Un château… pas très fort

Marie Duflot, enseignante-chercheuse
Université de Lorraine, CNRS, Inria, LORIA

Parler de thématiques proches de son domaine de recherche par le biais d'une activité d'informatique débranchée, c'est possible. Cet article illustre comment on peut aborder des concepts de vérification formelle de systèmes avec des élèves de l'enseignement secondaire. L'objectif n'est pas ici seulement d'expliquer des concepts pour les faire comprendre aux élèves mais d'amener ces derniers à réfléchir et à proposer eux-mêmes des solutions aux problèmes posés. Le champ d'application de cette activité – conçue initialement pour des lycéens – est plus étendu que prévu puisqu'elle a été utilisée dans un cours de master 2, pour présenter la logique temporelle et le *model checking*.

Le public

Si beaucoup d'activités d'informatique débranchée s'adaptent à une grande diversité de publics et peuvent souvent, du moins pour partie, être utilisées avec des publics jeunes, cette activité-ci est plutôt recommandée pour les élèves à partir du lycée (elle a été testée en partie avec des élèves de troisième… qui auraient pu ne pas aller jusqu'au bout, avec ou sans aide) ; pour travailler, on regroupe les participants par groupes de deux (on peut faire varier la taille du groupe), et il faut bien sûr un exemplaire du matériel par groupe.

Le matériel

Pour réaliser l'activité, il faut imprimer les deux dernières pages de cet article : le château d'une part, et les opérateurs logiques d'autre part. Ces opérateurs doivent ensuite être découpés pour les séparer

Comme on va avoir besoin d'écrire sur le château, il est fortement recommandé de le plastifier et d'utiliser des feutres pour tableau blanc ou ardoise, sinon après avoir gommé 5 ou 6 fois on risque de percer le papier.

DOI:10.48556/SIF.1024.25.51

Le contexte

La dernière page de l'article (annexe C) représente un château. Ce n'est pas le plan auquel on s'attend : il n'est pas à l'échelle, on ne reconnaît pas la forme des salles, ni des couloirs, mais il décrit tout ce qu'on a besoin de savoir pour se promener dans le château. Ici, chaque cercle représente une salle du château, et chaque flèche entre deux salles signifie qu'on peut aller de la première salle à la deuxième.

Dans un château classique, quand on peut aller d'une salle à une autre c'est qu'il y a une porte, et on peut donc faire le chemin inverse : revenir dans la première salle. Cela correspond aux deux flèches entre le cercle tout à gauche et celui en bas à droite de celui-ci. Mais Eusebius, l'architecte de notre château est un farceur et il a créé beaucoup de portes et de passages secrets que, pour la plupart, on ne peut ouvrir ou activer que d'un côté, comme si la porte n'avait une poignée que d'un seul côté. Il y a même dans une salle en haut à gauche un passage secret qui part d'une pièce et nous ramène (après un tour sur nous-même ?) dans la même pièce.

Quels sont les objectifs de cette activité ?

On va tout d'abord écrire des propriétés sur le château, par exemple « on peut atteindre une fenêtre » ou « on va obligatoirement passer devant un tableau » dans un langage particulier appelé *langage logique* ou tout simplement *logique*, où les mots ont une signification très précise. Dans un deuxième temps, le but est de déterminer si le château (ou une salle particulière du château) satisfait chacune de ces propriétés.

Sur les deux exemples donnés dans le paragraphe précédent, cela ne fonctionne pas trop mal, on arrive presque sans se tromper à répondre à ces questions. Mais si la propriété est plus compliquée, ou si le château est plus grand, comment fait-on ? Eh bien, justement, « on » ne fait plus, c'est l'ordinateur qui le fait. Et expliquer à l'ordinateur comment vérifier si une propriété est satisfaite ou non est précisément le dernier objectif de cette activité.

Déroulé

Étape 1 : Apprivoiser les opérateurs

On commence d'abord par découvrir le château, et expliquer tous les éléments. Il y a tout d'abord des cercles représentant les salles, dont une particulière indiquée comme le départ, donc l'entrée du château. On a ensuite des flèches, les passages entre les salles, qui peuvent exister dans les deux sens, dans un seul sens, ou même aller d'une salle à elle-même (ce que l'on appelle une boucle). À côté de chaque salle, on trouve de petits dessins qui représentent ce

qu'on trouve dans la salle. Sur notre exemple, ce sont des tableaux, des fenêtres, des torches et des trésors.

