Welcome to Animal Crossing New Horizons EST NP-DIFFICILE !

Bien entendu, dire qu'Animal Crossing est NP-Difficile est un abus de langage... Cependant, les mécaniques du jeu sont assez complexes pour rapidement faire émerger de tels problèmes ! Voyons cela !

Définition

La classe NP est celle des langages décidés par une machine de Turing non déterministe en temps polynomial. Un problème de décision est dit NP-Difficile si on peut, par réduction polynomiale, le ramener à tout problème dans NP.

Animal Crossing

C'est un jeu sorti sur Nintendo Switch en 2020, dans lequel les joueurs peuvent construire l'île de leur rêve et faire tout un tas d'autres activités !

Dans le jeu il est notamment possible d'aller visiter l'île d'un ami. Ce dernier peut placer des barrières dans son île afin d'empêcher le visiteur d'accéder à certains endroits. Ainsi il peut paraître naturel de se demander si, lors d'une visite, il est possible de visiter toute l'île entière ! Question facile ? Nous allons en fait montrer que cette question est un problème NP-Difficile en le réduisant à l'incontournable problème SAT !

ENTRÉE Une formule φ sous FNC du calcul propositionnel.

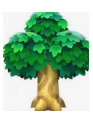
SORTIE OUI si et seulement si φ est satisfiable, c'est-à-dire s'il existe une valuation des variables apparaissant dans φ qui la rende vraie.

Exemple : $(A \vee \neg B \vee C) \wedge (\neg A \vee B \vee D) \wedge (\neg A \vee \neg D)$ est-elle satisfiable ?

Mécaniques utiles du jeu

 Un mur ne peut pas être franchi par le visiteur

 Un buisson bloque le passage, il peut être déterré

 Un arbre bloque le passage. Si le visiteur mange un fruit  , il peut enlever un (seul) arbre

Module de croisement

Nous allons utiliser les mécaniques du jeu vues précédemment pour fabriquer différents modules. Les murs permettront de faire des chemins qui guideront le visiteur, et pour faire des croisements nous mettrons une flaque d'eau au milieu (pour forcer le visiteur à sauter par dessus) .

Module de non-retour

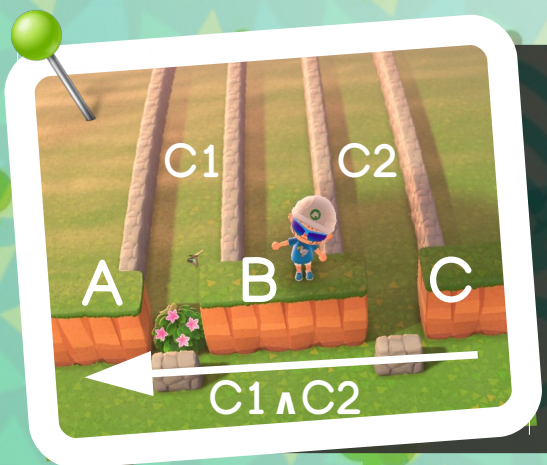
Il nous faut créer de l'irréversibilité !
Le jeu essaye d'empêcher ce genre de phénomène, mais avec la configuration ci-contre, il est possible d'aller de la rive gauche à la rive droite grâce à une perche. Mais ensuite, impossible de revenir à gauche !

Module de choix de variable

Une fois le module de non-retour franchi, le visiteur est bloqué. Pour avancer, il est contraint de manger le fruit disponible et d'enlever exactement un des deux arbres, symbolisant la valuation d'une variable à VRAI ou FAUX. Ce choix débloque certains chemins uniques et est définitif.

