LE « CINQUIÈME CASTOR AFFAIRÉ » : UNE VICTOIRE COLLECTIVE

Ce programme informatique échappait aux spécialistes depuis plus de soixante ans. Il vient enfin d'être identifié, au terme d'un magnifique travail collaboratif regroupant des enthousiastes de nombreuses nationalités.

L'AUTEUR

JEAN-PAUL DELAHAYE
professeur émérite à l'université de Lille et chercheur au laboratoire Cristal (Centre de recherche en informatique, signal et automatique de Lille)

Tous les records de calculs sont difficiles, mais ils le sont différemment et leur utilité varie beaucoup. Par exemple, pour gagner des bitcoins, émis toutes les dix minutes, il faut résoudre une énigme arithmétique qui nécessite de consommer beaucoup d'électricité pour faire fonctionner des algorithmes assez simples sur des machines très nombreuses. Pas besoin d'un grand volume de mémoire : seule importe la répétition d'un même algorithme, le plus vite possible. Il s'agit d'un travail brutal, parallélisé, mathématiquement sans grand intérêt. Si le calcul produit une erreur, elle sera aisément décelable, car la solution est facile à vérifier une fois trouvée.

Pour calculer le prochain nombre premier record, il faut cette fois-ci coordonner un très grand nombre de machines pour leur faire mener, ensemble, des calculs relativement simples, très longs et peu gourmands en mémoire. Comme dans le cas précédent, les solutions proposées sont faciles à vérifier. Mais ce calcul-ci a l'intérêt de faire progresser la connaissance des nombres premiers, pour le plus grand bonheur de la communauté mathématique.

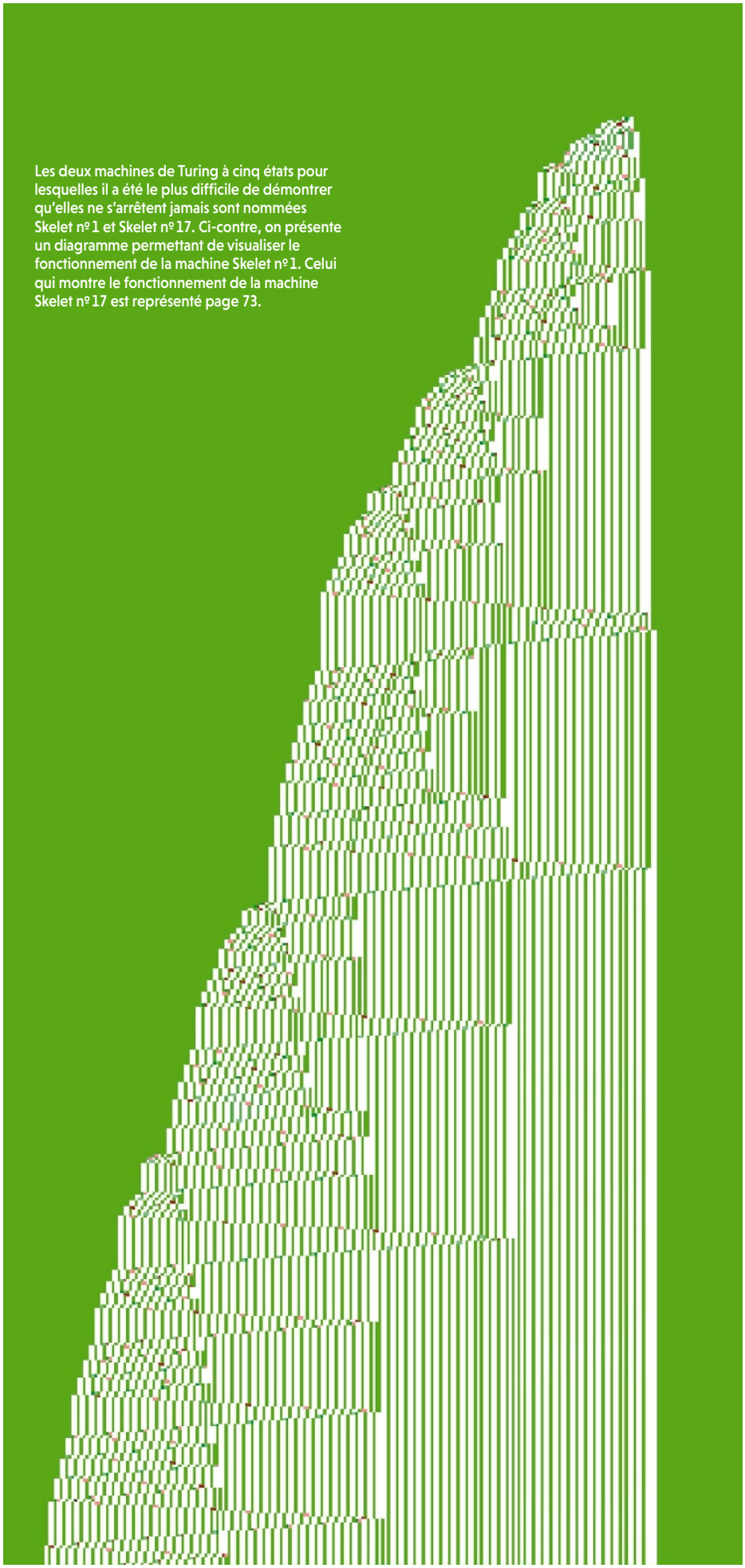
Troisième exemple : pour déterminer une nouvelle décimale du nombre π, il faut disposer localement d'une assez grande quantité de mémoire et réussir à organiser quelques machines pour leur faire effectuer un terrible travail d'endurance. Le résultat obtenu est, ici encore, intéressant pour la recherche en mathématiques. Mais, contrairement aux deux cas précédents, toute erreur dans le processus est grave, car on ne peut pas facilement vérifier le résultat.

