

Classic Nintendo Games are (Computationally) Hard

Greg Aloupis, Erik D. Demain, Alan Guo, Giovanni Viglieta
(10 Février 2015)

Complexité

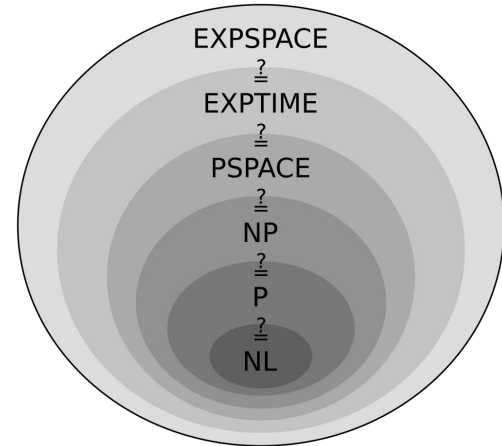
L'**analyse de la complexité d'un algorithme** consiste en l'étude formelle de la quantité de ressources (par exemple de [temps](#) ou d'[espace](#)) nécessaire à l'exécution de cet [algorithme](#). Celle-ci ne doit pas être confondue avec la [théorie de la complexité](#), qui elle étudie la difficulté intrinsèque des [problèmes](#), et ne se focalise pas sur un algorithme en particulier.

Master Theorem

If $T(n) = aT(n/b) + O(n^d)$ for constants $a > 0$, $b > 1$, $d \geq 0$, then

$$T(n) = \begin{cases} O(n^d) & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a \end{cases}$$

Algorithme => Temps d'exécution
Complexité



Problème => Meilleur temps possible
Théorie de la complexité

Complexité

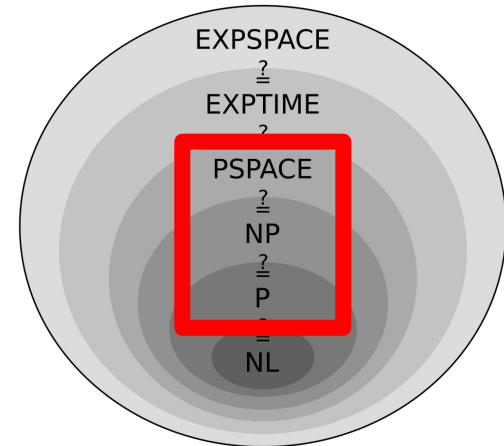
L'**analyse de la complexité d'un algorithme** consiste en l'étude formelle de la quantité de ressources (par exemple de [temps](#) ou d'[espace](#)) nécessaire à l'exécution de cet [algorithme](#). Celle-ci ne doit pas être confondue avec la [théorie de la complexité](#), qui elle étudie la difficulté intrinsèque des [problèmes](#), et ne se focalise pas sur un algorithme en particulier.

Master Theorem

If $T(n) = aT(n/b) + O(n^d)$ for constants $a > 0$, $b > 1$, $d \geq 0$, then

$$T(n) = \begin{cases} O(n^d) & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a \end{cases}$$

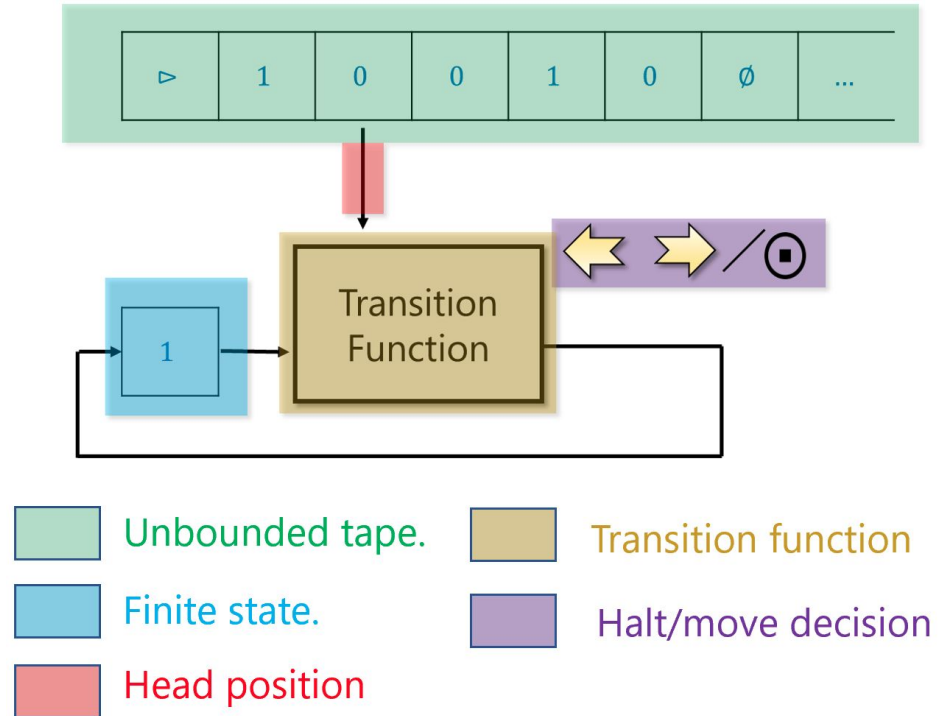
Algorithme => Temps d'exécution
Complexité



Problème => Meilleur temps possible
Théorie de la complexité

Machine de Turing déterministe (DTM)

