

Zaionc paradox revisited

Pierre Lescanne

▶ **To cite this version:**

Pierre Lescanne. Zaionc paradox revisited. 2022. hal-03197423v6

HAL Id: hal-03197423
https://hal.science/hal-03197423v6

Preprint submitted on 21 Mar 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ALMOST ALL CLASSICAL THEOREMS ARE INTUITIONISTIC

PIERRE LESCANNE

Abstract. Canonical expressions are representative of implicative propositions (i.e., propositions with only implications) up-to renaming of variables. Using a Monte-Carlo approach, we explore the model of canonical expressions in order to confirm the paradox that says that *asymptotically almost all classical theorems are intuitionistic*. Actually we found that more than $96,6\%$ of classical theorems are intuitionistic among propositions of size 100.

AMS Classification numbers: 11B73, 03F55, 06E30, 05-04, 05-08

Keywords: intuitionistic logic, classical logic, combinatorics, asymptotic, random generation, Bell number, Catalan number, Monte-Carlo method

1. Introduction

In 2007, Marek Zaionc coauthored two papers [8, 6], corresponding to two models of the calculus of implicative propositions and presenting the following paradox, namely that *asymptotically almost all classical theorems are intuitionistic*, which we call, in short, *Zaionc paradox*. This says that when the size of the propositions grows, the ratio of the number of intuitionistic theorems over the number of classical theorems goes up to one. In the current paper, we focus on the model of [8], which we call *canonical expressions*. They have been introduced by Genitrini, Kozik and Zaionc [8] and more recently by Tarau and de Paiva [16, 17]. A canonical expression is a representative of a class of implicative propositions (propositions that contain only implication $\rightarrow$) which differ only by the name assigned to the variables. Whereas Genitrini, Kozik and Zaionc addressed the mathematical aspect of this model, Tarau and de Paiva tried to explicitly generate all the canonical expressions of a given size and faced up to combinatorial explosion, because canonical expressions grow super exponentially in size when the number on variables increases. In this paper, I check experimentally Zaionc paradox, adopting a Monte-Carlo approach to observe how this paradox emerges. Indeed I designed a linear algorithm to randomly generate canonical expressions. Therefore I can consider large samples of random (for a uniform distribution) canonical expressions and count how many canonical expressions in that samples are intuitionistic theorems or classical theorems. The experiments, centered around canonical expressions of size 100, show that the numbers we get for both sets are very close confirming experimentally the paradox, with a ratio 96.6% much better than this obtained on the model of Genitrini et al. which yields 36% for canonical expressions of size 100. As a by product we obtain programs generating large random canonical expressions, large random intuitionistic theorems or large random classical theorems.

The programs used in this paper can be found on GitHub.

2. Intuitionistic vs classical theorems

In this paper we deal only with implicative propositions. An implicative proposition is a binary expression with propositional variables, which has only one binary operator namely the implication written $\rightarrow$. This can be seen as the type of a function in functional programming or in λ-calculus [1, 12]. Among the implicative propositions, some can be proven, using a proof system, the so called *natural deduction* [9]. Let us consider natural deduction. There are three rules used to prove *intuitionistic theorems*.

$$\mathsf{Axiom}\ \frac{}{\alpha \vdash \alpha} \qquad \rightarrow\text{-}\mathsf{Elim}\ \frac{\Gamma \vdash \alpha \rightarrow \beta \qquad \Gamma \vdash \alpha}{\Gamma \vdash \beta} \qquad \rightarrow\text{-}\mathsf{Intro}\ \frac{\alpha, \Gamma \vdash \beta}{\Gamma \vdash \alpha \rightarrow \beta}$$

Classical theorems are proved by adding the axiom:

$$\mathsf{Pierce}\ \frac{}{\vdash ((\alpha \rightarrow \beta) \rightarrow \alpha) \rightarrow \alpha}$$

which is called *Peirce law*. Usually one uses valuations, which assign booleans to variables. Let ρ be an assignment of booleans to variables. Valuations of expressions are defined by:

$$\begin{array}{rcl} [\![x]\!]_\rho & = & \rho(x) \\ [\![e \rightarrow e']\!]_\rho & = & [\![e']\!]_\rho \vee \overline{[\![e]\!]_\rho} \end{array}$$

where $b \mapsto \overline{b}$ is the negation and $b_1 \vee b_2 = 1$ except when $b_1 = b_2 = 0$ is *or*. An expression e is a classical theorem or a tautology, if for all valuation ρ, $[\![e]\!]_\rho = \mathsf{True}$.

