Une preuve pour le lycée de l'indécidabilité du problème de l'arrêt

Matthieu Journault, Pascal Lafourcade, Rémy Poulain, Malika More

Une preuve pour le lycée de l'indécidabilité du problème de l'arrêt

Matthieu Journault[1], Pascal Lafourcade[2], Rémy Poulain[1], and Malika More[2]

[1] Sorbonne Université, LIP6
[2] Université Clermont Auvergne, LIMOS, IREM

Résumé L'objectif de cette activité est de faire démontrer par des élèves à partir de la troisième qu'il existe des problèmes informatiques qui sont *indécidables*, c'est-à-dire pour lesquels il n'existe pas d'algorithme pour les résoudre. Pour montrer qu'il existe de tels problèmes, l'activité en exhibe un, *le problème de l'arrêt* : étant donnée une paire constituée de l'encodage d'une machine de Turing M et d'un paramètre d'entrée w, l'exécution de M sur w s'arrête-t-elle ? L'activité suit la preuve proposée par Turing, mettant ainsi en avant le principe du raisonnement par l'absurde et la notion de disjonction de cas. Tout au long de l'activité, les élèves se servent d'une version papier d'un modèle simplifié d'ordinateur pour exécuter à la main des programmes (ici écrits en Scratch). Ceci leur permet d'abord de découvrir des programmes simples, mais aussi de se familiariser avec un modèle de calcul proche de celui d'une machine à registres. Certains de ces programmes simples sont ensuite composés ensemble pour obtenir le résultat d'indécidabilité.

1 Introduction

La conception de programmes est une activité omniprésente à la fois dans toutes les sous-disciplines de l'informatique, mais aussi à tout les niveaux de compétences, c'est d'ailleurs souvent par la conception de programmes que débute l'apprentissage de l'informatique. Certaines activités d'initiation peuvent prendre la forme d'activités utilisant les langages de programmation Scratch ou bien Snap ! [Ber11]. D'autres peuvent se faire sans ordinateur (ou algorithmique débranchée [BRC12]), on peut citer par exemple l'activité du crêpier, au cours de laquelle un tas de crêpes est classé à l'aide d'une spatule [Bul13], ou encore les activités développées à l'IREM/MPSA de Clermont-Ferrand dans le groupe Informatique Sans Ordinateur [3] auquel appartiennent deux des auteurs. Cet article propose une activité sans ordinateur, utilisant comme support le langage de programmation Scratch. Les programmes utilisés peuvent sans aucune difficulté être traduits dans un autre langage de programmation, en fonction des connaissances des élèves.

Il nous semble important de montrer rapidement aux élèves que certains problèmes informatiques n'ont pas de solution algorithmique. Ce résultat fondamental de l'informatique, démontré en 1936 par Alan Turing [Tur36], peut être

3. http ://www.irem.univ-bpclermont.fr/informatique-sans-ordinateur-140

présenté assez simplement à des élèves de fin de collège et de lycée. Notons à ce propos qu'il sera attendu des élèves de la spécialité NSI en classe de Terminale de « *Comprendre que tout programme est aussi une donnée. Comprendre que la calculabilité ne dépend pas du langage de programmation utilisé. Montrer, sans formalisme théorique, que le problème de l'arrêt est indécidable* » [Nat19]. Le but de cet article est de présenter une activité débranchée qui fait travailler les élèves sur chacun de ces points. Cette activité peut être proposée à partir de la troisième. Pour des élèves plus grands, les programmes Scratch peuvent être remplacés par des programmes dans un autre langage de programmation comme Python ou bien en pseudo code.

Cette activité aborde de plus les problématiques suivantes :

— Lecture de programmes ;
— Exécution pas à pas d'un programme ;
— Composition de programmes ;
— Raisonnement par l'absurde ;
— Raisonnement par disjonction de cas ;
— Démonstration en informatique ;
— Modèles de calcul.