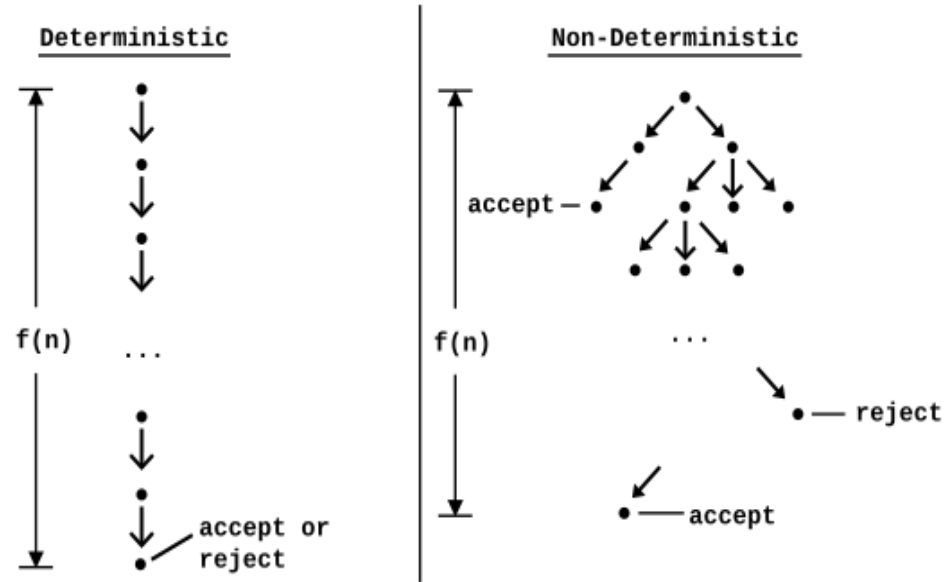
Une machine de Turing (déterministe) est un modèle de calcul mathématique qui manipule des symboles sur un ruban infini en suivant un ensemble de règles appelé fonction de transition.



Machine de Turing non déterministe (NTM)

Il s'agit d'un modèle ABSTRAIT de calcul de machine de Turing pour laquelle la fonction de transition définit plusieurs actions possibles.

Son prochain état n'est donc pas déterminé par son état courant et le symbole qu'elle lit.

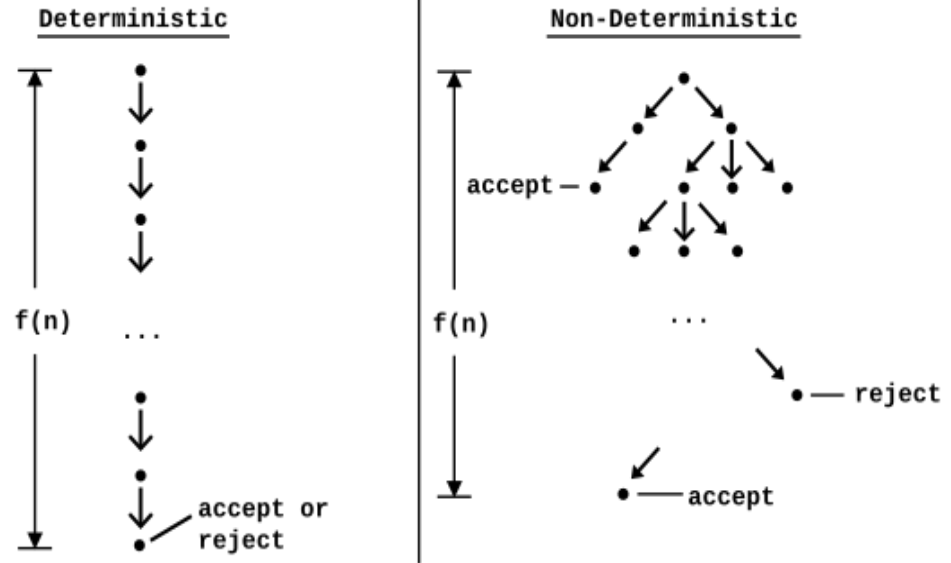


Machine de Turing non déterministe

Tout ce qui peut être fait par une DTM peut l'être par une NTM.

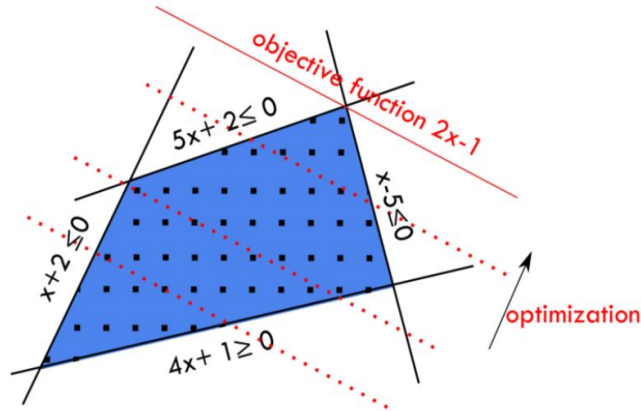
Tout ce qui peut être fait par une NTM peut être fait par une DTM (essentiellement: parcours en largeur des états).

En pratique ce parcours est exponentiel.



P (ou PTIME)

Un problème de décision est dit de classe P, s'il peut être résolu par une machine de turing DÉTERMINISTE en un nombre polynomial d'étapes.