Pour exprimer des propriétés sur le château et ce que l'on peut y faire, on a besoin d'un langage clair et sans ambiguïté. La langue naturelle n'est pas faite pour cela, elle permet beaucoup de nuances, mais est parfois ambiguë. La première étape de l'activité est d'apprendre à utiliser les opérateurs logiques (étiquettes à découper, en annexe B), pour formuler des propriétés du château.

On distribue à chaque groupe de participants un lot d'opérateurs, et on les aide à les appréhender en utilisant l'aide mémoire joint. Ces opérateurs se classent en plusieurs catégories :

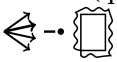
1. les opérateurs qui décrivent les propriété vraies de la salle ; il y a les torches, les trésors, les fenêtres et les tableaux ; l'opérateur tableau par exemple signifie qu'il y a un tableau dans la pièce où on se trouve ; on a aussi des opérateurs avec les mêmes objets barrés, qui signifient qu'il n'y a pas cet objet (torche, trésor, fenêtre, tableau) dans la pièce ;
2. les opérateurs logiques classiques : il est utile de les redéfinitir pour les participants, surtout l'implication ; donner la table de vérité de l'implication a, en particulier, été utile pour nombre de lycéens ;
3. les deux opérateurs comportant une ou plusieurs flèches permettent de parler des chemins qui partent d'une pièce ; celui avec une flèche $\longrightarrow$ dit qu'il existe un chemin à partir de cette pièce tel que… (et il faut écrire la suite de la formule avec d'autres opérateurs). Celui avec plusieurs flèches $\Lsh$ dit que pour tout chemin qui part de cette pièce on a … (et, là encore, la suite reste à écrire) ;
4. les quatre derniers opérateurs sont appelés opérateurs temporels car ils disent quand se passent les choses le long d'un chemin :
   – l'opérateur « un jour » dit qu'à un moment (c'est-à-dire « un jour ») le long de ce chemin, quelque chose (que l'on va écrire avec nos opérateurs) va être vrai,
   – l'opérateur « toujours » dit qu'à toutes les étapes de ce chemin, quelque chose (qu'on va écrire avec nos opérateurs) va être vrai,
   – l'opérateur « dans la prochaine pièce » dit que le long de ce chemin, dans la prochaine pièce (celle qu'on va atteindre juste après) quelque chose va être vrai,
   – enfin l'opérateur « jusqu'à » affirme qu'une chose est vraie jusqu'à ce qu'une autre devienne vraie ;
5. il y a aussi des parenthèses pour dire dans quel ordre appliquer les opérateurs ; par exemple : « un jour (fenêtre et torche) » est différent de « (un jour fenêtre) et torche » ; dans le second cas, la torche ce n'est pas un jour qu'on doit l'avoir mais tout de suite.

REMARQUE – Attention, si on ne met aucun opérateur temporel (ceux décrits dans le point 4. ci dessus) alors on parle de ce qu'il se passe dans la pièce actuelle. De plus on ne s'intéresse avec cette logique qu'à des chemins qui ne s'arrêtent jamais. Donc quand je dis « toujours » je n'ai pas le droit de m'arrêter dans la pièce qui me plaît et dire c'est bon j'ai toujours une fenêtre, je dois continuer à franchir des portes (mais si la porte me ramène dans la même pièce, alors là, j'ai le droit de la prendre encore et encore tout en restant dans la même pièce).

Voici quelques exemples permettant d'apprivoiser ces formules :

$\longrightarrow \bullet$  il existe un chemin qui me mène devant une torche (un jour) ;

$\longrightarrow \sim$  il existe un chemin le long duquel je suis toujours dans une pièce où il n'y a pas de torche, en d'autres mots il existe un chemin qui évite les torches (qui n'en croise aucune) ;

 dans la prochaine pièce j'aurai forcément un tableau (quelle que soit la porte que je prendrai) ;

$\longrightarrow \bullet \sim$  il existe un chemin le long duquel à partir d'un moment (c'est-à-dire un jour) j'ai toujours des fenêtres. En clair sur ce chemin peut-être qu'au début je n'ai pas de fenêtre, mais à partir d'un moment j'en verrai dans toutes les pièces par lesquelles je passerai.

Étape 2 – À la recherche des formules magi… euh… logiques.