Le calcul extraordinaire dont nous allons rendre compte dans cet article, celui du « cinquième castor affairé », est d'une nature à nouveau très différente de ces premiers exemples. Il comporte deux étapes principales. D'abord, une recherche systématique, longue mais peu difficile, consistant à mener divers calculs pouvant être faits indépendamment et en parallèle les uns des autres, et dont il faut mémoriser certains résultats pour aboutir à une proposition plausible. Dans un second temps, bien plus délicat, il faut opérer un travail minutieux de démonstration du résultat suggéré. On verra que les logiciels de validation de démonstration, appelés « assistants de preuve », se sont révélés essentiels dans cette seconde étape. On constatera aussi que cet exploit est le fruit d'un remarquable travail collaboratif entremêlant mathématiques et informatique, et qu'il témoigne d'une nouvelle façon de pratiquer la recherche.

« CASTORS AFFAIRÉS »

La plupart des programmes simples ont deux issues possibles : soit ils se terminent rapidement et renvoient le résultat attendu, soit ils sont piégés dans une boucle de fonctionnement qui a pour conséquence qu'ils ne

Jean-Paul Delahaye a également publié : **Au-delà du Bitcoin** (Dunod, 2022).



Les deux machines de Turing à cinq états pour lesquelles il a été le plus difficile de démontrer qu'elles ne s'arrêtent jamais sont nommées Skelet nº 1 et Skelet nº 17. Ci-contre, on présente un diagramme permettant de visualiser le fonctionnement de la machine Skelet nº 1. Celui qui montre le fonctionnement de la machine Skelet nº 17 est représenté page 73.

s'arrêtent jamais d'eux-mêmes. Il y a cependant quelques exceptions : certains programmes simples tournent très longtemps avant de s'interrompre d'eux-mêmes quand ils ont achevé leurs calculs. On les nomme des « castors », car, comme l'animal, ils s'agitent longuement avant d'atteindre leur but.

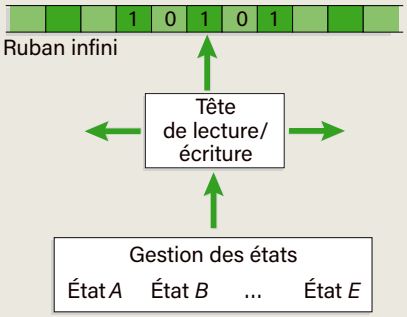
Si l'on fixe un langage de programmation L et une longueur maximum m, il est mathématiquement intéressant de rechercher le programme formulé dans L de longueur m qui calcule le plus longtemps avant de s'arrêter. On dira qu'un tel programme est un « castor affairé » de longueur m pour le langage L.

Le problème qui nous préoccupe concerne le langage de programmation le plus élémentaire, celui des machines de Turing. Elles ont été introduites en 1936 par le désormais célèbre mathématicien britannique Alan Turing, qui a notamment participé au déchiffrage des codes allemands pendant la Seconde Guerre mondiale et ainsi contribué de manière décisive au succès des alliés. La longueur k d'un programme de machine de Turing est donnée par le nombre de règles qu'il comporte (*voir l'encadré 1*). Pour plus de concision, on parlera de « machine de Turing à k règles » ou, de manière équivalente, de « machine de Turing à k états ».

Le mathématicien hongrois Tibor Radó est, en 1962, le premier à attirer l'attention sur le problème qui nous intéresse. Il introduit en particulier la fonction *BB* (pour *busy*

1 MACHINES DE TURING ET « CASTOR AFFAIRÉ »

Une machine de Turing est un dispositif théorique, introduit en 1936 par Alan Turing. C'est le modèle de base utilisé en informatique mathématique, et sa puissance comme calculateur en fait un ordinateur minimal parfait pour aborder les questions de complexité algorithmique.
Une telle machine possède une tête de lecture-écriture (T-L-É) et un ruban infini, découpé en cases, lesquelles sont toutes occupées par des 0 au départ d'un calcul. La T-L-É peut lire une case, et elle peut y écrire en effaçant ce qui y est écrit. La machine