Integer linear programming
 $O(n^2)$

○○○

```
def isPrime(n):  
    if n == 0 or n == 1:  
        return False  
  
    #iterate from 2 to n-1 to check the divisibility of n.  
    for i in range(2, n):  
        #If n is divisible by i, return false  
        if n % i == 0:  
            return False  
  
    return True
```

(Exact) Primality test
 $\not\sim O(N^6)$

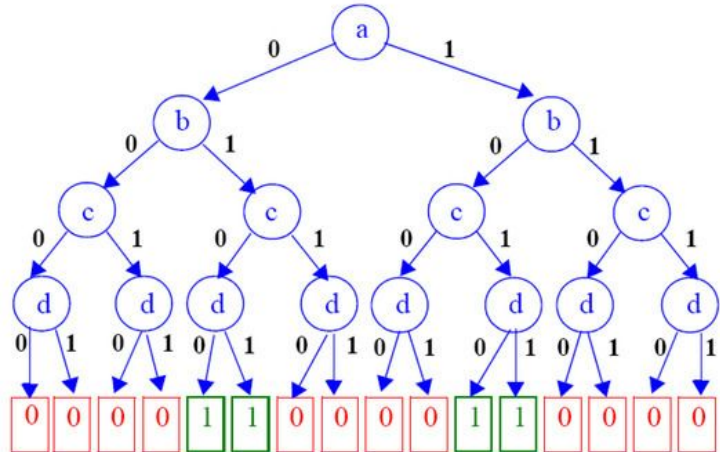
NP : Non-deterministic polynomial

Un problème de décision est dit de classe NP, s'il peut être résolu par une machine de turing NON DÉTERMINISTE en un nombre polynomial d'étapes.

Exemple: (k)-SATisfiability

Est-ce qu'une expression booléenne en forme normale conjonctive admet une solution ?

$$f(a, b, c, d) = (a \vee b \vee c) \cdot (a \vee b \vee \bar{c}) \cdot (\bar{a} \vee c \vee d) \\ (\bar{a} \vee c \vee \bar{d}) \cdot (\bar{b} \vee \bar{c} \vee d) \cdot (\bar{b} \vee \bar{c} \vee d)$$



Problèmes NP-complets

En [théorie de la complexité](#), un **problème NP-complet** ou **problème NPC** (c'est-à-dire un problème [complet](#) pour la classe [NP](#)) est un [problème de décision](#) vérifiant les propriétés suivantes :

- il est possible de vérifier une solution efficacement (en [temps polynomial](#)) ; la classe des problèmes vérifiant cette propriété est notée NP ;
- tous les problèmes de la classe NP se ramènent à celui-ci via une [réduction polynomiale](#) ; cela signifie que le problème est au moins aussi difficile que tous les autres problèmes de la classe NP.

Un problème **NP-difficile** est un problème qui remplit la seconde condition, et donc peut être dans une classe de problème plus large et donc plus difficile que la classe NP.

PSPACE (=NPSACE)

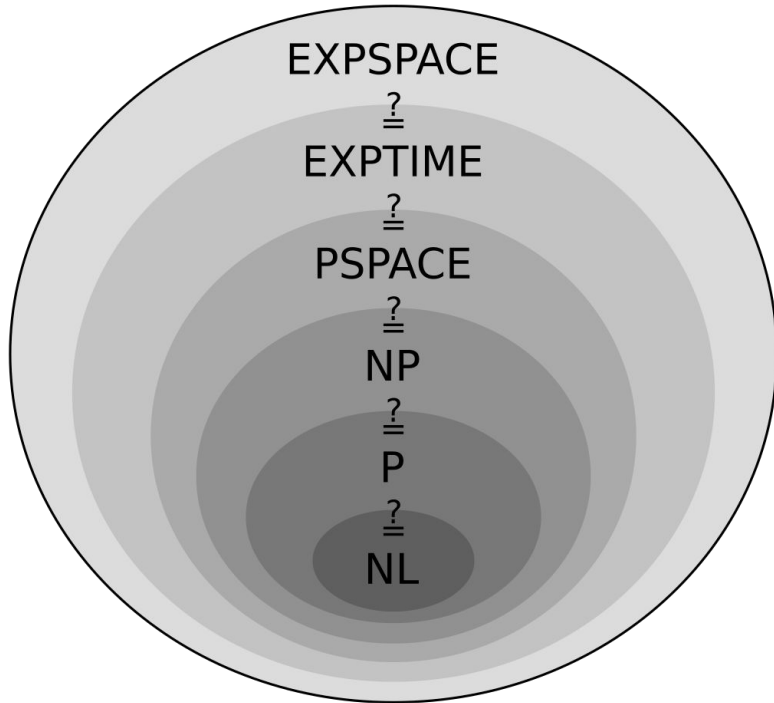
Un problème de décision est dit de classe PSPACE, s'il peut être résolu par une machine de turing (NON)DÉTERMINISTE avec un espace polynomial.

Note: une machine de turing ne peut pas utiliser plus d'espace que de temps. En N étapes, elle visite au plus N + 1 case de son ruban. L'espace peut être réutilisé, pas le temps.

Problème type : True Quantified Boolean Formulas

$$\forall x \exists y \exists z ((x \vee z) \wedge y)$$

Space hierarchy theorem



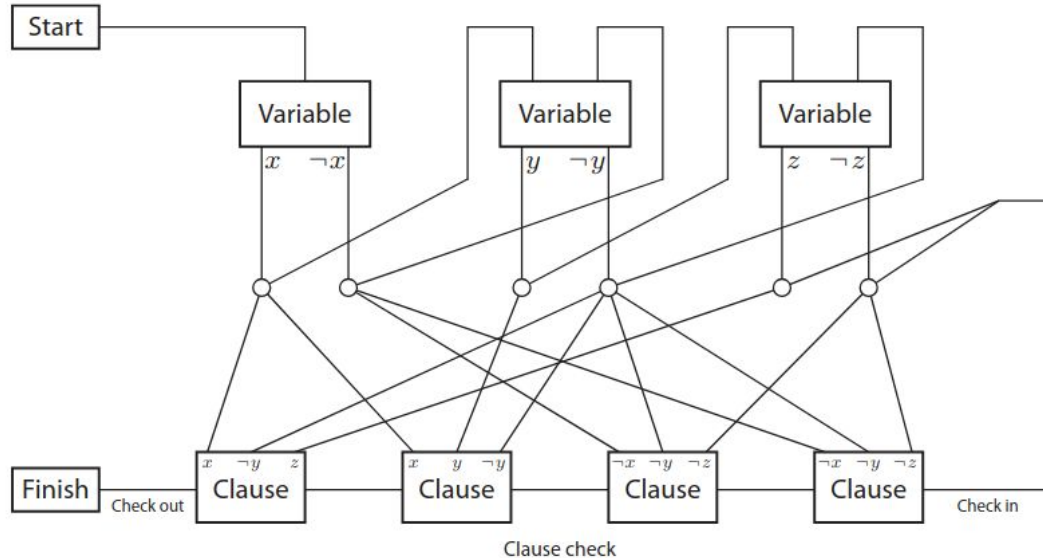
Mais ils ne peuvent pas être tous égaux en même temps:

- NL est inclu (strict) dans PSPACE
- PSPACE est inclu (strict) dans EXPSPACE
- P est inclus (strict) dans EXPTIME

Les jeux nintendo sont NP-difficile

Problème: est-il possible de finir un niveau ?

L'idée générale: réduction vers le problème 3-SAT (NP-complet).



Il faut donc 6 modules:

- Début
- Fin
- Chemin à sens unique
- Variable
- Croisement
- Vérification

Figure 1: General framework for NP-hardness

Super-Mario Bross

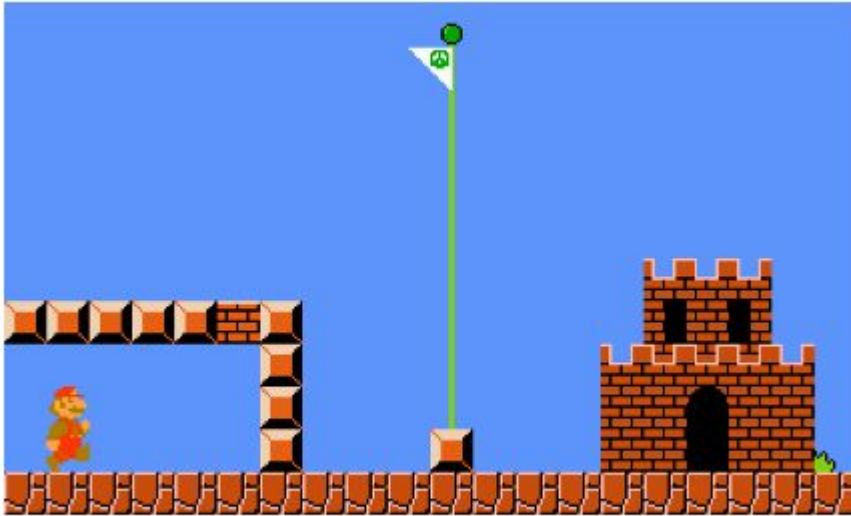
Module de début:



On force mario à prendre un champignon, cela aide pour la vérification. En forme grande, mario fait deux blocs de haut et peut casser les blocs.

Super-Mario Bros

Module de fin:

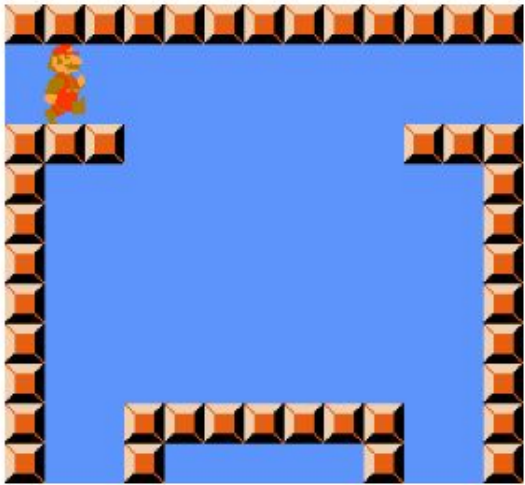


La fin n'est accessible que si mario est grand.

Figure 9: Finish gadget for Super Mario Bros.

Super-Mario Bros

Module de variable:

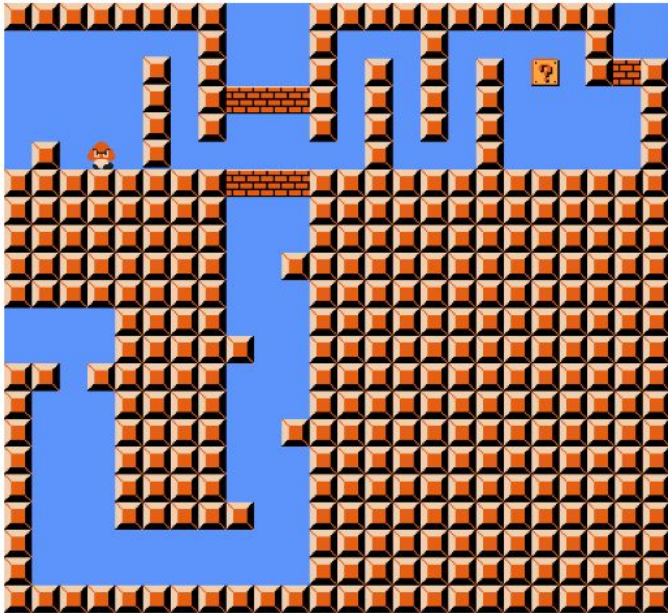


Mario ne peut aller que en bas à gauche (variable = vrai) ou droite (variable = fausse).

Figure 10: Variable gadget for Super Mario Bros.

Super-Mario Bross

Module de croisement:



En entrant d'en haut à gauche, mario peut se faire toucher par le Gumba (pour devenir petit et passer). Il ne peut pas aller en haut car petit mario ne peut pas casser les briques.

En rentrant du bas (en grand mario), il ne pourra ni aller à droite ni à gauche car trop grand, mais pourra casser les blocs.