Cette activité ne nécessite pas d'ordinateur, mais un peu de matériel imprimable pour représenter la machine ainsi que des programmes Scratch à imprimer [4]. Afin de permettre les interactions entre les élèves, ils peuvent être répartis en groupes de trois. La description faite dans cet article suit la progression pédagogique créée afin de prouver l'indécidabilité du problème de l'arrêt. Lors de la création de cette activité nous nous sommes inspirés d'une vidéo [5] de la chaîne Youtube *Udiprod*. Il est intéressant de montrer cette vidéo aux élèves une fois l'activité finie, afin de synthétiser la preuve, ceci en sept minutes et cinquante et une secondes.

Plan : Dans la section 2, une présentation du matériel nécessaire pour l'activité est faite et des activités préparatoires sont proposées pour introduire progressivement les programmes et notions nécessaires pour la preuve d'indécidabilité. Dans la section 3, une activité simple pour sortir d'un labyrinthe est décrite afin d'introduire les notions de preuve par disjonction de cas mais aussi de preuve par l'absurde. Dans la section 4, l'activité en elle-même est décrite. Enfin, avant de conclure dans la dernière section, nous décrivons dans la section 5 différents retours d'expérience pour cette activité dans trois situations différentes, au collège, au lycée et en formation continue d'enseignants.

2 Activités préparatoires

Tout d'abord nous présentons le matériel nécessaire pour l'activité et ensuite nous décrivons chacun des programmes préparatoires à la preuve du théorème d'indécidabilité.

4. http ://www.irem.univ-bpclermont.fr/Indecidabilite-du-probleme-de-l
5. https ://www.youtube.com/watch ?v=92WHN-pAFCs

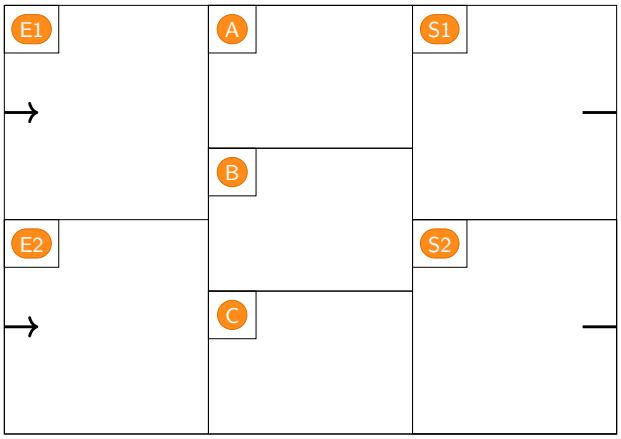

FIGURE 1: Représentation d'un ordinateur.

2.1 Matériel

Afin de représenter l'exécution de programmes sur un ordinateur, nous adoptons un modèle très simplifié de celui-ci. L'ordinateur est modélisé par une "ardoise" sur laquelle est dessiné le schéma de la Figure 1 ou bien par une feuille de papier sur laquelle est imprimé ce même schéma. L'activité dure une heure en comptant le temps pour faire des remarques d'introduction et de conclusion.

Cet ordinateur simplifié comporte deux entrées dénotées E1 et E2, deux sorties dénotées S1 et S2, et trois cases mémoires dénotées A, B et C. L'objectif de la prochaine section est de se familiariser avec le fonctionnement de cet ordinateur simplifié à l'aide de programmes Scratch simples qui sont réutilisés dans la preuve de la section 4. De plus, c'est une bonne façon de familiariser les élèves avec un modèle de calcul abstrait. Le modèle d'ordinateur utilisé dans cette activité est proche de ce qu'un informaticien appellerait une machine à trois registres.

2.2 Exécution de programmes simples

Le terme *machine* désigne un groupe de trois participants, équipé d'une ardoise sur laquelle est dessiné le schéma de la Figure 1. Cette ardoise sert à représenter l'évolution de l'état de la machine lors de l'exécution d'un programme.

Nous expliquons, via des exemples simples de programmes Scratch donnés dans les Figures 2 à 7, comment des programmes s'exécutent sur notre modèle d'ordinateur. Pour chaque exemple, le programme est donné à chaque groupe d'élèves. Ils doivent exécuter le programme sur des entrées choisies par le groupe. L'ordre dans lequel ces exemples sont présentés aux élèves correspond à une progression pédagogique afin de leur faciliter la compréhension du fonctionnement du modèle d'ordinateur utilisé dans l'activité.