*A*0	*A*1	*B*0	*B*1	*C*0	*C*1	*D*0	*D*1	*E*0	*E*1
1*RB*	1*LC*	1*RC*	1*RB*	1*RD*	0*LE*	1*LA*	1*LD*	*Halt*	0*LA*

déplace sa T-L-É d'une case à la fois, vers la droite ou vers la gauche. La catégorie la plus simple de machines de Turing n'utilise que deux symboles, le 0 et le 1, et c'est uniquement de celles-là dont nous parlons ici.
À chaque instant d'un calcul, la machine est dans un état pris dans une liste finie d'états possibles, qu'on peut noter par exemple *A*, *B*, *C*, *D* et *E* si la machine est à cinq états. Au début d'un calcul, la machine est dans l'état *A*, et la T-L-É est posée n'importe où sur le ruban. Une étape de calcul consiste à lire la case sous la T-L-É puis, selon l'état dans lequel se trouve la machine et ce qui a été lu, à réécrire la case lue, puis à déplacer la T-L-É d'une case vers la droite ou vers la gauche, et enfin à changer d'état.
On présente ci-dessus le tableau de fonctionnement du « castor affairé à cinq états », ce programme qui a fait l'objet d'une grande quête pendant plus de soixante ans. Il calcule pendant *BB*(5) = 47 176 870 étapes, puis il atteint l'état *E* avec un 0 sous la T-L-É et s'arrête alors.
À chaque instant, la règle correspondant à l'état dans lequel se trouve la machine s'exécute. Les deux premières colonnes, (*A*0, 1*RB*) et (*A*1, 1*LC*), décrivent la règle *A*, qui indique ce que fait la machine lorsqu'elle est dans l'état *A*. Les deux colonnes suivantes décrivent la règle *B*, etc. On parle indifféremment de « machine à cinq états » ou de « machine à cinq règles ».
L'interprétation de la règle *A* est :
- (*A*0, 1*RB*) : si la machine est dans l'état *A* et si la T-L-É lit un 0, elle remplace ce 0 par un 1, puis déplace la T-L-É d'une case vers la droite (*R* pour *right*, en anglais) et passe dans l'état *B*.
- (*A*1, 1*LC*) : si la machine est dans l'état *A* et si la T-L-É lit un 1, elle remplace ce 1 par un 1 (donc elle ne change rien) puis déplace la T-L-É d'une case vers la gauche (*L* pour *left*, en anglais) et passe dans l'état *C*.

beaver qui signifie « castor affairé » en anglais) : pour tout entier positif k, $BB(k)$ désigne le nombre maximum d'étapes de calcul que peut exécuter une machine de Turing à k règles, qui utilise seulement deux symboles et qui s'arrête (*voir l'encadré 2*). Tibor Radó démontre que, bien que la fonction soit parfaitement définie, aucun algorithme ou programme d'ordinateur ne peut calculer toutes les valeurs de *BB*. Cela établit qu'il existe des fonctions non calculables – ce qu'on savait déjà, car Turing en avait apporté la preuve dans son article de 1936 –, et fournit une démonstration simple de cet important théorème.

Les premières valeurs de $BB(k)$ sont calculées en 1965 par Tibor Radó et son élève Shen Lin. Ils démontrent à la main que $BB(1)=1$ et $BB(2)=6$, et ils se servent d'un ordinateur pour établir que $BB(3)=21$. La valeur suivante, $BB(4)=107$, n'est obtenue qu'en 1983 par l'américain Allen Brady. Commence alors la longue quête de la valeur de $BB(5)$, qui s'est révélée si ardue qu'elle a parfois été considérée sans espoir !

UN DÉFI DE TAILLE

Le nombre de machines de Turing à cinq règles est faramineux : plus de 16 mille milliards ! Les étudier toutes à raison d'une par seconde demanderait environ un demi-million d'années. Heureusement des méthodes de réduction permettent de les regrouper et de repérer rapidement certaines de celles qui tournent en rond ; cela ramène le nombre de machines à étudier à un nombre plus abordable. Malheureusement, ces réductions ne suffisent pas à conclure aisément sur la valeur de $BB(5)$, car les meilleures méthodes ne réduisent le champ d'étude qu'à une centaine de millions de machines, ce qui reste considérable.

Le cas des machines à cinq règles qui s'arrêtent assez rapidement est facile à traiter. Pour une telle machine *MT*, en la faisant fonctionner jusqu'à l'arrêt, on connaît le nombre d'étapes qu'elle exécute ; si on a déjà rencontré une machine à cinq règles qui calcule plus longtemps avant de s'arrêter, on peut oublier *MT* et passer aux machines suivantes. Si au contraire, *MT* calcule plus longtemps que les machines à cinq états essayées auparavant et que *MT* s'interrompt, elle devient la détentrice temporaire du record. En notant R le nombre d'étapes de calcul de *MT* avant qu'elle ne s'arrête, on peut alors affirmer que $BB(5) \geq R$.