Super-Mario Bross

Module de vérification:

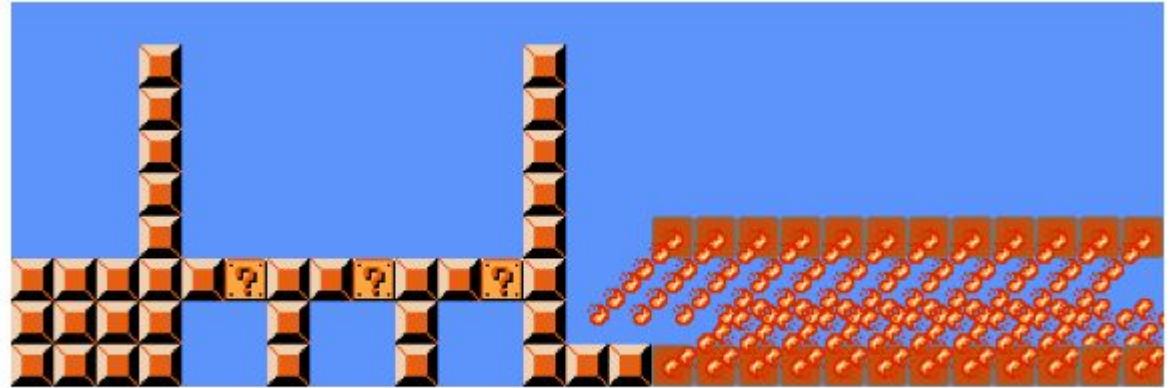


Figure 11: Clause gadget for Super Mario Bros.

Mario rentre par l'un des trois chemins du bas, et tape le bloc ce qui libère une étoile.

Lors de l'étape de la vérification, il entre par en haut et si une étoile est présente, il peut passer par le chemin du bas.

Zelda

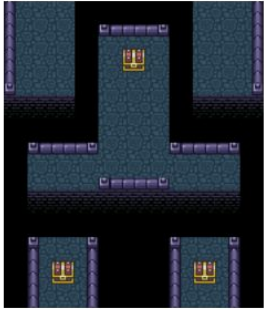


Figure 25: Variable gadget for Zelda

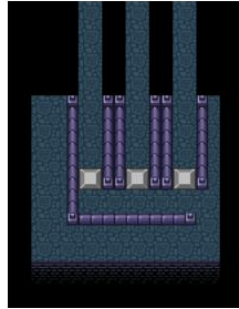


Figure 26: Clause gadget for Zelda

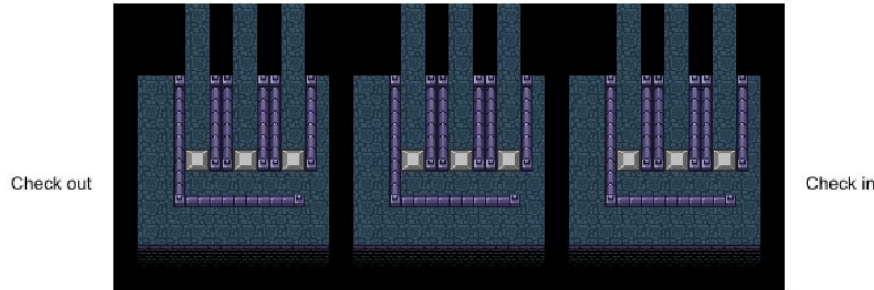
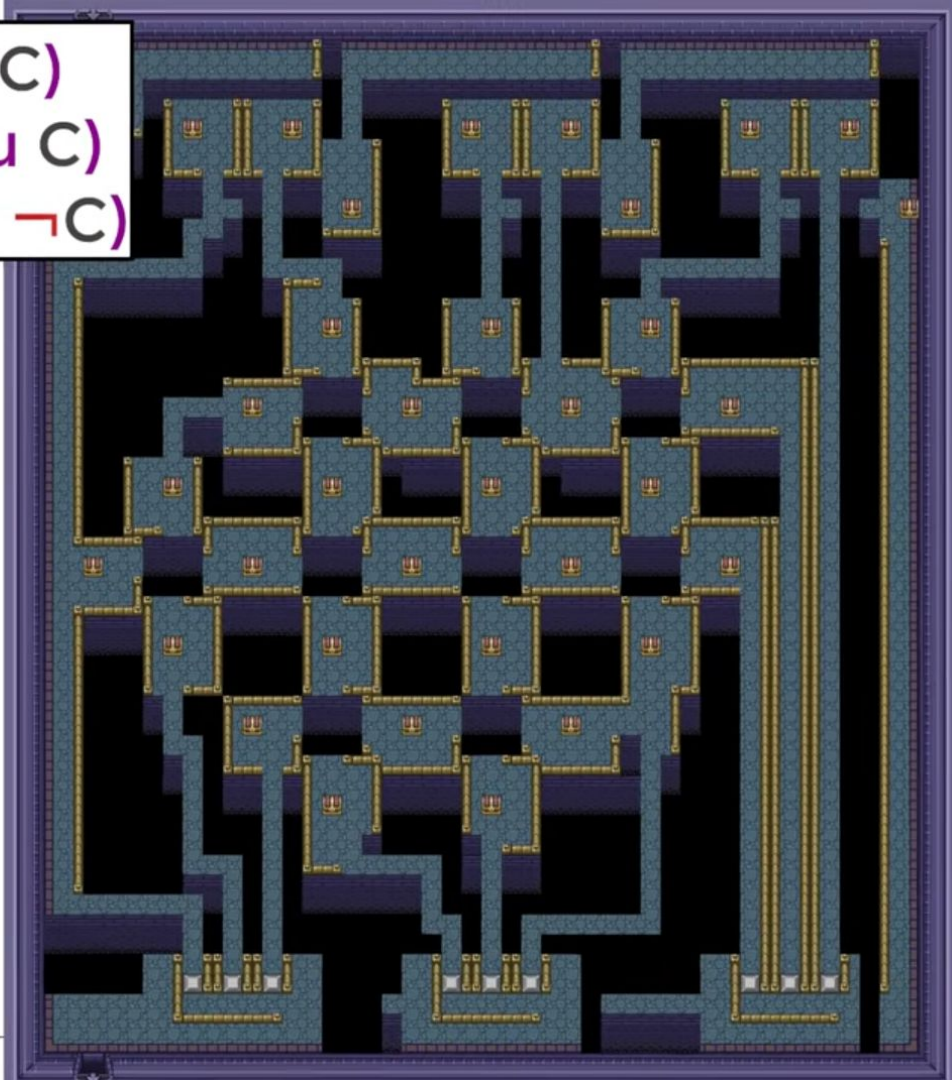
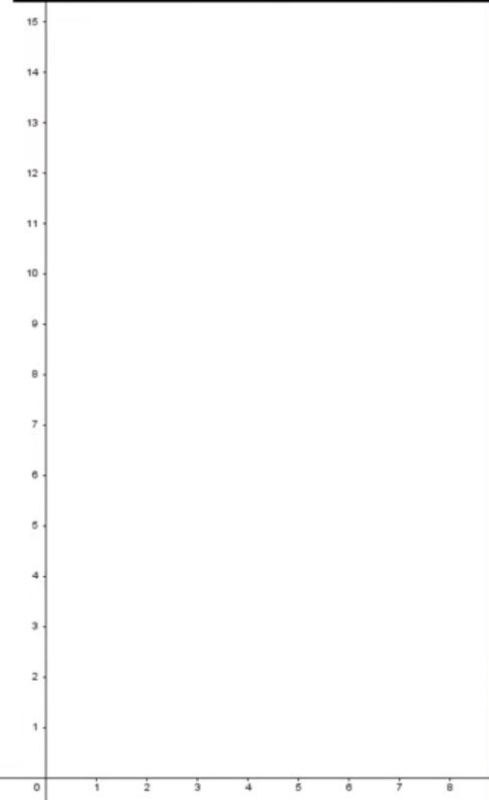