INCRÉMENT *(Figure 2) :* Chaque élève du groupe choisit à son tour un entier qu'il écrit en entrée E1, et observe qu'au terme de l'exécution de ce programme, il obtient l'entier suivant écrit en sortie S1. L'objectif est simplement de se familiariser avec l'utilisation du matériel sur un programme simple qui ajoute 1 à l'entrée choisie.

INCRÉMENT *puis* INCRÉMENT *:* Ce deuxième temps de l'activité nécessite deux ardoises par groupe d'élèves. L'objectif est d'attirer l'attention sur le fait que les ardoises distribuées aux élèves (voir Figure 1) comportent des demi-flèches, sortantes sur les cases S1 et S2 et entrantes sur les cases E1 et E2. En accolant deux machines M_1 et M_2, on en obtient ainsi une nouvelle machine M, dont le comportement est le suivant :

- les entrées de la machine M sont les entrées de M_1 ;
- les sorties de la machine M sont les sorties de M_2 ;
- exécuter M revient à exécuter M_1, recopier les résultats obtenus dans les sorties de M_1 dans les entrées de M_2 (S1 dans E1, S2 dans E2), puis à exécuter M_2.

Il s'agit ici d'appréhender la notion de composition de programmes : chaque groupe d'élèves utilise deux machines, chacune exécutant le programme INCRÉMENT. Il est important de tester la machine composée ainsi obtenue sur plusieurs entiers et d'insister sur la transmission des sorties de l'une dans les entrées de l'autre.

PHOTOCOPIE *(Figure 3) :* Ce programme est à exécuter d'abord en choisissant une entrée numérique, et ensuite en choisissant comme entrée un programme. Pour cela, les fichiers fournis sur le site [6] contiennent une version du programme INCRÉMENT imprimé en petit pour qu'il rentre dans les cases E1, S1 et S2. Il en faut trois exemplaires pour cet exercice. Ensuite, le programme PHOTOCOPIE est exécuté en prenant comme entrée le programme PHOTOCOPIE lui-même, il faut donc aussi trois exemplaires d'une "petite version" de ce programme. Le but de ces exercices est de présenter un point crucial de la la preuve d'indécidabilité : le fait que les ordinateurs peuvent accepter en entrée non seulement des nombres mais aussi des programmes. Ce comportement surprend au premier abord les élèves mais, après discussion avec l'enseignant, ne leur pose pas de problème de compréhension.

MOINS *(Figure 4) :* Dans chaque groupe, chaque élève à son tour choisit deux entrées avec $E1 > E2$, et exécute ce programme. L'objectif est de se familiariser avec un programme qui utilise des variables et une boucle (et dont l'exécution se termine, voir ci-dessous).

MOINS* *(Figure 5) :* Dans chaque groupe, chaque élève à son tour choisit deux entrées avec $E1 > E2$, et exécute ce programme. Il s'agit souvent de la

6. http ://www.irem.univ-bpclermont.fr/Indecidabilite-du-probleme-de-l

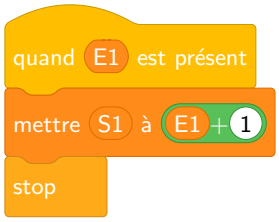

FIGURE 2: Programme INCRÉMENT

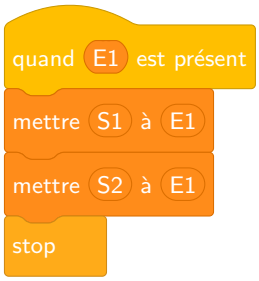

FIGURE 3: Programme PHOTOCOPIE

première rencontre avec un programme qui boucle indéfiniment, c'est-à-dire dont l'exécution ne se termine pas. Il est important d'expliquer que ce programme a été obtenu à partir du programme MOINS en "oubliant" simplement une instruction, et qu'il est par conséquent souvent difficile de déterminer si l'exécution d'un programme se terminera ou pas en lisant son code.