Notice that, with the Curry Howard isomorphism [1, 12], the results of this paper apply also to types.

3. The model of canonical expressions

We call *canonical expression* the representative of an equivalence class of binary expressions up-to renaming of variables. In other words, a canonical expression can be seen as a binary expression, in which variables are named canonically, from right to left. That means that the rightmost variable is x_0, then if processing to the left, the next new variable is x_1, then the next new variable, which is neither x_0 nor x_1 is x_2 etc. Recall that in an expression, a variable corresponds to a position into the expression. In other words a variable in an equivalence class of positions. Therefore naming canonically a variable corresponds to naming canonically an equivalence class in the set of position. Therefore if a variable belongs to the i^{th} class it will be named α_i and vice-versa, if a class is the class of α_i, it is the i^{th} class. In canonical expressions, the classes are numbered from right to left. For instance, assume an expression of size 10, i.e., with 10 occurrences of variables. This is an expression with 10 positions of variables, like :

$$x\ y\ y\ x\ y\ x\ z\ x\ x\ x$$

or

$$\beta\ \alpha\ \alpha\ \beta\ \alpha\ \beta\ \gamma\ \beta\ \beta\ \beta$$

In the first expression we see 3 variables namely $\{x, y, z\}$, hence 3 congruence classes. As said above, for technical reasons, not hard to guess, variables are numbered from right to left, starting at 0. Hence x which corresponds to positions $\{1, 4, 6, 8, 9, 10\}$ is class 0, z which corresponds to positions $\{7\}$ is class 1 and y which corresponds

to positions $\{2, 3, 5\}$ is class 2. Therefore the list of canonically named variables, associated with the above list of variables is.

$$\alpha_0 \ \alpha_2 \ \alpha_2 \ \alpha_0 \ \alpha_2 \ \alpha_0 \ \alpha_1 \ \alpha_0 \ \alpha_0 \ \alpha_0$$

The above congruence class is canonically represented by the string 0220201000. Canonical presentation of congruence classes over a set of n elements by a string of natural numbers of size n is known. It is called *restricted growth string* by Knuth ([10], fascicle 3, §7.2.1.5, p. 62) and *irregular staircase* by Flajolet and Sedgewick [5] (p. 62-63). In this paper, we consider classes from right to left wherever the cited authors consider them from left to right, but this is a detail. Intuitively, a restricted growth string is a string whose last item is 0 and when one progresses to the left, one meets items that have been met already or if not the new item is just the successor of the largest item met until this point. Here $[0..i]^*$ is the set of strings made of integers k such that $0 \leq k \leq i$.

Definition 1 (Restricted growth string)**.** *The set* $\mathcal{W}_n$ *of* n-restricted right to left growth strings *is defined as follows*

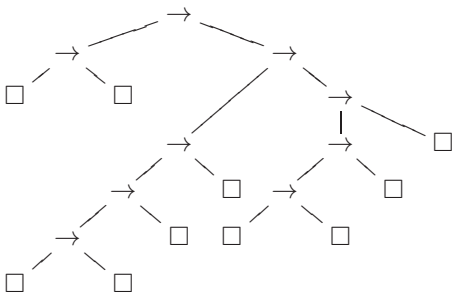
- $\mathcal{W}_0 = [0..0]^*$,
- $\mathcal{W}_{n+1} = [0..(n+1)]^* \, (n+1) \, \mathcal{W}_n$

For instance $W_2 = [0..2]^* \, 2 \, \mathcal{W}_1 = [0..2]^* \, 2 \, [0..1]^* \, 1 \, \mathcal{W}_0 = [0..2]^* \, 2 \, [0..1]^* \, 1 \, [0..0]^*$ One sees that $0220201000 \in \mathcal{W}_2$ where items larger that those on the right are put in brown.