En pratique, pour mener une telle recherche, on est obligé de fixer une borne B correspondant au nombre d'étapes à partir duquel on cessera l'essai des machines testées. En effet, il n'est pas possible de laisser fonctionner indéfiniment les machines essayées, et l'on sait que certaines ne s'arrêtent jamais. Quelle que soit la borne B qu'on se donnera, il y aura

d'ailleurs un assez grand nombre de machines à cinq règles dont le nombre d'étapes de calcul atteindra la borne B avant que la machine ne s'arrête. Pour une telle machine MT, deux situations sont alors possibles :

(a) MT calcule indéfiniment. Dans ce cas, elle n'est pas susceptible de battre le record temporaire, et il n'est donc pas grave d'avoir interrompu son exécution dans le cadre de notre recherche.

(b) MT s'arrête au bout d'un nombre d'étapes de calcul fini mais supérieur à B. Ce cas est problématique, car en interrompant l'exécution de MT on passe potentiellement à côté de la valeur exacte de $BB(5)$.

La difficulté provient donc des machines qui ne s'arrêtent pas avant que soit atteinte la borne temporaire B. Les plus coriaces d'entre elles, même en petit nombre, empêchent de conclure sur la valeur exacte de $BB(5)$.

Il faut donc déterminer intelligemment la borne B, et démontrer mathématiquement que toutes les machines qui continuent à calculer au-delà de B étapes sont des machines dont l'exécution ne s'arrête jamais. Pour certaines d'entre elles, ce sera assez facile, mais pas pour d'autres : c'est finalement là que la difficulté du problème se concentre.

UNE QUÊTE COLLECTIVE

Il aura fallu attendre 2024 et un formidable travail collectif pour que la valeur soit indubitablement établie : $BB(5) = 47\,176\,870$.

Pourtant, dès 1964, Milton Green prouve que $BB(5) \geq 79$ en identifiant une machine de Turing à cinq règles qui calcule pendant 79 étapes puis s'arrête. Malheureusement, fixer $B = 79$ se révèle très insuffisant. En effet, à l'occasion d'un concours organisé en 1983 à Dortmund, en Allemagne, l'informaticien Uwe Schult présente une machine qui effectue 134 467 étapes de calcul puis s'arrête. Il remporte ainsi le concours… mais cette valeur est encore trop petite ! Quelques années plus tard, en 1989, Heiner Marxen et Jürgen Buntrock prouvent que $BB(5) \geq 47\,176\,870$ en exhibant une machine à cinq règles trouvée un an plus tôt, qui effectue un tel nombre d'étapes de calcul puis s'arrête.

Dès lors, on ne parvient plus à battre le record. Une fois bien installée, chez les spécialistes, la conviction que cette valeur est la bonne, le travail cesse d'être calculatoire et devient mathématique : il faut établir rigoureusement que toutes les machines à cinq règles qui ne s'arrêtent pas avant l'étape 47 176 871 ne s'arrêtent, en fait, jamais.

Au début des années 2000, l'informaticien bulgare Georgi Georgiev – connu par son surnom « Skelet » car il était très maigre – propose un nouveau résultat fondé sur une méthode nouvelle et un programme de 6 000 lignes en langage de programmation Pascal. Au bout d'une semaine d'exécution, ce programme conclut que toutes les machines de Turing à cinq règles qui s'arrêtent terminent leur exécution au plus tard à l'étape 47 176 870… sauf peut-être 43 d'entre elles, qui seront dès lors appelées les « 43 Skelet ».

La difficulté se retrouve donc concentrée sur 43 machines, dont il faut comprendre le fonctionnement pour démontrer qu'aucune ne s'arrête. Au fur et à mesure des essais menés par divers autres chercheurs, avec des bornes de plus en plus grandes, la conviction se renforce que ce nombre de 47 176 870 est bien la

2 LA TERRIBLE FONCTION *BB*

La fonction *BB* indique, pour chaque entier $k \geq 1$, le nombre maximum d'étapes de calcul qu'une machine de Turing à k états peut exécuter avant de s'arrêter. C'est un exemple de fonction non calculable : aucun programme ne peut en donner toutes les valeurs. En effet, s'il existait un programme *bb* qui calculait chaque $BB(k)$, alors on en déduirait un programme *ar* qui indiquerait, pour chaque machine de Turing, si elle s'arrête ou pas. Le programme *ar* fonctionnerait de la façon suivante : pour n'importe quelle machine de Turing M à k états, il calculerait $BB(k)$ en utilisant *bb* comme sous-programme. Il ferait ensuite fonctionner M pendant $BB(k)$ étapes, et renverrait « M s'arrête » si et seulement si M s'arrêtait pendant ce calcul – il renverrait « M ne s'arrête pas » sinon. Le programme aurait raison, puisque par définition aucune machine à k états ne s'arrête après l'étape $BB(k)$. Or, on sait grâce aux travaux d'Alan Turing qu'un tel programme indiquant l'arrêt ou le non-arrêt de toute machine de Turing ne peut pas exister. Par conséquent, ce programme *bb* ne peut exister.

