}\big((a+b)^\star \parallel (c + (d;e) + (f;g)), 3\big)$$

Rule: $P_n = Q_0 R_0 \binom{n}{0} + Q_1 R_{n-1} \binom{n}{1} + Q_2 R_{n-2} \binom{n}{2} + \cdots$
Pick $k \in [\![0;n]\!]$ with probability $Q_k R_{n-k} \binom{n}{k} / P_n$

$$1 \cdot 0 \cdot \binom{3}{0} + 2 \cdot 2 \cdot \binom{3}{1} + 4 \cdot 1 \cdot \binom{3}{2} + 8 \cdot 0 \cdot \binom{3}{3} = 24 \cdot (0 + 1/2 + 1/2 + 0)$$

Random sampling of executions

Algorithm: $P \xrightarrow{\text{prev. slides}} [\![P]\!] \xrightarrow[\text{[FZC'93]}]{\text{recursive method}}$ uniform execution

$$\text{SHUFFLE}(\text{SAMPLE}((a+b)^\star, 1), \text{SAMPLE}((c+(d;e)+(f;g)), 2))$$

Rule:

$$1 \cdot 0 \cdot \binom{3}{0} + 2 \cdot 2 \cdot \binom{3}{1} + 4 \cdot 1 \cdot \binom{3}{2} + 8 \cdot 0 \cdot \binom{3}{3} = 24 \cdot (0 + 1/2 + 1/2 + 0)$$

Random sampling of executions

Algorithm: $P \xrightarrow{\text{prev. slides}} [\![P]\!] \xrightarrow[\text{[FZC'93]}]{\text{recursive method}}$ uniform execution

$$\text{SHUFFLE}(\text{SAMPLE}((a+b)^\star, 1), \ldots)$$

Rule: $P^\star \to 0 + P; P^\star$

$$1 \cdot 0 \cdot \binom{3}{0} + 2 \cdot 2 \cdot \binom{3}{1} + 4 \cdot 1 \cdot \binom{3}{2} + 8 \cdot 0 \cdot \binom{3}{3} = 24 \cdot (0 + 1/2 + 1/2 + 0)$$

Random sampling of executions

Algorithm: $P \xrightarrow{\text{prev. slides}} [\![P]\!] \xrightarrow[\text{[FZC'93]}]{\text{recursive method}}$ uniform execution

$$\textsc{shuffle}(\textsc{sample}(0 + (a + b); (a + b)^\star, 1), \ldots)$$

Rule: $P^\star \to 0 + P; P^\star$

$$1 \cdot 0 \cdot \binom{3}{0} + 2 \cdot 2 \cdot \binom{3}{1} + 4 \cdot 1 \cdot \binom{3}{2} + 8 \cdot 0 \cdot \binom{3}{3} = 24 \cdot (0 + 1/2 + 1/2 + 0)$$

Random sampling of executions

Algorithm: $P \xrightarrow{\text{prev. slides}} [\![P]\!] \xrightarrow[\text{[FZC'93]}]{\text{recursive method}}$ uniform execution

---

$$\textsc{shuffle}(\textsc{sample}((a+b);(a+b)^\star,1),\ldots)$$

Rule:

$$1\cdot 0\cdot \binom{3}{0} + 2\cdot 2\cdot \binom{3}{1} + 4\cdot 1\cdot \binom{3}{2} + 8\cdot 0\cdot \binom{3}{3} = 24\cdot(0+1/2+1/2+0)$$

Random sampling of executions

Algorithm: $P \xrightarrow{\text{prev. slides}} [\![P]\!] \xrightarrow[\text{[FZC'93]}]{\text{recursive method}}$ uniform execution

$$\textsc{shuffle}(\textsc{sample}((a+b);(a+b)^\star,1),\ldots)$$

Rule: $P_n = Q_0R_0 + Q_1R_{n-1} + Q_2R_{n-2} + \cdots$

Pick $k \in [\![0;n]\!]$ with probability Q_kR_{n-k}/P_n

$$1\cdot 0\cdot\binom{3}{0} + 2\cdot 2\cdot\binom{3}{1} + 4\cdot 1\cdot\binom{3}{2} + 8\cdot 0\cdot\binom{3}{3} = 24\cdot(0+1/2+1/2+0)$$

$$0\cdot 1 + 2\cdot 1 = 2\cdot(0+1)$$

Random sampling of executions

Algorithm: $P \xrightarrow{\text{prev. slides}} [\![P]\!] \xrightarrow[\text{[FZC'93]}]{\text{recursive method}}$ uniform execution

$$\textsc{shuffle}(\textsc{sample}(a+b,1),\ldots)$$

Rule:

$$1 \cdot 0 \cdot \binom{3}{0} + 2 \cdot 2 \cdot \binom{3}{1} + 4 \cdot 1 \cdot \binom{3}{2} + 8 \cdot 0 \cdot \binom{3}{3} = 24 \cdot (0 + 1/2 + 1/2 + 0)$$

$$0 \cdot 1 + 2 \cdot 1 = 2 \cdot (0 + 1)$$

Random sampling of executions

Algorithm: $P \xrightarrow{\text{prev. slides}} [\![P]\!] \xrightarrow[\text{[FZC'93]}]{\text{recursive method}}$ uniform execution

$$\textsc{shuffle}(\textsc{sample}(a + b, 1), \ldots)$$

Rule: $P_n = Q_n + R_n$
Choose Q with probability Q_n/P_n

$$1 \cdot 0 \cdot \binom{3}{0} + 2 \cdot 2 \cdot \binom{3}{1} + 4 \cdot 1 \cdot \binom{3}{2} + 8 \cdot 0 \cdot \binom{3}{3} = 24 \cdot (0 + 1/2 + 1/2 + 0)$$

$$0 \cdot 1 + 2 \cdot 1 = 2 \cdot (0 + 1)$$

$$1 + 1 = 2 \cdot (1/2 + 1/2)$$

Random sampling of executions

Algorithm: $P \xrightarrow{\text{prev. slides}} [\![P]\!] \xrightarrow[\text{[}FZC'93\text{]}]{\text{recursive method}}$ uniform execution

---

$\textsc{shuffle}(a, \ldots)$

Rule:

$$1 \cdot 0 \cdot \binom{3}{0} + 2 \cdot 2 \cdot \binom{3}{1} + 4 \cdot 1 \cdot \binom{3}{2} + 8 \cdot 0 \cdot \binom{3}{3} = 24 \cdot (0 + 1/2 + 1/2 + 0)$$

$$0 \cdot 1 + 2 \cdot 1 = 2 \cdot (0 + 1)$$

$$1 + 1 = 2 \cdot (1/2 + 1/2)$$

Random sampling of executions

Algorithm: $P \xrightarrow{\text{prev. slides}} [\![P]\!] \xrightarrow[\text{[FZC'93]}]{\text{recursive method}}$ uniform execution

$$\text{SHUFFLE}(a, \text{SAMPLE}((c + (d; e) + (f; g)), 2))$$

Rule:

$$1 \cdot 0 \cdot \binom{3}{0} + 2 \cdot 2 \cdot \binom{3}{1} + 4 \cdot 1 \cdot \binom{3}{2} + 8 \cdot 0 \cdot \binom{3}{3} = 24 \cdot (0 + 1/2 + 1/2 + 0)$$

$$0 \cdot 1 + 2 \cdot 1 = 2 \cdot (0 + 1)$$

$$1 + 1 = 2 \cdot (1/2 + 1/2)$$

Random sampling of executions

Algorithm: $P \xrightarrow{\text{prev. slides}} [\![P]\!] \xrightarrow[\text{[FZC'93]}]{\text{recursive method}}$ uniform execution

---

$$\textsc{shuffle}(a, \textsc{sample}((c + (d; e) + (f; g)), 2))$$

Rule: $P_n = Q_n + R_n + S_n$
Choose Q (or R) with probability Q_n/P_n (or R_n/P_n)

$$1 \cdot 0 \cdot \binom{3}{0} + 2 \cdot 2 \cdot \binom{3}{1} + 4 \cdot 1 \cdot \binom{3}{2} + 8 \cdot 0 \cdot \binom{3}{3} = 24 \cdot (0 + 1/2 + 1/2 + 0)$$

$$0 \cdot 1 + 2 \cdot 1 = 2 \cdot (0 + 1)$$

$$1 + 1 = 2 \cdot (1/2 + 1/2)$$

$$0 + 1 + 1 = 2$$

Random sampling of executions

Algorithm: $P \xrightarrow{\text{prev. slides}} [\![P]\!] \xrightarrow[\text{[FZC'93]}]{\text{recursive method}}$ uniform execution

$$\text{SHUFFLE}(a, \text{SAMPLE}((d; e), 2))$$

Rule:

$$1 \cdot 0 \cdot \binom{3}{0} + 2 \cdot 2 \cdot \binom{3}{1} + 4 \cdot 1 \cdot \binom{3}{2} + 8 \cdot 0 \cdot \binom{3}{3} = 24 \cdot (0 + 1/2 + 1/2 + 0)$$

$$0 \cdot 1 + 2 \cdot 1 = 2 \cdot (0 + 1)$$

$$1 + 1 = 2 \cdot (1/2 + 1/2)$$

$$0 + 1 + 1 = 2$$

Random sampling of executions

Algorithm: $P \xrightarrow{\text{prev. slides}} [\![P]\!] \xrightarrow[\text{[FZC'93]}]{\text{recursive method}}$ uniform execution