Once the variables are chosen, how operators $\to$ are associated has to be done. Here we are interested in parenthesized expressions with the only binary operator $\to$. For instance, for an expression of size 10, we look for a binary tree with 10 external leaves like:

$$((\square\to\square)\to((((\square\to\square)\to\square)\to\square)\to(((\square\to\square)\to\square)\to\square)))$$

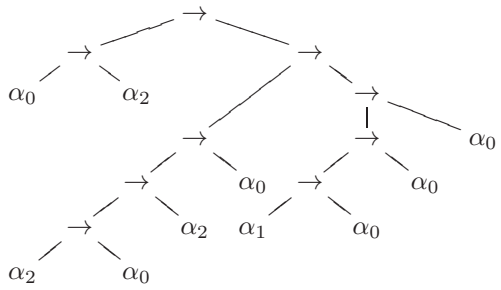
which can be drawn as the tree :



To get a canonical expression one matches a restricted growth string and a binary tree. In our case, we get by matching the above restricted tree and the above parenthesized expression, the following canonical expression

$$((\alpha_0 \to \alpha_2) \to ((((\alpha_2 \to \alpha_0) \to \alpha_2) \to \alpha_0) \to (((\alpha_1 \to \alpha_0) \to \alpha_0) \to \alpha_0)))$$

which corresponds to the tree:



Canonical expressions are therefore pairs of binary trees and restricted left to right growth strings, counted by $K_n = C_{n-1}\varpi_n$ where C_n are *Catalan numbers* (counting binary trees) and ϖ_n are *Bell numbers* (counting restricted growth strings). This corresponds to sequence A289679 in the *Online encyclopedia of integer sequences* [14]. Asymptotically,

$$C_{n-1} \sim \frac{4^{n-1}}{\sqrt{\pi(n-1)^3}}$$

$$\varpi_n \sim n! \frac{e^{e^r-1}}{r^n\sqrt{2\pi r(r+1)e^r}}$$

where $r \equiv r(n)$ is the positive root of the equation $re^r = n+1$. Therefore

$$K_n \quad \sim \quad n! \frac{4^{n-1}e^{e^r-1}}{\pi\sqrt{2(n-1)^3 r(r+1)e^r}}$$

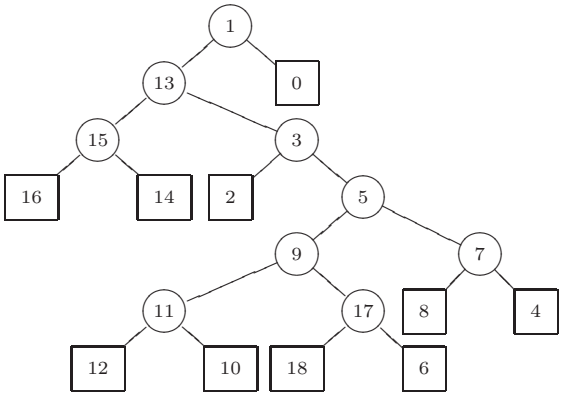
The first values of K_n are $1, 2, 10, 75, 728, 8526, 115764, 1776060, 30240210, ...$ whereas $K_{100} \sim 9.62\ 10^{168}$ and $K_{400} \sim 1.51\ 10^{880}$.

4. Random canonical expressions

Since canonical expressions are pairs of well-known combinatorial objects, namely binary trees and congruence classes, we can use well-known algorithm to generate each constituents of the pairs.

4.1. **Random binary trees.** For generating random binary trees, I use *Rémy algorithm* [13] which is linear. This algorithm is described by Knuth in [10] § 7.2.1.6 (pp. 18-19). I have taken his implementation. The idea of the algorithm is that a random binary tree can be built by iteratively and randomly picking an internal node or a leaf in a random binary tree and inserting a new internal node and a new leaf either on the left or on the right. A binary tree of size n has $n-1$ internal nodes and n leaves. Inserting a node in a binary tree of size n requires throwing randomly a number between 1 and $4n-2$ (a random number between 0 and $4n-3$ in my Haskell implementation). This process can be optimized by representing a binary tree as a list (a vector in Haskell), an idea sketched by Rémy and described by Knuth. In this vector, even locations are for internal nodes and odd locations are for leaves. Here is a vector representing a binary tree with 10 leaves and its drawing.