SUPER *(Figure 6) :* Dans chaque groupe, chaque élève à son tour choisit une entrée, et exécute ce programme. Après quelques essais, l'enseignant demande aux élèves d'expliquer sur quelles entrées l'exécution de ce programme ne se termine pas et sur quelles entrées elle se termine. L'exécution de ce programme se termine sur certaines entrées (les nombres négatifs ou nuls), mais ne se termine pas pour les nombres positifs. Trouver ces deux conditions demande un peu de réflexion pour les élèves, et cela montre qu'il n'est pas facile de déterminer si l'exécution d'un programme sur une entrée donnée se terminera ou pas, juste en lisant son code.

NÉGATION *(Figure 7) :* Il s'agit maintenant de demander aux élèves de chaque groupe de tester ce programme sur les entrées SE TERMINE puis NE SE TERMINE PAS (noter ici que SE TERMINE et ne NE SE TERMINE PAS sont des chaînes de caractères, s'ajoutant ainsi aux types de données que nos machines sont autorisées à manipuler). Ils peuvent constater que son comportement est inverse de ce qu'il reçoit en entrée : l'exécution du programme NÉGATION sur l'entrée SE TERMINE ne se termine jamais, alors que sur l'entrée NE SE TERMINE PAS (ou n'importe quelle autre valeur), elle se termine.
Les programmes NÉGATION et PHOTOCOPIE sont au cœur de la démonstration du théorème d'indécidabilité, il est donc important que les élèves comprennent bien leur fonctionnement.

3 Preuve par disjonction de cas et preuve par l'absurde

Afin d'introduire la notion de preuve par disjonction de cas, qui est très utilisée en informatique, et également la notion de preuve par l'absurde, nous

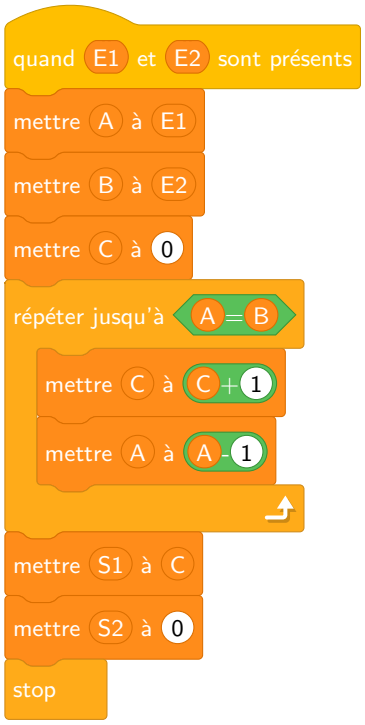

FIGURE 4: Le programme MOINS, sous la condition que $E1 > E2$.

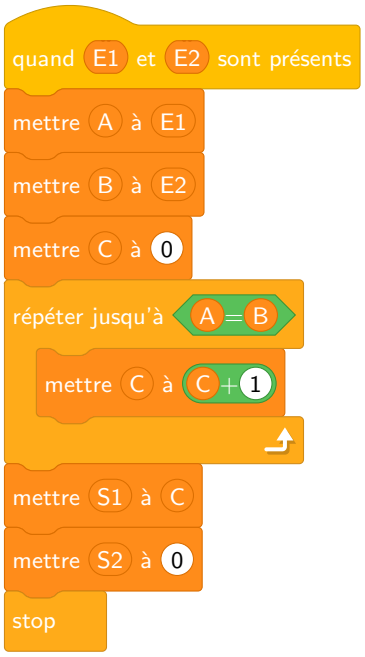

FIGURE 5: Le programme MOINS*, sous la condition que $E1 > E2$.

considérons dans cette section une situation simple et accessible sans mathématiques. Soit le labyrinthe de la Figure 8, qui contient un précipice mortel et infranchissable noté P et un monstre invincible noté M. Alice part du point A, et il s'agit de montrer qu'elle ne peut pas sortir vivante du labyrinthe. Les élèves se convainquent bien entendu facilement que c'est vrai, et l'activité a pour objectif d'expliciter une démonstration de ce résultat.

La preuve est une preuve par l'absurde. Supposons qu'Alice puisse sortir vivante du labyrinthe. Elle a seulement deux choix possibles à partir de sa position initiale A : partir vers la droite ou partir vers le bas.