Module porte logique OU

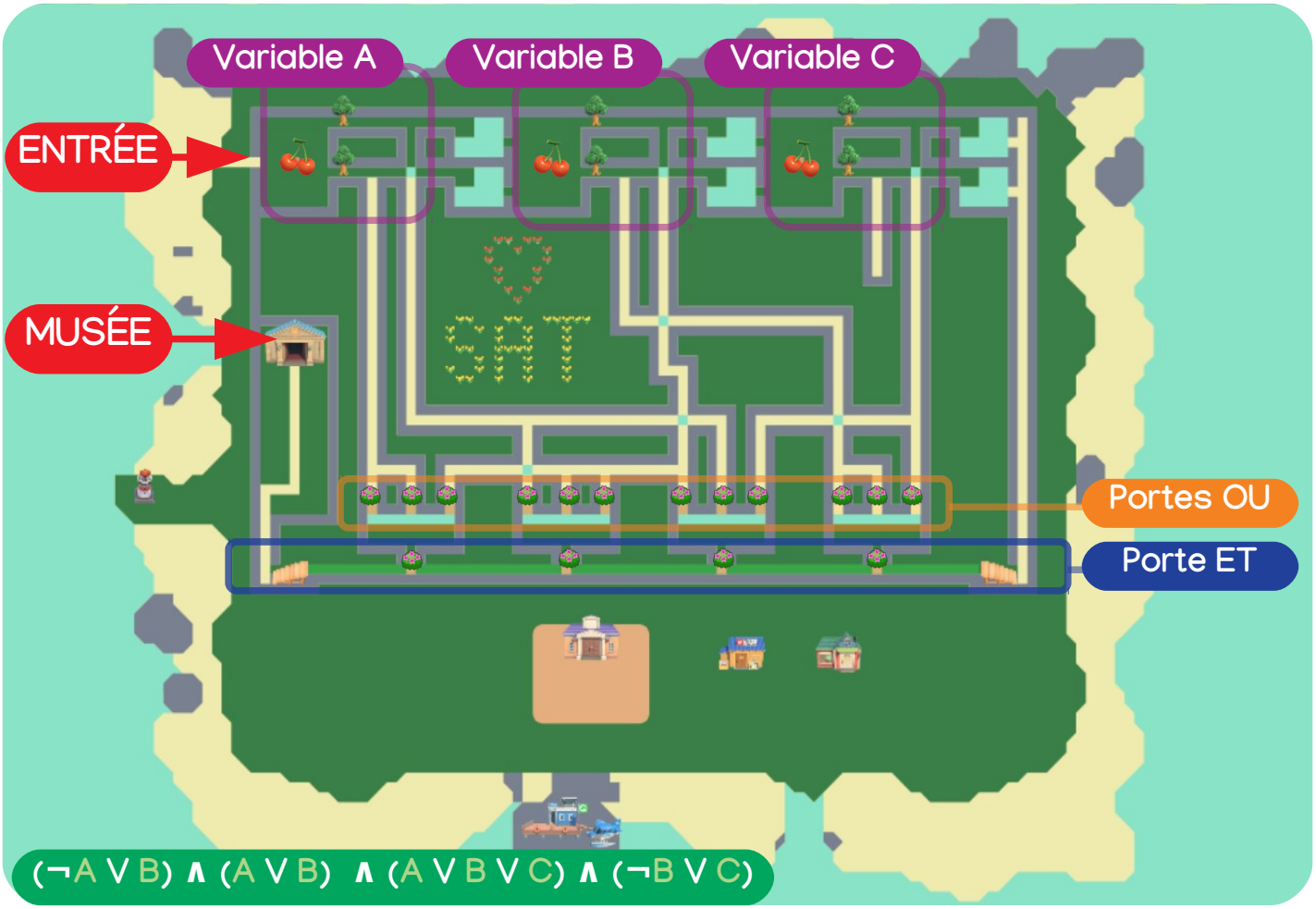
Pour enlever le buisson du bas, il faut que le chemin C1 OU C2 OU C3 soit accessible, pour pouvoir venir de là. Ainsi, on peut enlever le buisson en descendant (ici C1), puis sauter par dessus la flaque d'eau pour atteindre le buisson du bas. En remontant, on ne peut pas accéder à un chemin non visité.



Module porte logique ET

Lorsque le visiteur est sur le chemin horizontal surélevé, il ne peut pas descendre. Pour aller au bout du chemin, il faut traverser les trous. Pour cela il peut sauter par dessus, mais seulement s'il n'y a pas de buisson qui occupe le trou ! Par exemple, dans l'image ci-contre on peut aller de C à B mais pas de B à A. Il faut donc avoir enlevé le buisson de gauche ET de droite pour passer.

En combinant les modules précédents, on peut désormais former une île avec un labyrinthe équivalent à une formule sous FNC ! Voici le résultat !



Explications !

Le visiteur entre, puis mange le fruit et ne peut déterrer qu'un seul des deux arbres (par convention, celui du haut représente la valeur VRAI et celui du bas la valeur FAUX pour la variable en question). Ce choix ne libère qu'un des deux chemins, et donne l'accès aux clauses dans lesquelles le littéral en question intervient (les portes OU du bas). Puis les deux chemins passent un point de non-retour et se rejoignent pour la valuation de la variable suivante. Une fois toutes les variables évaluées, le visiteur doit traverser la porte ET finale, ce qui n'est possible que si les buissons de toutes les clauses ont été déterrés !

Soit une formule φ de la logique propositionnelle sous FNC : on peut en temps polynomial construire une île qui représente cette formule grâce à nos modules. Un visiteur ne pourra atteindre le musée que s'il trouve une valuation qui rende φ vraie pour sortir du labyrinthe. Ainsi, s'il était possible de savoir si toute l'île est visitable grâce à un algorithme en temps polynomial, on aurait trouvé un algorithme qui résout SAT en temps polynomial ce qui est absurde !

Savoir si on peut visiter l'île toute entière est donc un problème NP-Difficile !

Bien entendu, supposer des formules de taille arbitrairement grande est matériellement impossible, on suppose pour cela l'île de taille infinie. Je vous encourage à visiter ce blog dont je me suis inspiré http://eljjdx.canalblog.com/archives/2014/11/09/30904067.html