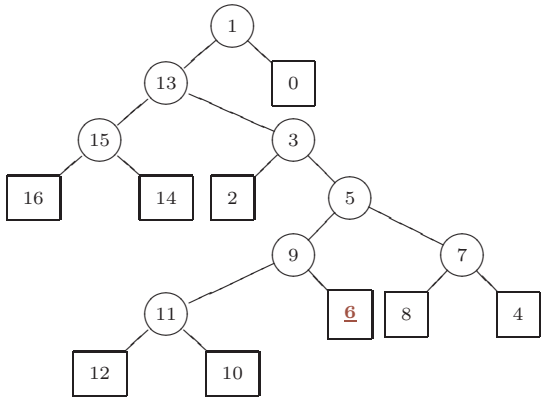
indices	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
values	1	13	0	2	5	9	7	8	4	11	17	12	10	15	3	16	14	18	6



This tree was built by inserting the node 17 together with the leaf 18 in the following tree.

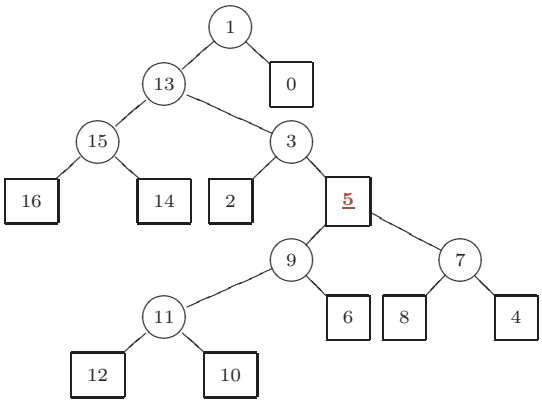
indices	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
values	1	13	0	2	5	9	7	8	4	11	6	12	10	15	3	16	14

coding the tree

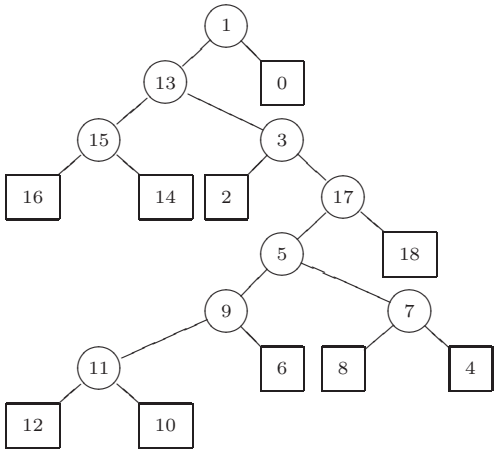


This was done by picking a node (internal node or leaf, here the node with label 6) and a direction (here right) and by inserting above this node a new internal node (labeled 17) and, below the new inserted internal node, a new leaf of the left (labeled 18). This double action (inserting the internal node and attaching the leaf) is done by choosing a number in the interval [0..33] (in general, in the interval $[0..(4n-3)]$). Assume that in this case the random generator returns 21. 21 contains two informations : its parity (a boolean) and its half. Half of 21 is 10, which tells that the new node 17 must be inserted above the 10^{th} (in the array) node namely 6. Since 21 is odd, the rest of the tree (here reduced to the leaf 6) is inserted on the right (otherwise it would be inserted on the left). A new leaf 18 is inserted on the left (otherwise it would be inserted on the right).

Consider same tree and suppose that the random value is 8. Half of 8 is 4. Hence the new leaves are inserted above the node labeled by 5



and since 8 is even the rest of tree is inserted on the left and a new leaf (labeled 18) is inserted on the right.



The algorithm works as follows. If $n = 0$, Rémy's algorithm returns the vector starting at 0 and filled with anything, since the whole algorithm works on the same vector with the same size. In general, say that, for $n-1$, Rémy's algorithm returns a vector v. One picks a random integer x between 0 and $4n-3$. Let k be half of x. In the vector v one replaces the k^{th} position by $2n-1$ and one appends two elements, namely the k^{th} item of v followed by $2n$ if x is even and $2n$ followed by the k^{th} item of v if x is odd.

If we admit that given a `seed` and a positive integer `n`, `randForRemy seed n` returns a random integer between 0 and $4n-3$ inclusive, the program in Haskell of the function `rbtV` which yields a random binary tree of size n coded as a vector of length $2n$ is given in Fig. 1.