Examinons les deux cas et montrons qu'ils sont tous deux impossibles.

1. **Vers la droite :** Si Alice part vers la droite, alors elle va finir par tomber dans le précipice P. Donc elle ne sortira pas vivante du labyrinthe par ce chemin. Cette solution est impossible.
2. **Vers le bas :** Si Alice part vers le bas, alors elle va se faire dévorer par le monstre M. Donc elle ne sortira pas vivante du labyrinthe par ce chemin. Cette solution est impossible également.

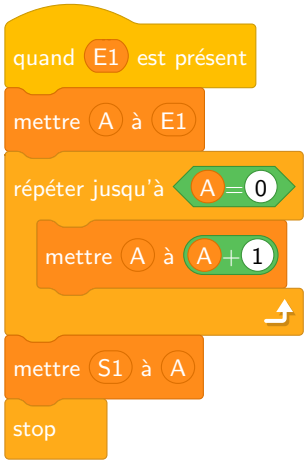

FIGURE 6: Programme SUPER

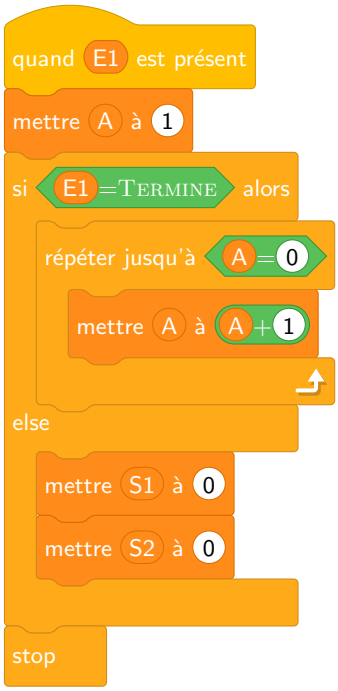

FIGURE 7: Le programme NÉGATION.

Finalement, nous pouvons conclure que l'hypothèse selon laquelle Alice peut sortir vivante du labyrinthe est fausse. Nous en déduisons qu'Alice ne peut pas sortir vivante du labyrinthe. Cette partie de l'activité peut prendre la forme d'une discussion interactive en classe entière avec les élèves, avec l'objectif d'expliquer pas à pas la preuve et de montrer comment utiliser les deux techniques de preuve sur un "théorème" simple.

4 Preuve d'indécidabilité

Il s'agit d'une preuve par l'absurde. Nous supposons l'existence d'un programme, appelé ARRÊT, ayant deux données d'entrée : la première représente un programme P et la seconde représente une donnée d'entrée E du programme P. Le programme ARRÊT détermine *sans jamais se tromper* si l'exécution du programme P sur l'entrée E se termine ou pas. Dans un premier temps, l'enseignant va manipuler ce programme ARRÊT avec les élèves, avant de montrer qu'un tel programme ne peut pas exister. La preuve est constructive : nous allons construire un programme qui contredit l'existence du programme ARRÊT à partir des trois programmes PHOTOCOPIE, ARRÊT et NÉGATION.

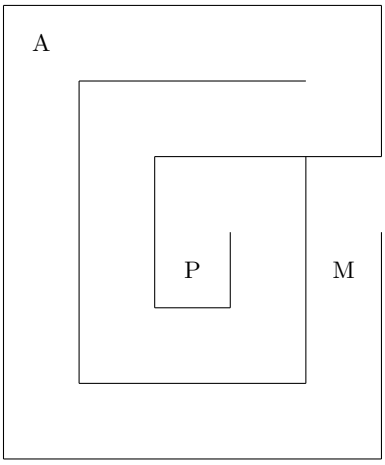

FIGURE 8: Labyrinthe.

4.1 Le programme Arrêt

Soit le programme, appelé ARRÊT, qui prend en entrées E1 et E2, et répond SE TERMINE si l'exécution du programme E1 sur l'entrée E2 se termine et NE SE TERMINE PAS sinon. Nous fournissons en Figure 9 ce programme ARRÊT écrit en "Scratch". Il est bien évidemment fallacieux, puisque l'activité démontre qu'il n'existe pas. Les élèves doivent être conscients que la ligne importante car difficile (en réalité impossible) à réaliser est le test :

l'exécution du programme E1 se termine sur l'entrée E2.