Figure 27: Check path for Zelda (with three clauses)

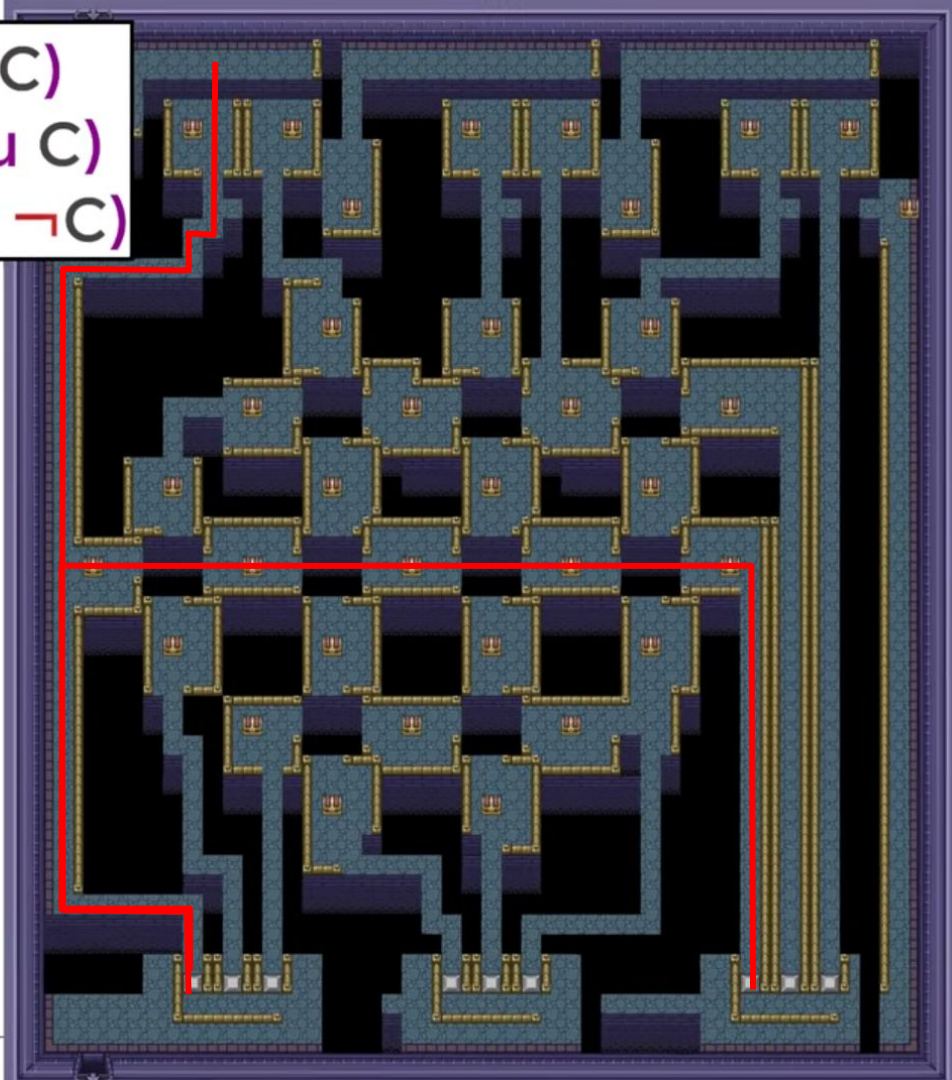
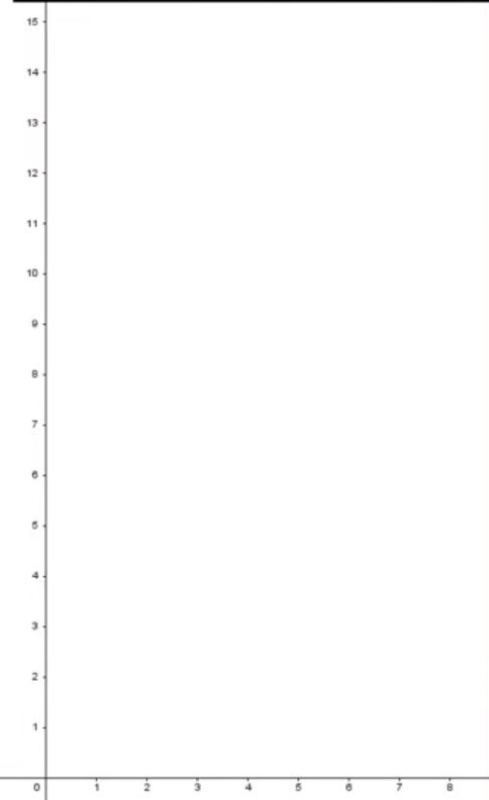
Link peut lancer son grappin sur les coffres afin de passer d'un endroit à l'autre.