4.2. **Random restricted growth string.** For generating random partitions or random restricted growth strings an algorithm due to A. J. Stam [15] and described by Knuth in [11] § 7.2.1.3 (p. 74) was implemented. The implementation requires, for each value of n (the size of the underlying set –for us, this is the number of variables or the size of the expression–), a preliminary construction of a table of

```
rbtV :: Int -> Int -> Vector Int
rbtV seed 0 = Data.Vector.replicate sizeOfVector (-1) // [(0,0)]
rbtV seed n =
  let x = randForRemy seed n -- a random value between 0 and 4n-3 inclusive
      v = rbtV seed (n-1)
      k = x `div` 2
  in case even x of
    True -> v // [(k,2*n-1),(2*n-1,v!k),(2*n,2*n)]
    False -> v // [(k,2*n-1),(2*n-1,2*n),(2*n,v!k)]
```

FIGURE 1. Haskell program for Rémy's algorithm

reals in which indices are looked up (the number M of classes). Those reals are probabilities

$$p_n = \frac{m^n}{em!\varpi_n}$$

that an n-partition has m classes. Thus in my program, I implemented the algorithm for size $n = 10$, 25, 50 100, 500 and 1000. In order to get accurate values, the p_n's for those integers were computed elsewhere in a dedicated computer algebra software namely Sagemath [18]. From this table and a randomly chosen number between 0 and 1, one gets a random number M of equivalence classes. Thereafter, for each element in $[0..n]$ one picks up randomly uniformly and independently individuals in $[0..(M-1)]$. This method yields *class descriptions* (classes are a priori numbered from 0 to $M-1$ and the elements 0,..., $n-1$ are distributed in those classes), but one wants *restricted growth strings* as described in Section 3. So a function that transforms a class description into a restricted growth string was implemented.

Putting together those two algorithms, namely binary tree random generation and restricted growth string random generation, produces an algorithm for canonical expression random generation.

5. Selecting intuitionistic theorems

Once a canonical expression is randomly generated, one has to check whether it is an intuitionistic theorem, a classical theorem, or not a proposition of those sorts

The program selects two kinds of trivial intuitionistic expressions. At first glance this selection looks coarse, but from experience, the first one (*simple* theorems) collects a large majority of the expressions and the second selects (*arrowElim* theorems) most of the others, because it is associated with a trick which consists in cleaning expressions by removing recursively trivial subexpressions that are theorems. Indeed a "cleaned" sub-expression can become trivial and be removed in turn. This might allow cleaning an expression where a trivial premise appears, which might be removed in turn. Therefore this section lists six methods for selecting more and more intuitionistic theorems. Except "simple", these adjectives are mine.

5.1. **Simple intuitionistic theorems.** A *simple intuitionistic theorem* (see [8] Definition 1) is a theorem, in which the goal is among the premises. In other

words, this is a theorem of the form:

$$... \to \alpha_i \to ... \to \alpha_i$$

5.2. **MP intuitionistic theorems.** Let us call *MP intuitionistic theorem* (for *modus ponens* theorem), a theorem which is a direct application of the modus ponens. This is a theorem with goal α_i and two premises α_j and $\alpha_j \to \alpha_i$. Therefore it has the form:

$$... \to (\alpha_j \to \alpha_i) \to ... \to \alpha_j \to ... \to \alpha_i$$

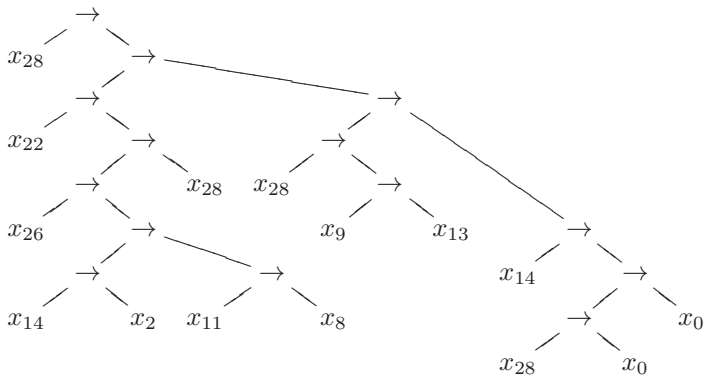
or

$$... \to \alpha_j \to ... \to (\alpha_j \to \alpha_i) \to ... \to \alpha_i$$

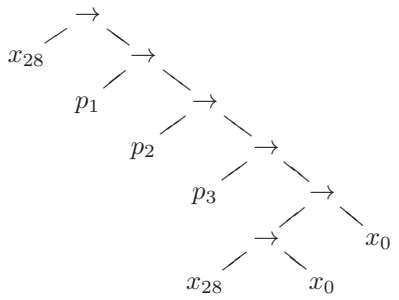
During the experiments I met, for instance, the term, which is not a canonical expression, but obtained by cleaning:

$$(x_{28} \to ((x_{22} \to ((x_{26} \to ((x_{14} \to x_2) \to (x_{11} \to x_8))) \to x_{28})) \to ((x_{28} \to (x_9 \to x_{13})) \to (x_{14} \to ((x_{28} \to x_{0)} \to x_0)))))$$

which can be drawn as the labeled binary tree:



which can be written



It is clearly an intuitionistic theorem and isMP checks it.

5.3. **Easy intuitionistic theorems.** Let us call *easy intuitionistic theorems*, expressions that are *simple* or *mp*.

5.4. **Removing easy premises.** In intuitionistic logic if a premise is a theorem, it can be removed. Consider the predicate $\vdash p$ that says that p is a theorem. Clearly under the assumption $\vdash p$, the two statements $\vdash p \to q$ and $\vdash q$ are equivalent. Note that p is not necessarily the first premise of the implication. Hence if an expression becomes *easy after removing easy premises*, it is an intuitionistic theorem.

In the process of "cleaning" expressions, expressions that are *easy* are removed inside-out. This way *easy* expressions that can be removed recursively are detected.

5.5. **Minor intuitionistic theorems.** A *minor* theorem is a theorem of the form $... \to p \to ... \to p$, whatever p is. *Simple* propositions are *minor*, but *minor* propositions are not always *simple*. For instance, $x \to (y \to z) \to y \to z$ is minor, but is not *simple*. Detecting such expressions has a cost, I decided to not detect *minor* intuitionistic theorem recursively, but only after *easy* subexpressions have been removed recursively.

5.6. **Cheap intuitionistic theorems.** Let us call *cheap intuitionistic theorems*, expressions that are *minor* or *easy* after removing (recursively) *easy* premises. Actually, my experiments lead naturally to the statement that 96.6% of classical theorems with 100 variables are *cheap* intuitionistic (see Section 7).

6. Classical tautologies

The selection of classical tautologies is by valuations. If all the valuations of an expression yield True this expression is a classical tautology. But this method is obviously intractable [4]. It should be applied only to expressions on which other more efficient methods do not work and with a limitation on the number of variables in expressions

6.1. **Simple antilogies.** Trivial non classical propositional theorems are eliminated before applying valuations. The predicate simpAntilogy finds in quadratic time a large set of propositions which are not tautologies and which we call *simple antilogies*. Thereafter, boolean valuations are checked only on the positions that are not *simple antilogies*. For more efficiency, the predicate simpAntilogy is applied on expressions in which easy premises have been recursively removed, like for intuitionistic expressions.

An expression is a *simple antilogy* if it is of the form $... \to e_i \to ... \to x_0$ where the premises e_i are of one of the following forms:

(i) $\to ... \to x_i$ with $x_i \neq x_0$ i.e., with a goal which is not x_0
(ii) $... \to x_0 \to ... \to x_0$ i.e., are *simple* with goal x_0.

One sees easily that applying the valuation ρ such that $\rho(x_0) =$ False and $\rho(x_i) =$ True for $i \neq 0$ to *simple antilogies* yields $False$. Therefore *simple antilogies* are not classical theorems.

In [8], Genitrini, Kozik and Zaionc consider only the first case, namely the case where the premises have a goal which is note x_0. They call such expressions, *simple non tautologies*.

6.2. **Expressions with too many variables.** Assume we recursively remove *simple antilogies*, there are still expressions intractable by the valuation method, because they have too many variables, i.e., they have a too large index. In my experiment with expressions of size 100, an index is too large if it is larger than 31. Fortunately those expressions are rare and one may expect that there is a valuation

that rejects them. For this, I rename all the too large indices as they would be the same as the bound. The valuations are checked on this renamed expression. If the renamed expression is not a tautology, then the given expression is not a tautology. In the experiment of Section 7 this trick works and eliminates expressions with too large indices which need not to be checked further.