Dans chaque groupe, les élèves exécutent le programme ARRÊT sur les entrées suivantes afin de se familiariser avec son fonctionnement :

1. E1 est le programme INCRÉMENT et E2 est un nombre quelconque.
2. E1 est le programme SUPER et E2 est 5.
3. E1 est le programme SUPER et E2 est 0.

Pendant ces exécutions, il est important de faire remarquer aux élèves que, contrairement aux exécutions précédentes où il suffisait de suivre pas à pas les instructions des programmes, il faut ici réfléchir pour exécuter le test vert.

4.2 Preuve de l'indécidabilité

Une fois toute ces activités réalisées, nous sommes prêts pour démontrer par l'absurde que le programme ARRÊT ne peut pas exister. Pour cela, nous supposons que ce programme existe c'est-à-dire qu'il existe un programme qui se comporte comme nous l'avons vu à la section précédente. La preuve repose sur

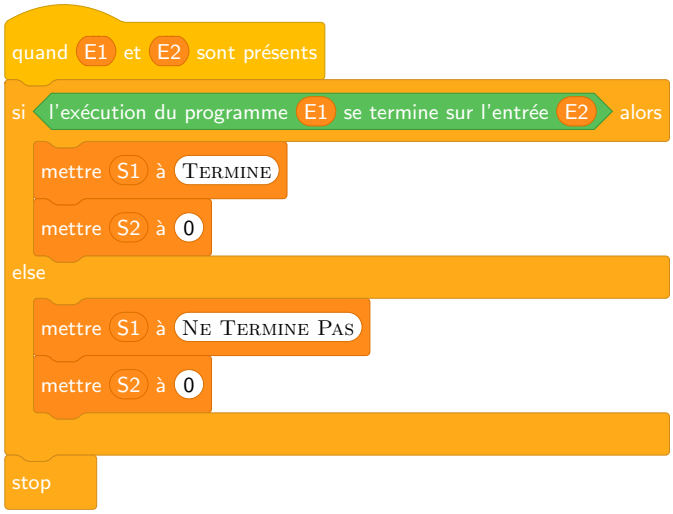

FIGURE 9: Le programme ARRÊT.

la construction d'un programme, noté X, qui utilise le programme ARRÊT et qui permet de contredire l'existence de ce même programme. Le programme X est le composé de trois programmes. Le premier programme est le programme PHOTOCOPIE, le deuxième est le programme ARRÊT, et le troisième est le programme NÉGATION. L'idée est d'exécuter le programme X avec comme entrée le programme X lui-même. Cela revient, pour commencer, à exécuter le programme PHOTOCOPIE en mettant X dans l'entrée E1. Ensuite, par construction du programme X, il faut déterminer si l'exécution du programme ARRÊT donne pour résultat SE TERMINE ou NE SE TERMINE PAS quand le programme X lui est passé à la fois en entrée E1 et en entrée E2. Il faut donc distinguer deux cas pour le résultat de l'exécution du programme ARRÊT sur les entrées X et X.

1. L'exécution du programme ARRÊT sur les entrées X et X donne pour résultat SE TERMINE. Dans ce cas, le programme NÉGATION reçoit comme entrée SE TERMINE et son exécution ne se termine jamais. Finalement, dans ce cas, l'exécution du programme X appliqué à l'entrée X ne se termine pas, ce qui contredit le résultat donné par le programme ARRÊT. Ce dernier programme étant réputé ne jamais se tromper, ce cas est impossible.

2. L'exécution du programme ARRÊT sur les entrées X et X donne pour résultat NE SE TERMINE PAS. Dans ce cas, le programme NÉGATION reçoit comme entrée NE SE TERMINE PAS et son exécution se termine. Finalement, dans ce cas, l'exécution du programme X appliqué à l'entrée X se termine, ce qui contredit le résultat donné par le programme ARRÊT.