Link peut pousser les pierres, afin de bloquer le passage et permettre à Zelda de passer avec le grappin.

$(A \text{ ou } \neg B \text{ ou } C)$
et $(\neg A \text{ ou } B \text{ ou } C)$
et $(A \text{ ou } \neg B \text{ ou } \neg C)$



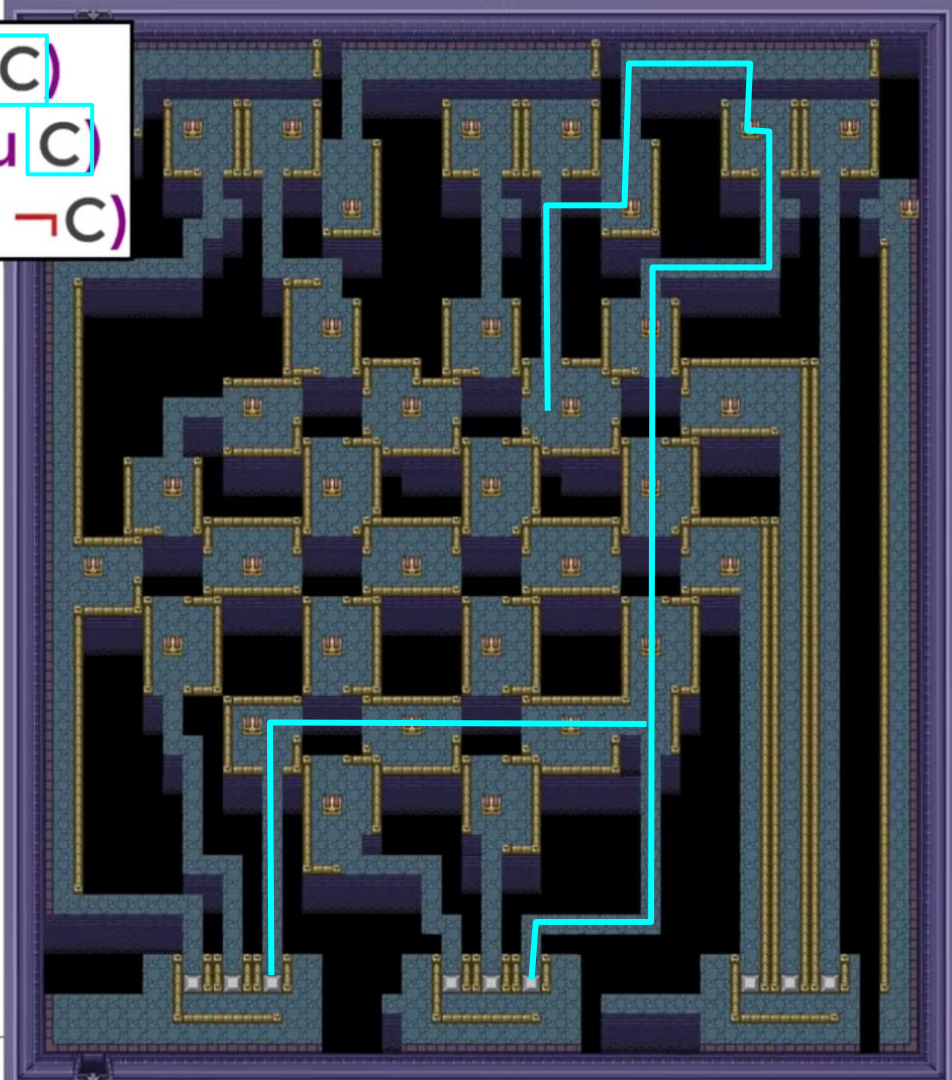
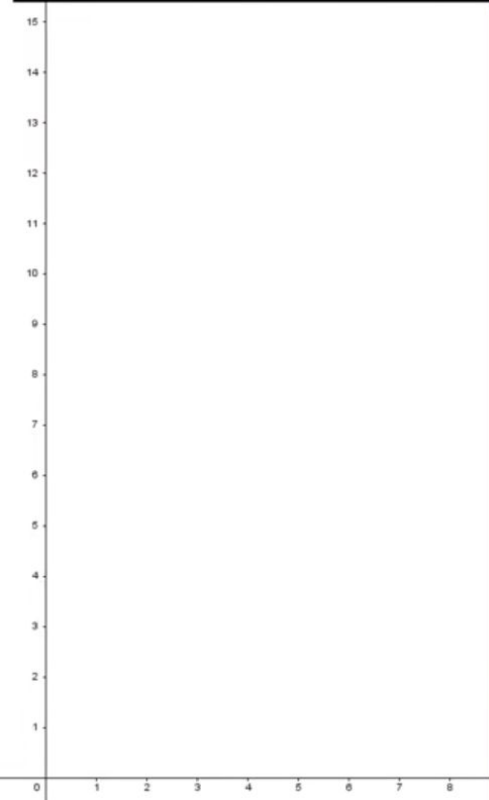
(A ou $\neg$ B ou C)
et ($\neg$ A ou B ou C)
et (A ou $\neg$ B ou $\neg$ C)