Maintenant qu'on a un peu manipulé les opérateurs, on peut donner des phrases aux participants et leur demander de trouver les formules qui les expriment. Voici une liste exemple de propriétés pour s'entraîner à utiliser les opérateurs afin d'exprimer des propriétés :

1. je peux accéder à un trésor ;
2. je peux atteindre une pièce où je peux me remplir les poches (de trésor) tout en admirant un tableau ;
3. je peux aller dans une pièce où je pourrai correctement admirer un tableau (pour l'admirer il faut une source de lumière, il y en a deux possibles) ;
4. toute balade dans le château permet de voir au moins une fenêtre ;
5. je peux choisir de ne me balader que dans des pièces avec des torches ;
6. je vais forcément me balader, uniquement dans des pièces avec des torches ;
7. je peux arriver devant un tableau en exactement 3 déplacements ;
8. je peux arriver devant un tableau en 1 déplacement maximum ;
9. quoi qu'on fasse, on verra toujours une torche, un tableau ou une fenêtre dans sa pièce ;

10. je ne peux pas me retrouver devant un tableau sans torche ;
11. si je me promène dans le château, à partir d'un moment je ne verrai plus jamais de tableau ;
12. il existe un chemin (infini) qui ne passe jamais par le trésor ;
13. je peux me balader et ne plus voir de torches après un certain moment ;
14. quoi qu'on fasse, après deux déplacements on est forcément dans une pièce avec une torche ;
15. il y a moyen de se perdre dans le château dans un endroit à partir duquel je ne peux plus atteindre le trésor ;
16. quelqu'un qui a peur du noir peut atteindre le trésor (les gens qui ont peur du noir ne peuvent aller dans les pièces où il n'y a ni torche ni fenêtre) ;
17. si j'arrive dans une pièce avec un tableau alors deux pièces plus tard je verrai forcément une fenêtre ;
18. je peux atteindre le trésor en passant d'abord par des salles ayant toutes un tableau, puis des salles ayant toutes une torche, puis le trésor.

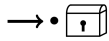
Quand on fait cet exercice il faut garder deux choses à l'esprit. Tout d'abord il peut exister plusieurs formules correctes pour une même propriété, on dit alors que ces formules sont équivalentes. Il faut reconnaître si une solution proposée est équivalente à celle qu'on a en tête. D'autre part lorsqu'une formule proposée est fausse, la façon de le prouver est de trouver un château qui satisfait notre propriété (écrite en français) mais pas la formule logique, ou inversement, un château qui satisfait la formule mais pas la phrase en français. Cela demande un peu d'imagination et aussi, préalablement, de tomber d'accord sur ce que veut dire la phrase en français. Et quand les participants disent que les phrases françaises sont ambiguës, c'est une bonne chose, cela fait un argument de poids pour justifier d'utiliser des langages logiques précis.

Étape 3 : Comment vérifier les formules ?

Un réflexe des participants est, dès qu'on leur donne une propriété, de regarder à la main si elle est vraie sur notre château. Les formules ci-dessus ne sont pas toutes vraies dans notre exemple de château, du moins pas à partir de la pièce de départ. On peut alors se poser la question de savoir si une autre pièce satisfait la formule et, si la réponse est encore non, de trouver un château non trivial qui satisfait la formule.

Un exemple de formule intéressante est la numéro 9. Quoi qu'on fasse, on verra toujours une torche, un tableau ou une fenêtre dans sa pièce. Quand on demande aux participants si elle est satisfaite dans le système, on obtient en général les deux réponses : oui et non. Les gens qui disent non le justifient en disant que la pièce tout en haut du plan n'a aucun des trois éléments demandés. Seulement cette pièce n'est pas accessible depuis le départ. Ainsi toutes les pièces qu'on atteint ont au moins un des trois éléments et la formule est vraie.

Pour faire comprendre l'utilité de la vérification automatique des formules, il faut se rappeler que les propriétés peuvent être un peu plus compliquées que celles que l'on a ici, mais surtout que les systèmes à vérifier sont **beaucoup** plus grands que notre petit château avec sa dizaine de pièces. En pratique on peut avoir des millions d'états (nos pièces), voire plus. Et, même pour une quarantaine de pièces, c'est trop long de le faire à la main et on a toutes les chances de se tromper. Pour trouver l'algorithme de vérification on peut utiliser la formule ci-dessous qui peut se lire : « il existe un chemin le long duquel un jour je rencontre un trésor » :



c'est l'une des formules les plus simples à vérifier. On commence par se demander s'il y a des pièces qui satisfont cette propriété et pour lesquelles on peut le voir facilement. Et effectivement il y en a : les pièces qui contiennent un trésor satisfont cette propriété car il existe bien un chemin (on peut prendre celui que l'on veut, peu importe) le long duquel un jour (et plus précisément maintenant dès le départ) il y a un trésor.

On va donc, sur notre feuille plastifiée, marquer d'un point les cercles correspondant à ces pièces.