Ce dernier programme étant réputé ne jamais se tromper, ce cas aussi est impossible.

Ainsi, nous aboutissons dans les deux cas à une contradiction, ce qui montre l'impossibilité de l'existence du programme ARRÊT. Il n'existe donc pas de programme qui détermine d'une façon générale si un programme donné se termine sur une entrée donnée.

5 Retours d'expérience en classe

L'activité a été testée par les auteurs dans trois situations différentes :

— En classe de troisième, avec des groupes de 14 élèves.
— En classe de seconde, en classe entière de 35 élèves.
— En formation continue d'enseignants dans le cadre du Diplôme Inter-Universitaire Enseigner l'Informatique au Lycée, avec des professeurs du secondaire de plusieurs disciplines scientifiques (mathématiques, informatique et technologie).

Les élèves travaillaient par groupe de 3 ou 4 pendant une séance d'une heure. Ils commencent par manipuler les premiers programmes, ensuite pour la preuve le programme ARRÊT leur est présenté et enfin le programme X leur est donné avec la consigne de regarder ce qui se passe quand X s'exécute avec comme entrée le code de lui-même.

5.1 Classe de Troisième

Ce groupe a testé l'activité juste après sa création. Les élèves ont été très perturbés par la démonstration par l'absurde, difficile à appréhender pour des élèves si jeunes. Ainsi nous avons ajouté l'exemple du labyrinthe pour clarifier ce point. Nous pouvons citer quelques retours d'élèves : *"C'était intéressant car j'y ai compris l'ordinateur et le fonctionnement de l'intérieur."*. Pour cet élève, plus que de comprendre la démonstration elle-même, le grand apport de cet atelier a été de comprendre l'exécution des programmes. Mais certains autres ont bien suivi la preuve et compris la portée du résultat : *"Cela nous a permis de comprendre que l'ordinateur ne peut pas tout faire et qu'il est bien moins intelligent que ce que l'on peut supposer."*. Il est satisfaisant de constater que dans la majorité des retours, les élèves avaient réussi à dégager une bonne intuition de ce théorème fondamental de l'informatique.

5.2 Classe de Seconde

Dans le cadre de l'option Informatique et Création Numérique (ICN) en Seconde, l'un des auteurs est intervenu dans une classe de 35 élèves pour réaliser cette activité en présence du professeur. Les élèves avaient déjà écrit des programmes en Scratch, ce qui a permis d'aller plus rapidement sur les programmes simples. Par contre, ils n'avaient jamais vu de programmes dont, involontairement, l'exécution ne se termine pas. L'activité avec le labyrinthe a permis

d'introduire le raisonnement par l'absurde et par disjonction de cas. La réussite de cette étape est cruciale pour que les élèves soient ensuite bien outillés pour faire la preuve du résultat principal d'indécidabilité. En reprenant pas à pas ce que nous avions fait pour le labyrinthe, les élèves sont arrivés par eux même à concevoir la preuve attendue, bien entendu une fois que nous leur avons présenté la machine X.

Le résultat démontré a vraiment surpris les élèves et les a fait beaucoup réfléchir sur ce qu'il est possible de faire ou non avec un ordinateur.

5.3 Formation continue de professeurs du secondaire

L'activité a été présentée dans le cadre du Diplôme Inter-Universitaire Enseigner l'Informatique au Lycée, destiné à former les professeurs de la spécialité Numérique et Sciences Informatiques (NSI) en première et en terminale. La plupart des enseignants présents ont découvert ce résultat fondamental lors de cette séance. Ils ont été surpris par la puissance de ce résultat, et ont aussi pris conscience qu'il faut faire attention lorsqu'ils construisent des énoncés pour leurs devoirs car certains pourraient ne pas avoir de solution. À cette occasion, nous avons parlé d'autres problèmes indécidables classiques, comme le théorème de Rice [Ric53,Wol06], le pavage du quart de plan par les tuiles de Wang [Rob71], la résolution des équations diophtantiennes [Mat93] [7], le problème de la mortalité des matrices [CHHN14] [8], ou encore le problème des correspondances de Post (PCP) [Sip06]. Les professeurs ont été convaincus de l'utilité de présenter la notion d'indécidabilité à leurs élèves. Ils ont aussi apprécié le fait que la démonstration proposée utilise des programmes Scratch simples qui peuvent facilement être traduit dans un autre langage, la rendant ainsi accessible pour des lycéens.