La fonction *BB* permet aussi de concevoir des indécidables précis possédant un sens assez simple. En 2016, Adam Yedidia et Scott Aaronson ont ainsi démontré que l'énoncé mathématique indiquant la valeur de $BB(7910)$ était un indécidable de la théorie des ensembles de Zermelo-Fraenkel avec axiome du choix (ZFC). Le résultat fut ensuite amélioré : on sait aujourd'hui que les énoncés mathématiques indiquant respectivement les valeurs de $BB(1919)$ et $BB(748)$ sont aussi des indécidables de ZFC.

On sait aussi que cette fonction *BB* est très rapidement croissante. Pavel Kropitz a par exemple démontré en 2022 que

$$BB(6) \geq 10 \uparrow\uparrow 15 = 10^{10^{10^{\cdot^{\cdot^{\cdot^{10}}}}}}$$

(15 élévations à la puissance 10 successives, dans l'exposant). On établit aussi qu'elle est plus rapidement croissante que n'importe quelle fonction calculable, en ce sens que pour toute fonction calculable f, il existe une infinité d'entiers n tels que $BB(n) > f(n)$.

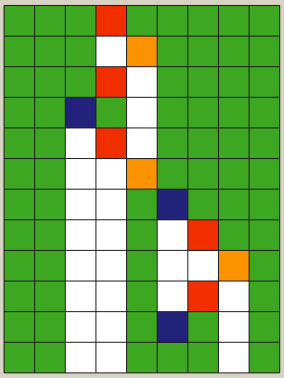
3 VISUALISATION DU FONCTIONNEMENT

Tristan Stérin a créé un programme permettant la visualisation du comportement des machines de Turing sous la forme de diagrammes. Chaque ligne de ces diagrammes représente une étape de l'évolution d'une machine de Turing et de son ruban. On présente dans cet encadré quelques exemples.

MACHINE 1

*A*0	*A*1	*B*0	*B*1	*C*0	*C*1
1*RB*	0*LC*	1*LA*	0*RC*	1*RA*	*Halt*

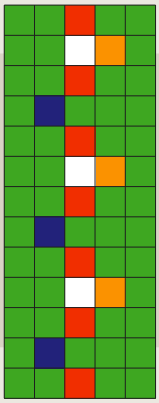
La machine 1 s'arrête après onze pas de calcul.
La première ligne du diagramme correspond à la position de la T-L-É au moment du départ. Sa position est représentée en rouge car l'état initial de la machine est *A*, que nous convenons de dessiner en rouge. Le vert est utilisé pour les cases où se trouve un 0, le blanc pour celles où se trouve un 1.
La première étape de calcul, déterminée par la consigne 1*RB*, consiste à écrire un 1 à la place du 0, à déplacer d'une case vers la droite la T-L-É, et à passer dans l'état *B* qu'on représente en orange.
La deuxième étape, déterminée par la consigne 1*LA*, consiste à écrire un 1 à la place du 0, à déplacer d'une case vers la gauche la T-L-É, et à passer dans l'état *A*, toujours représenté en rouge.
La troisième étape, déterminée par la consigne 0*LC*, consiste à écrire un 0 à la place du 1, à déplacer d'une case vers la gauche la T-L-É, et à passer dans l'état *C* qu'on représente en bleu, etc.
Arrivée à l'étape 10, la machine est dans l'état *C*, et sa T-L-É lit un 1 : elle passe donc dans l'état Halt, c'est-à-dire qu'elle s'arrête.

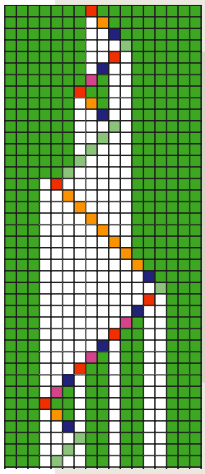


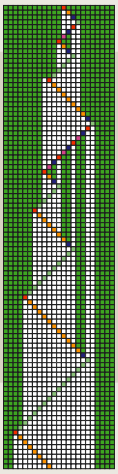
MACHINE 2

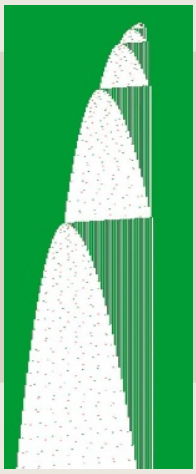
*A*0	*A*1	*B*0	*B*1	*C*0	*C*1
1*RB*	0*LC*	0*LA*	1*RA*	0*RA*	*Halt*

La machine 2 s'engage dans un calcul infini. Après avoir déplacé sa T-L-É à droite en passant dans l'état *B*, puis vers la gauche deux fois de suite en passant dans l'état *A* puis dans l'état *C*, la machine se retrouve dans l'état *A* dans une situation exactement identique à celle de départ. Elle refait donc indéfiniment les mêmes mouvements et écritures.