Pour la suite on fait chercher aux participants d'autres pièces à partir desquelles on peut aller dans les pièces à trésor. La réponse émerge assez facilement : toutes les pièces qui ont une flèche qui amène dans une pièce à trésor ont bien un chemin qui un jour amène au trésor. On va donc pouvoir les marquer avec un point aussi. Et on continue : maintenant toutes les pièces qui en une étape nous amènent dans une pièce pointée permettent d'atteindre le trésor ; et en continuant on va toutes les trouver.

On peut formaliser un peu le procédé par l'algorithme suivant (que l'on peut faire chercher aux participants, en les aidant si nécessaire) :

```
Mettre un point dans toutes les pièces qui ont un trésor
tant que il y a des pièces avec un point dedans faire
    Choisir une pièce p avec un point dedans
    Remplacer le point dans p par une croix # pour exprimer le fait que
        l'on s'est occupé de cette pièce
    pour chaque pièce qui a une porte qui arrive dans p faire
        si il n'y a ni point ni croix dans cette pièce alors
            [S'il y avait déjà quelque chose on sait que la pièce ne
            sera pas oubliée]
            Y mettre un point
        fin si
    fin pour
fin tant que
```

On peut tout d'abord constater que cet algorithme se termine. Grâce au système de points et de croix, on met un point dans une pièce au maximum une fois, et une fois qu'on traite la pièce on remplace le point par une croix (pour garder l'information que la pièce a été traitée et ne pas la considérer une autre fois). Comme le nombre de pièces est fini, le nombre d'étapes aussi.

Ensuite, quand l'algorithme se termine, on n'a plus de point dans aucune pièce (sinon on ne sortirait pas de la boucle tant que), mais juste des pièces avec des croix et d'autres sans rien. De par notre construction, on ne met des points (qui se transforment ensuite en croix) que dans des pièces qui satisfont la formule « il existe un chemin le long duquel un jour j'atteins un trésor ». Et si une pièce p n'a pas de croix en fin d'algorithme, alors aucune des pièces que je peux atteindre en une étape à partir de p n'a de croix non plus. Et si elles n'ont pas de croix c'est qu'aucune des pièces qu'on atteint en une étape à partir d'elles n'en a. En continuant ce raisonnement, aucune pièce accessible à partir de p n'a de croix. Comme les pièces ayant un trésor ont forcément un point dès le départ (et donc une croix plus tard) c'est qu'elles ne sont pas accessibles à partir de p et que p ne satisfait pas notre formule.

Pour les autres opérateurs, il existe des algorithmes similaires, un peu plus compliqués suivant les cas, mais non abordés dans cette activité.

Retour à l'informatique

En cherchant un algorithme qui nous permette de trouver toutes les salles qui satisfont une propriété on se doute bien qu'on est en train de faire de l'informatique. Mais cet algorithme n'a pas été inventé pour cette activité, les formules non plus.

Tout d'abord le dessin du château est une façon formelle de représenter un système. Une fois le dessin réalisé, on connaît précisément tous les déplacements possibles dans le château et où s'y trouve chaque élément. Notre combinaison de cercles pour les salles, de flèches pour les passages entre deux salles et de petits dessins disant ce qu'on peut trouver dans la salle existent en informatique. Ce type de modèle est appelé *structure de Kripke*, les cercles sont des *états*, les flèches sont des *transitions* et les dessins des *propositions atomiques* (des propriétés très simples pour lesquelles on peut immédiatement dire si elles sont vraies dans un état ou pas).

Pour s'assurer que le comportement du système est bien tel qu'on s'y attend, on va exprimer des propriétés sur ce système à l'aide d'opérateurs logiques, exactement comme on l'a fait avec les petites étiquettes. Les opérateurs s'appellent E (there Exists a path / il existe un chemin) A (for All paths / pour tout chemin) F (Finally / un jour) X (neXt / après une transition) G (Globally / toujours) et U (Until / jusqu'à ce que) au lieu d'être des petits dessins comme on l'a vu pour le château, mais l'activité utilise bien tous les opérateurs d'un

langage logique appelé CTL (*computation tree logic*), et permet de découvrir les véritables algorithmes qui permettent de s'assurer qu'une formule (simple) de CTL est vraie.

Comme on l'a vu dans l'activité, il existe des algorithmes qui permettent de trouver tous les états qui satisfont une formule de CTL. Il suffit alors de regarder si l'état de départ en fait partie pour dire avec certitude si notre système satisfait une propriété ou non. Et cette méthode qui consiste à construire un modèle formel d'un système, à exprimer formellement également des propriétés attendues ou redoutées pour le système puis à lancer un programme qui nous dit automatiquement et avec certitude si le modèle satisfait la propriété s'appelle le *model checking* (vérification de modèles).