6 Conclusion

Dans cet article, nous avons présenté une activité pouvant être proposée à des élèves de fin de collège, de lycée ou au niveau post-bac. Cet activité établit l'existence de problèmes informatiques pour lesquels aucun programme ne peut donner le résultat en toute généralité. Il est à noter que la démonstration de ce résultat est maintenant au programme de la spécialité NSI du lycée en classe de Terminale. L'activité peut-être faite sans recours à l'outil informatique et s'inscrit donc une démarche d'informatique sans ordinateur. Malgré l'absence d'ordinateurs, l'activité fait usage de programmes écrits en Scratch qui sont facilement transposables dans un autre langage. Ceux-ci sont exécutés à la main par les élèves mais aussi manipulés comme des données (à l'image d'entiers ou de chaînes de caractères). Bien que l'activité soit faite sans ordinateur, il nous

7. Aussi connu sous le nom de 10ème problème de Hilbert.

8. Il s'agit de savoir si, à partir d'un ensemble de matrices à coefficients entiers, il est possible d'en trouver un sous-ensemble qui, une fois multipliées entre elles, donne la matrice nulle.

a semblé que faire jouer à l'élève le rôle de la machine qui exécute un programme lui permet de comprendre un peu plus le fonctionnement d'un ordinateur. En effet, il est attendu des élèves qu'ils exécutent divers programmes sans prendre d'initiative, à la manière d'un processeur. Notre activité est aussi l'occasion d'aborder une autre notion fondamentale en informatique : la composition de plusieurs programmes, mais aussi de manière duale : la décomposition d'un problème en sous-problèmes. Les notions de preuve par l'absurde et par disjonction de cas sont aussi abordées ce qui est à notre avis le premier obstacle dans la compréhension de la preuve. L'exemple du labyrinthe peut servir pour créer des activités préliminaires pour que les élèves acquièrent ces mécanismes souvent utilisés en informatique. Le second obstacle vient de la compréhension de la contradiction dans chaque cas, cela nécessite de prendre du recul sur ce que la machine fait et ce qu'elle a dit qu'elle allait faire. Enfin, comme souligné en Section 5 le résultat obtenu pendant l'activité donne souvent lieu à des discussions intéressantes sur ce qui peut être attendu ou pas d'un ordinateur, mais aussi sur la distinction entre la puissance d'un ordinateur (sa rapidité) et son pouvoir d'expression (ce qu'il peut faire).

Références

[Ber11] Berkley. Snap !, 2011. https ://snap.berkeley.edu.

[BRC12] Timothy Bell, Frances Rosamond, and Nancy Casey. Computer science unplugged and related projects in math and computer science popularization. pages 398–456, 01 2012.

[Bul13] Laurent Bulteau. *Ordre et désordre dans l'algorithmique du génome.* PhD thesis, Université de Nantes, Faculté des sciences et des techniques, 2013.

[CHHN14] Julien Cassaigne, Vesa Halava, Tero Harju, and François Nicolas. Tighter undecidability bounds for matrix mortality, zero-in-the-corner problems, and more. *CoRR*, abs/1404.0644, 2014.

[Mat93] Yuri V. Matiyasevich. *Hilbert's Tenth Problem.* MIT Press, Cambridge, MA, USA, 1993.

[Nat19] Education Nationale. Nouveau programme nsi. Journal officiel, 2019.

[Ric53] H. G. Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society*, 74(2) :358–366, 1953.

[Rob71] Raphael M Robinson. Undecidability and nonperiodicity for tilings of the plane. *Inventiones mathematicae*, 12(3) :177–209, 1971.

[Sip06] Michael Sipser. *Introduction to the Theory of Computation.* Course Technology, second edition, 2006.

[Tur36] Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42) :230–265, 1936.

[Wol06] Pierre Wolper. *Introduction à la calculabilité.* Dunod, 2006.