7. Results

7.1. **Ratio cheap vs classical.** My Haskell program was run on a sample of 20 000 randomly generated canonical expressions of size 100 and I found 759 classical tautologies, among which 733 were cheap expressions, hence guaranteed to be intuitionistic theorems. Therefore the ratio of cheap theorems over classical theorems is 96.6%. Said otherwise, less that 3.4% of the classical theorems are not cheap, i.e., are not intuitionistic. Are these 3.4% classical non cheap theorems still intuitionistic? The experience cannot tell. I presume that there are likely more than 733 intuitionistic theorems and therefore among propositions of size 100, more than 96.6% of classical theorems that are intuitionistic, or less that 3.4% of intuitionistic classical propositions that are not inuitionistic.

7.2. **Simple intuitionistic theorems vs not simple non tautologies.** In [8], Genitrini, Kozik and Zaionc take the ratio of the number of simple intuitionistic theorems over the number of non simple non tautologies as the quantity that goes to 1 and is a lower bound of the ratio of the number of intuitionistic theorems over the number of classical theorems. Among 10 000 random canonical expressions of size 100, I found 238 simple intuitionistic theorems and 685 non simple non tautologies, for a ratio closed to 36%, a ratio largely smaller than the above one.

7.3. **Simple intuitionistic theorems.** Besides, another number of interest is the ratio R_n of simple intuitionistic theorems over all canonical expressions of size n. In the next array, this is compared with the formula $\frac{log(n)}{n}$.

n	$\frac{log(n)}{n}$	R_n
25	$0,128755033$	0.2214
50	$0,07824046$	0.1248
100	$0,046051702$	0.0506
500	$0,012429216$	0.0119
1000	$0,006907755$	0.006

Genitrini, Kozik and Zaionc [8] gave $\frac{e\ log(n)}{n}$ in Lemma 2, for the same quantity, but after viewing my results Genitrini [7] found a mistake and corrected the formula to $\frac{log(n)}{n}$, which now corresponds to what I found. Notice that this does not affect their other results.

Acknowledgments. I thank Valeria De Paiva for an interesting interaction and the incentive to address this problem, Jean-Luc Rémy for discussions on binary tree generation and Antoine Genitrini for discussions on Zaionc paradox.

8. Conclusion

Algorithms for random generation presented in *The Art of Computer Programming* [10, 11] allow implementing Monte-Carlo methods that confirm experimentally Zaionc paradox and show that the convergence (as the size of the expressions grows) of the set of intuitionistic theorems toward this of classical theorems is faster than expected from the asymptotic approximations proposed by the analytic combinatorial theory [8]. Indeed, whereas I compare the set of cheap intuitionistic theorems (Section 5.6) with this of classical theorems, Genitrini, Kozik and Zaionc compare the set of simple intuitionistic theorems (see Section 5.1) with the set of non simple non tautologies (Section 6.1). This is a too rough approximation and this suggests to complete the analytic development to justify this faster convergence.

Notice that Tarau and de Paiva [17] looked at a phenomenon similar to Zaionc paradox for linear logic. Therefore, it should be interesting to extend my approach to this case. Likewise, it would be interesting to investigate experimentally other models of expressions, for both traditional logic and linear logic. Currently I am exploring expressions made of a binary operator, like $\wedge$, $\vee$ or $\rightarrow$.

It seems that this result on the distribution of propositions has to do with the amazing efficiency of SAT-solvers [3, 2]. The fact that most of the classical theorems can be solved as "cheap" intuitionistic propositions may explain why SAT-solvers are so efficient and the connection should be further investigated. Likely, the remaining true classical propositions contribute to the hardness of SAT for the worst case analysis.

References

[1] Hendrik Pieter Barendregt, Wil Dekkers, and Richard Statman. *Lambda Calculus with Types*. Perspectives in logic. Cambridge University Press, 2013. URL: http://www.cambridge.org/de/academic/subjects/mathematics/logic-categories-and-sets/lambda-calculus-types.

[2] Armin Biere, Marijn J.H. Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS press, 2009.

[3] Curtis Bright, Jürgen Gerhard, Ilias S. Kotsireas, and Vijay Ganesh. Effective problem solving using SAT solvers. In Jürgen Gerhard and Ilias S. Kotsireas, editors, *Maple in Mathematics Education and Research - Third Maple Conference, MC 2019, Waterloo, Ontario, Canada, October 15-17, 2019, Proceedings*, volume 1125 of *Communications in Computer and Information Science*, pages 205–219. Springer, 2019. doi:10.1007/978-3-030-41258-6_15.