Bien évidemment plus le système est grand et plus la propriété est longue, plus l'algorithme de *model checking* va prendre de temps, mais il permet en temps fini de vérifier des propriétés d'un système qui a un nombre infini de chemins (on peut boucler sur les états qu'on veut dans l'ordre qu'on veut et on ne s'arrête pas). L'autre contrainte de cette méthode est qu'elle nécessite de construire un modèle formel de notre système (ici le plan du château)[1] et de la propriété (notre formule logique). Cela signifie que si on se trompe en créant le modèle, les vérifications effectuées dessus ne servent à rien. On a donc encore besoin d'humains pour créer ce modèle, et s'assurer qu'il correspond bien à la réalité, ainsi que pour écrire la propriété comme une formule logique. Le reste par contre est fait par des logiciels spécialisés.

1. Il existe des travaux visant à appliquer les algorithmes de *model checking* directement sur un programme. Cela évite de créer un modèle à la main, et donc d'y introduire potentiellement des erreurs, mais le programme contient souvent trop de détails, non nécessaires pour la vérification, et qui rendent la vérification impossible en un temps raisonnable.

Annexe A : solution de l'exercice sur les formules à traduire

1) → · 

2) → · ( ET )

3) → · ( ET ( OU ))

4)  · 

5) → 

6)  

7) → 

8) → ( OU ())

9)   OU  OU 

10)    ou encore NON → · ( ET )

11)  · 

12) → 

13) → · 

14)  

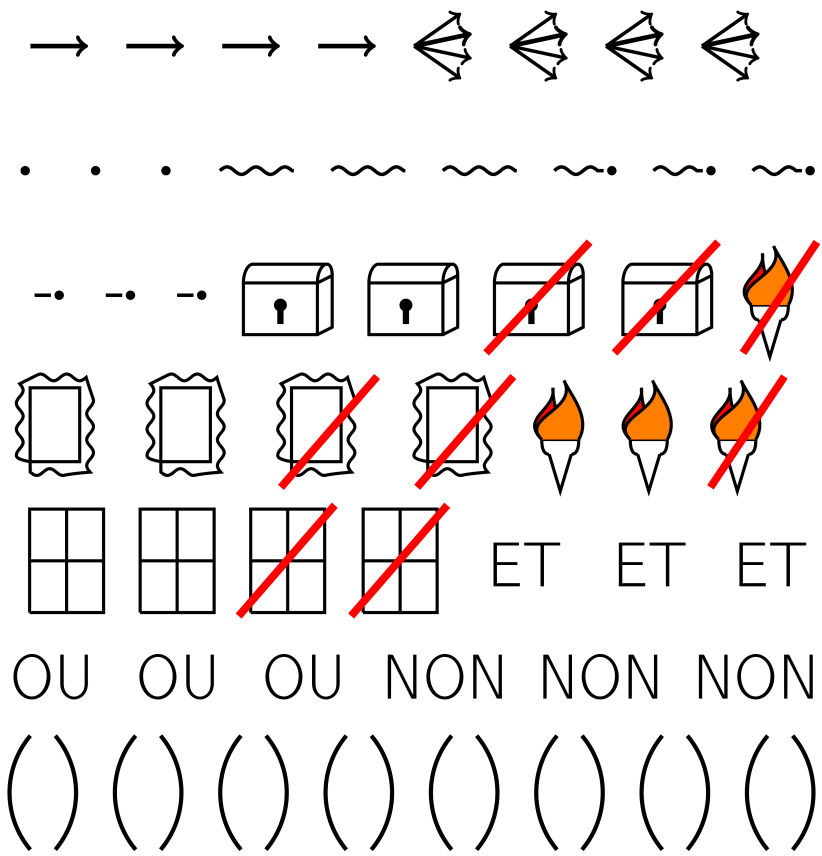
15) → ·  

16) → (( OU )  (ET ( OU)))

17)   ())

18) → ((   ))

Annexe B : les opérateurs (imprimer à 141% et découper)



→ : il existe un chemin tel que...
 : pour tout chemin on a...
. : un jour le long de ce chemin...
 : dans toutes les pièces de ce chemin...
 : dans la pièce suivante...
 : le long du chemin on a... jusqu'à ce qu'on aie...

Annexe C : le château (imprimer à 141 % et plastifier)