MACHINE 3 : LE « CASTOR AFFAIRÉ À CINQ ÉTATS »

*A*0	*A*1	*B*0	*B*1	*C*0	*C*1	*D*0	*D*1	*E*0	*E*1
1*RB*	1*LC*	1*RC*	1*RB*	1*RD*	0*LE*	1*LA*	1*LD*	*Halt*	0*LA*

Le calcul du castor affairé ne peut pas être représenté complètement. Ses premières étapes sont dessinées à différentes échelles ci-contre, selon les mêmes principes que pour les deux premières machines.

valeur de $BB(5)$. En 2020, Scott Aaronson, de l'université du Texas à Austin, émet formellement cette conjecture.

Las et à court d'idées, Georgi Georgiev finit néanmoins par abandonner le projet de traiter les 43 dernières machines « Skelet ». Mais en 2021 arrive la relève : le français Tristan Stérin, aujourd'hui responsable de la recherche pour l'entreprise prgm.dev, qu'il a cofondée, prend connaissance de la conjecture de Scott Aaronson de 2020. Conscient de la difficulté du problème et ne pensant pas pouvoir le traiter seul, il lance le *Busy Beaver Challenge* (« Challenge du castor affairé ») : une collaboration en ligne ouverte à toutes et tous, dont l'objectif est d'établir définitivement que $BB(5) = 47\,176\,870$.

Tristan Stérin sait que pour qu'une preuve soit unanimement considérée valide, elle doit être bien documentée et reproductible. Or le programme de Georgi Georgiev réduisant le problème à l'étude des « 43 Skelet » est extrêmement complexe et mal documenté – certains le considèrent même incompréhensible. La collaboration ne peut donc pas s'appuyer sur ce résultat pour construire les éléments définitifs d'une preuve recevable ; il faut retrouver ces conclusions avec des méthodes plus rigoureuses, avant de traiter les 43 machines qui bloquent la conclusion.

Dès 2021, Tristan Stérin écrit un programme utilisant une nouvelle méthode d'énumération qui, indépendamment de la méthode de Georgi Georgiev, l'amène à une liste d'environ 120 millions de machines de Turing dont l'étude est suffisante pour conclure rigoureusement sur la valeur de $BB(5)$. Un quart de ces machines s'arrête avant la machine de Heiner Marxen et Jürgen Buntrock, ce qui réduit l'étude des cas difficiles à celle de 88 millions de machines.

Pour traiter ces cas, Tristan Stérin crée un programme permettant de visualiser le comportement des machines (*voir l'encadré 3*). Cela facilite le travail, mais ne dispense pas de prouver que les 88 millions de machines résultant de la première phase de la démonstration calculent toutes indéfiniment.

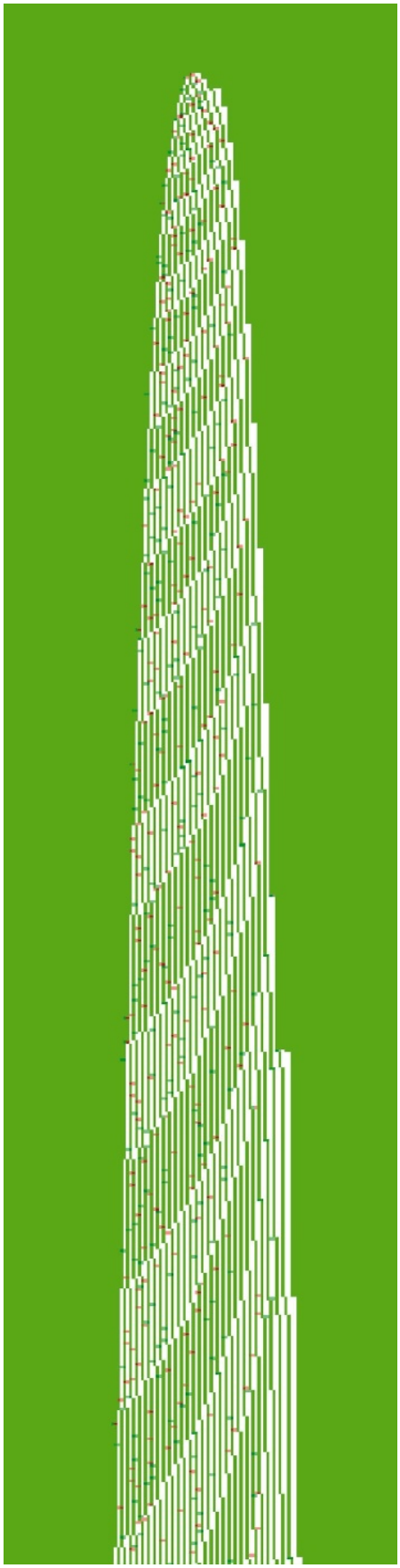
Divers participants au défi mettent au point et perfectionnent des méthodes automatiques de preuve, et de nombreux cas sont ainsi domptés. On remarque toutefois que, parmi les machines récalcitrantes, la machine Skelet n° 1 (une des 43 repérées par Georgi Georgiev) a un comportement très étrange : elle alterne des phases de calcul qui semblent aléatoires et des phases de calcul plus régulières, et cela presque indéfiniment. Alors que toutes les machines à cinq règles arrivent avant la millième étape dans une phase de calcul pas trop désordonnée, dont on peut espérer montrer qu'elle ne s'arrête jamais, la machine Skelet n° 1, elle, ne montre un tel comportement qu'après 10^{51} étapes !