[4] Stephen A. Cook. The complexity of theorem-proving procedures. In Michael A. Harrison, Ranan B. Banerji, and Jeffrey D. Ullman, editors, *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*, pages 151–158. ACM, 1971. doi:10.1145/800157.805047.

[5] Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2008.

[6] Hervé Fournier, Danièle Gardy, Antoine Genitrini, and Marek Zaionc. Classical and intuitionistic logic are asymptotically identical. In Jacques Duparc and Thomas A. Henzinger, editors, *CSL*, volume 4646 of *Lecture Notes in Computer Science*, pages 177–193. Springer, 2007.

[7] Antoine Genitrini. Erratum for the paper *Intuitionistic vs Classical Tautologies, Quantitative Comparisons*. Personal communication, April 2021.

[8] Antoine Genitrini, Jakub Kozik, and Marek Zaionc. Intuitionistic vs. classical tautologies, quantitative comparison. In Marino Miculan, Ivan Scagnetto, and Furio Honsell, editors, *Types for Proofs and Programs, International Conference, TYPES 2007, Cividale del Friuli, Italy, May 2-5, 2007, Revised Selected Papers*, volume 4941 of *Lecture Notes in Computer Science*, pages 100–109. Springer, 2007. doi:10.1007/978-3-540-68103-8_7.

[9] G. Gentzen. Untersuchungen über das logische schließen. i. *Mathematische Zeitschrift*, 39:176–210, 1934. URL: http://gdz.sub.uni-goettingen.de/dms/resolveppn/?PPN=GDZPPN002375508.

[10] Donald E. Knuth. *The Art of Computer Programming, Volume 4, Fascicle 3: Generating All Combinations and Partitions*. Addison-Wesley Publishing Company, 2005.

[11] Donald E. Knuth. *The Art of Computer Programming, Volume 4, Fascicle 4: Generating All Trees, History of Combinatorial Generation*. Addison-Wesley Publishing Company, 2006.

[12] Samuel Mimram. *PROGRAM = PROOF*. Independently published, 2020. URL: https://www.amazon.fr/dp/B08C97TD9G/.

[13] Jean-Luc Rémy. Un procédé itératif de dénombrement d'arbres binaires et son application à leur génération aléatoire. *RAIRO Theor. Informatics Appl.*, 19(2):179–195, 1985. doi:10.1051/ita/1985190201791.

[14] N. J. A. Sloane. The on-line encyclopedia of integer sequences. Published electronically at https://oeis.org/, 2021.

[15] A. J. Stam. Generation of a random partition of a finite set by an urn model. *J. Comb. Theory, Ser. A*, 35(2):231–240, 1983. doi:10.1016/0097-3165(83)90009-2.

[16] Paul Tarau. A hiking trip through the orders of magnitude: Deriving efficient generators for closed simply-typed lambda terms and normal forms. In Manuel V. Hermenegildo and Pedro López-García, editors, *Logic-Based Program Synthesis and Transformation - 26th International Symposium, LOPSTR 2016, Edinburgh, UK, September 6-8, 2016, Revised Selected Papers*, volume 10184 of *Lecture Notes in Computer Science*, pages 240–255. Springer, 2016. doi:10.1007/978-3-319-63139-4_14.

[17] Paul Tarau and Valeria de Paiva. Deriving theorems in implicational linear logic, declaratively. In Francesco Ricca, Alessandra Russo, Sergio Greco, Nicola Leone, Alexander Artikis, Gerhard Friedrich, Paul Fodor, Angelika Kimmig, Francesca A. Lisi, Marco Maratea, Alessandra Mileo, and Fabrizio Riguzzi, editors, *Proceedings 36th International Conference on Logic Programming (Technical Communications), ICLP Technical Communications 2020, (Technical Communications) UNICAL, Rende (CS), Italy, 18-24th September 2020*, volume 325 of *EPTCS*, pages 110–123, 2020. doi:10.4204/EPTCS.325.18.

[18] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 7.4)*, 206. https://www.sagemath.org.

École Normale Supérieure de Lyon,, LIP (UMR 5668 CNRS ENS Lyon UCBL),, 46 allée d'Italie, 69364 Lyon, France

Email address: pierre.lescanne@ens-lyon.fr