$\textsc{shuffle}(a, de)$

Rule:

$$1 \cdot 0 \cdot \binom{3}{0} + 2 \cdot 2 \cdot \binom{3}{1} + 4 \cdot 1 \cdot \binom{3}{2} + 8 \cdot 0 \cdot \binom{3}{3} = 24 \cdot (0 + 1/2 + 1/2 + 0)$$

$$0 \cdot 1 + 2 \cdot 1 = 2 \cdot (0 + 1)$$

$$1 + 1 = 2 \cdot (1/2 + 1/2)$$

0 + 1 + 1 = 2

Random sampling of executions

Algorithm: $P \xrightarrow{\text{prev. slides}} [\![P]\!] \xrightarrow[\text{[}FZC'93\text{]}]{\text{recursive method}}$ uniform execution

$$\frac{}{dae}$$

Rule:

$$1 \cdot 0 \cdot \binom{3}{0} + 2 \cdot 2 \cdot \binom{3}{1} + 4 \cdot 1 \cdot \binom{3}{2} + 8 \cdot 0 \cdot \binom{3}{3} = 24 \cdot (0 + 1/2 + 1/2 + 0)$$

$$0 \cdot 1 + 2 \cdot 1 = 2 \cdot (0 + 1)$$

$$1 + 1 = 2 \cdot (1/2 + 1/2)$$

$$0 + 1 + 1 = 2$$

Random sampling of executions

$P_n = Q_0 R_n + Q_1 R_{n-1} + Q_2 R_{n-2} + \cdots + Q_n R_0$

How to draw $k \in [\![0; n]\!]$ with probability $Q_k R_{n-k} / P_n$?

Random sampling of executions

$$P_n = Q_0 R_n + Q_1 R_{n-1} + Q_2 R_{n-2} + \cdots + Q_n R_0$$

How to draw $k \in [\![0; n]\!]$ with probability $Q_k R_{n-k} / P_n$?

Solution 1:

Draw $x \sim \textsc{Unif}([\![0; P_n[\![)$ and take the minimum k such that $x < Q_0 R_n + Q_1 R_{n-1} + \cdots + Q_k R_{n-k}$.

Random sampling of executions

$P_n = Q_0 R_n + Q_1 R_{n-1} + Q_2 R_{n-2} + \cdots + Q_n R_0$

How to draw $k \in [\![0; n]\!]$ with probability $Q_k R_{n-k} / P_n$?

Solution 1:

Draw $x \sim \textsc{Unif}([\![0; P_n[\![)$ and take the minimum k such that $x < Q_0 R_n + Q_1 R_{n-1} + \cdots + Q_k R_{n-k}$.

Solution 2 (boustrophedonic alg., [FZC'93,Molinero'05]):

Draw $x \sim \textsc{Unif}([\![0; P_n[\![)$ and take the minimum k such that $x < Q_0 R_n + Q_n R_0 + Q_1 R_{n-1} + Q_{n-1} R_1 + Q_2 R_{n-2} + \ldots$ (k terms).

Random sampling of executions — complexity

Theorem [FZC'93,Molinero'05]

The recursive method with solution 2 has complexity $O(n \ln n)$.

Theorem [Molinero'05]

The recursive method has complexity $O(n)$ on recursion-free specifications.

Random sampling of executions — complexity

Theorem [FZC'93,Molinero'05]

The recursive method with solution 2 has complexity $O(n \ln n)$.

Theorem [Molinero'05]

The recursive method has complexity $O(n)$ on recursion-free specifications.

The constant hidden in the O depends on the specification!

Random sampling of executions — complexity

Theorem [FZC'93,Molinero'05]

The recursive method with solution 2 has complexity $O(n \ln n)$.

Theorem (updated)

The recursive method has complexity $O(h \cdot n)$ on recursion-free specification, where h is the number of nested operators of the spec.

The constant hidden in the O depends on the specification!

Random sampling of executions — complexity

Theorem

Random sampling of executions has complexity $O(n \cdot \min(h(P), \ln n))$ where h denotes the "height" of P i.e. its maximum number of nested constructors.

Outline

Conclusion

Take away:

- Specifications can be used as first-class objects

Conclusion

Take away:

- Specifications can be used as first-class objects

Future work:

- Generalize the model
- Analytic properties of the OGF of $[\![P]\!]$?
- Statistical model-checking

Thanks for your attention