En quelques mois, les chercheurs du *Busy Beaver Challenge* démontrent néanmoins que les machines récalcitrantes ne terminent, effectivement, jamais leurs calculs. Ils parviennent en particulier à traiter les cas des machines Skelet n° 1 et Skelet n° 17, les plus coriaces.

Encore fallait-il regrouper toutes ces démonstrations sous une forme compréhensible et acceptable par la communauté mathématique. C'est de la jeune informaticienne Maja Kądziołka que viendra l'idée d'utiliser l'assistant de preuve Coq dans cette quête de $BB(5)$. Cet outil, développé en France à partir de 1984, confère la garantie que des démonstrations complexes sont sans erreur (*voir l'encadré 4*). En avril 2024, un contributeur, connu uniquement sous le pseudonyme de Mxdys, parvient à formaliser, dans un langage compréhensible par Coq, l'ensemble de la preuve. À ce jour, ce Mxdys conserve encore l'anonymat ! Mais une chose est certaine : le 10 mai 2024, il annonce dans un post en ligne que « la démonstration en Coq concernant $BB(5)$ est terminée ». Depuis, tous les contrôles effectués ont confirmé le résultat. La démonstration est bien sûr disponible en ligne, sur la plateforme Github, pour celles et ceux qui souhaiteraient la consulter et la faire vérifier eux-mêmes par Coq.

UNE PREUVE CRÉDIBLE

L'utilisation d'outils informatiques pour une démonstration où aucune vérification « à la main » n'est possible est susceptible d'engendrer des doutes sur le résultat. Tristan Stérin rassure cependant : « La preuve Coq de Mxdys a été examinée en juillet par Yannick Forster, un chercheur à l'Inria expert de Coq. Depuis, Yannick et moi avons passé beaucoup de temps sur la preuve pour en améliorer certains aspects (rapidité, lisibilité, commentaires, etc.). Le travail de Yannick a d'ailleurs fait passer le temps de compilation de 13 heures à 45 minutes. Pour une preuve Coq, le plus important assurant la validité du résultat est d'examiner non pas la preuve elle-même (qui est garantie par Coq, à qui les mathématiciens font confiance), mais l'énoncé prouvé : seuls des humains peuvent assurer que l'énoncé formel qui est démontré correspond à ce que l'on veut effectivement. C'est pourquoi Yannick avait demandé à Mxdys, dès juillet, d'isoler l'énoncé dans un fichier à part. Ce fichier est très court et assez transparent – 75 lignes, contre 50 000 lignes pour la preuve au total. Pour être précis, l'énoncé fait 3 lignes, qui ne prennent sens qu'avec 72 lignes donnant les définitions utilisées dans l'énoncé. » Quant à savoir si certains cas pourraient avoir été oubliés, le chercheur l'assure : « Aucun cas ne peut avoir été oublié, cela est garanti par



4 UN COQ FRANÇAIS

L'assistant de preuve Coq est un logiciel qui permet de vérifier la validité de démonstrations mathématiques. Il a joué un rôle central dans la validation de la démonstration établissant que *BB*(5) = 47 176 870. Cet outil a été développé à partir de 1984, et continuellement perfectionné depuis. Il est basé sur le « calcul des constructions » de Thierry Coquand, aujourd'hui professeur d'informatique à l'université de Göteborg, en Suède, et de son directeur de thèse Gérard Huet, alors à l'Inria. Christine Paulin-Mohring, aujourd'hui professeuse à l'université de Paris-Saclay a considérablement contribué à la version actuelle de Coq, utilisée pour la preuve concernant le « cinquième castor affairé ».
Cet outil peut vérifier des preuves mathématiques, aider à rechercher des démonstrations ou encore synthétiser des programmes. C'est un logiciel libre fonctionnant sous la licence GNU LGPL. Il a notamment été exploité pour vérifier la démonstration du célèbre théorème des 4 couleurs, assurant que toute carte planaire peut être coloriée avec quatre couleurs sans que deux pays voisins ne portent la même couleur. Il a aussi permis de démontrer la correction de programmes et circuits utilisés par l'entreprise Airbus.

Coq ; si un cas était oublié, la preuve ne pourrait pas être validée, et si un cas avait été pris comme axiome, on le verrait. » Ne pourrait-il pas, néanmoins, y avoir un bug dans Coq qui permettrait la validation d'une preuve incorrecte ? Tristan Stérin tempère : « Coq peut avoir des bugs, mais les experts qui ont examiné la preuve indiquent qu'elle n'utilise que des fonctionnalités relativement basiques de Coq, qui sont testées régulièrement depuis des années. Ils considèrent qu'exploiter un éventuel bug dans Coq pour prouver *BB*(5) est infiniment peu probable. Par ailleurs, il n'y avait pas de motivation financière ou autre à prouver le résultat, et Mxdys a acquis depuis longtemps la confiance de la communauté par sa participation fréquente et de très bonne qualité à nos échanges en ligne. »

Pour le chercheur, cependant, la publication d'un article dans une revue ou un congrès scientifique reste nécessaire : « Je pense qu'il très important d'expliquer la preuve dans un papier "classique". Nous publierons d'abord un article préliminaire sur *arXiv*, puis un article de congrès, puis un article de revue. Très peu de personnes ont les capacités de comprendre et d'ingurgiter 50 000 lignes de Coq. Les techniques développées par le "*bbchallenge*", qui sont à la base de cette preuve et qui ont impliqué plus de quinze personnes, doivent être expliquées. Dans notre communauté, il y a consensus sur ce point, car pour nous la connaissance serait quasiment perdue si elle ne restait qu'encodée en Coq. Certains choix de programmation sont peu évidents. Il faut aussi comprendre que la preuve Coq est très "computationnelle" : elle contient des programmes qui déterminent les comportements des machines de Turing et les preuves de correction de ces programmes, c'est-à-dire des preuves que quand ces programmes disent "telle machine ne s'arrête pas", celle-ci ne s'arrête effectivement pas. Pour que tout cela soit clair, il faut publier un article. »

UNE NOUVELLE ÈRE MATHÉMATIQUE ?

Insistons sur le fait que le résultat obtenu pour *BB*(5) s'appuie sur une nouvelle façon de pratiquer les mathématiques, associant à la fois du calcul (pour trouver la machine record), de nombreux raisonnements mathématiques, un travail d'équipe coordonné en ligne, le regroupement en une seule preuve de résultats partiels, et la validation du tout par un assistant de preuve.

L'idée d'organiser la collaboration de nombreux de mathématiciens et mathématiciennes pour traiter une question difficile a déjà été mise en œuvre dans ce qu'on appelle des « projets Polymath ». Depuis 2009, plus d'une dizaine de projets de ce type ont été mis en place. L'un d'eux, par exemple, porte sur le problème des nombres premiers jumeaux. Un autre, concernant la théorie de Ramsey, a d'ores et déjà abouti et donné lieu à une publication dans la prestigieuse revue *Annals of Mathematics*.

Ce qui est nouveau, ici, c'est la certification des résultats par un outil informatique, qui exclut pratiquement toute possibilité d'erreur dans la démonstration, pourtant très longue et très complexe. Ce type d'outil a aussi été utilisé dans un récent projet promu par le célèbre mathématicien médaillé Fields Terrence Tao – le projet portait sur les théories équationnelles et s'appuyait sur un autre assistant de preuve, Lean. C'est l'émergence d'une nouvelle conception des mathématiques.

Bien sûr, le succès du calcul de *BB*(5) pousse à s'interroger sur la valeur de *BB*(6). Il y a cependant deux raisons de penser que ce défi sera extrêmement difficile à relever. D'abord, bien sûr, le nombre de machines de Turing à six états est encore bien supérieur au nombre de machines à cinq états : il y en a plus de 3 000 fois plus ! Même en réduisant beaucoup ce nombre, il est compliqué d'imaginer pouvoir s'attaquer au nouveau problème par la même méthode que celle utilisée pour *BB*(5). De plus, il a récemment été démontré qu'avec six règles, certaines machines de Turing codent des problèmes ressemblant beaucoup à la célèbre conjecture de Collatz (aussi appelée « conjecture de Syracuse »), qui défie les mathématiciens depuis plus de soixante-dix ans sans que personne n'ait encore réussi à la démontrer. Cela suggère que calculer *BB*(6) exigera de la part des spécialistes la démonstration de résultats bien plus complexes que ceux rencontrés pour calculer *BB*(5). ■

BIBLIOGRAPHIE

B. Brubaker, With fifth busy beaver, researchers approach computation's limits, *Quanta Magazine*, 2024.

P. Michel, The Busy Beaver Competition : A historical survey, *arXiv preprint*, 2022.

T. Stérin, The Busy Beaver Challenge, *site web collaboratif*, 2021.

S. Aaronson, The busy beaver frontier, *ACM SIGACT News*, 2020.

P. Michel, Problems in number theory from busy beaver competition, *Logical Methods in Computer Science*, 2015.

H. Marxen et J. Buntrock, Attacking the Busy Beaver 5, *Bulletin of the EATCS*, 1990.

A. Brady, *Solutions of Restricted Cases of the Halting Problem*, Oregon State University (thèse de doctorat), 1965.

T. Radó, On non-computable functions, *The Bell System Technique Journal*, 1962.