A blank coordinate system with a vertical y-axis labeled 1 to 14 and a horizontal x-axis labeled 0 to 8. The axes are represented by solid black lines. The y-axis has tick marks at every integer from 1 to 14, with the label '0' at the origin. The x-axis has tick marks at every integer from 0 to 8. The grid area is empty.



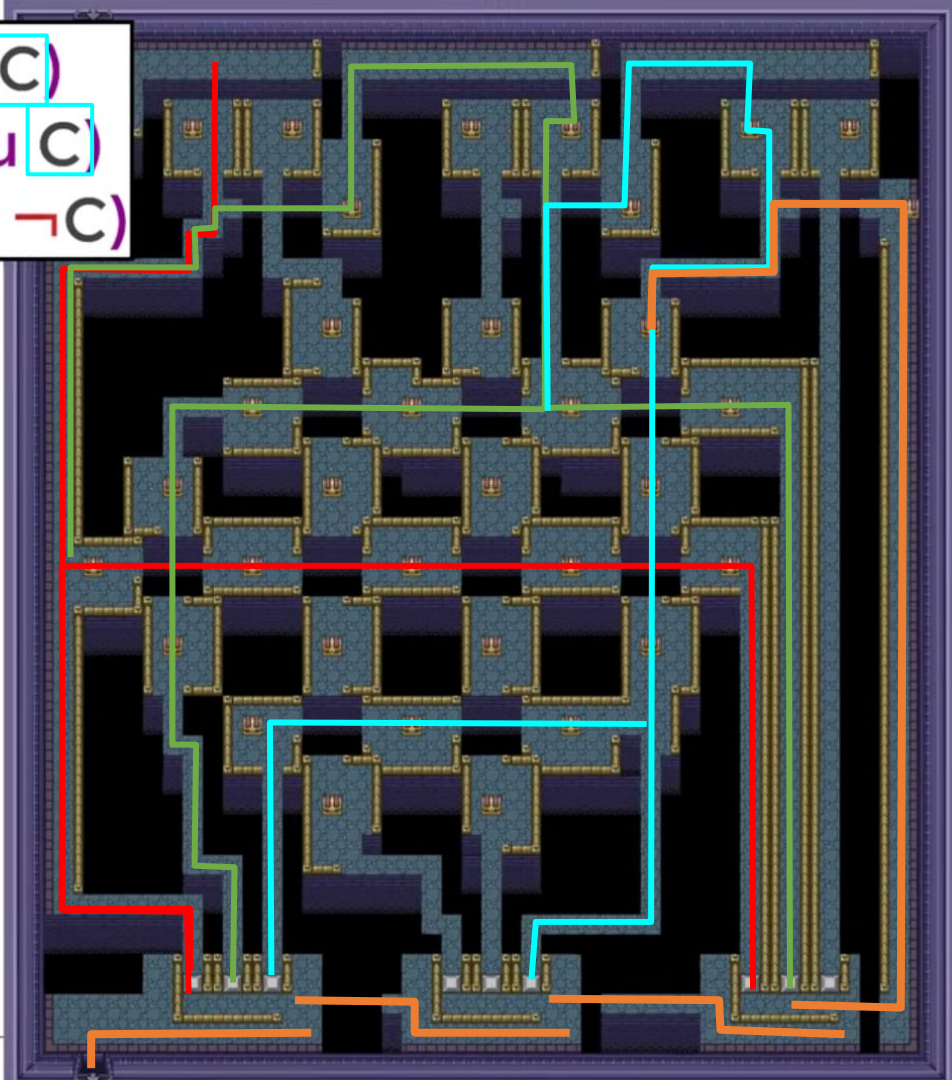
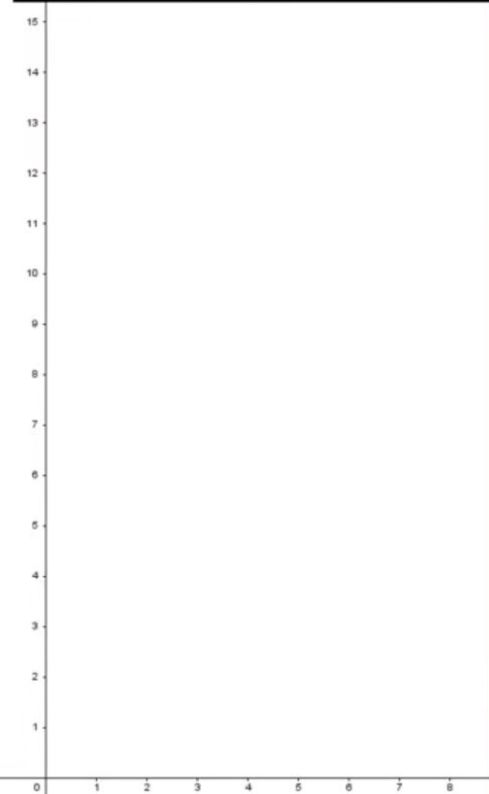
$(A \text{ ou } \neg B \text{ ou } C)$
et $(\neg A \text{ ou } B \text{ ou } C)$
et $(A \text{ ou } \neg B \text{ ou } \neg C)$



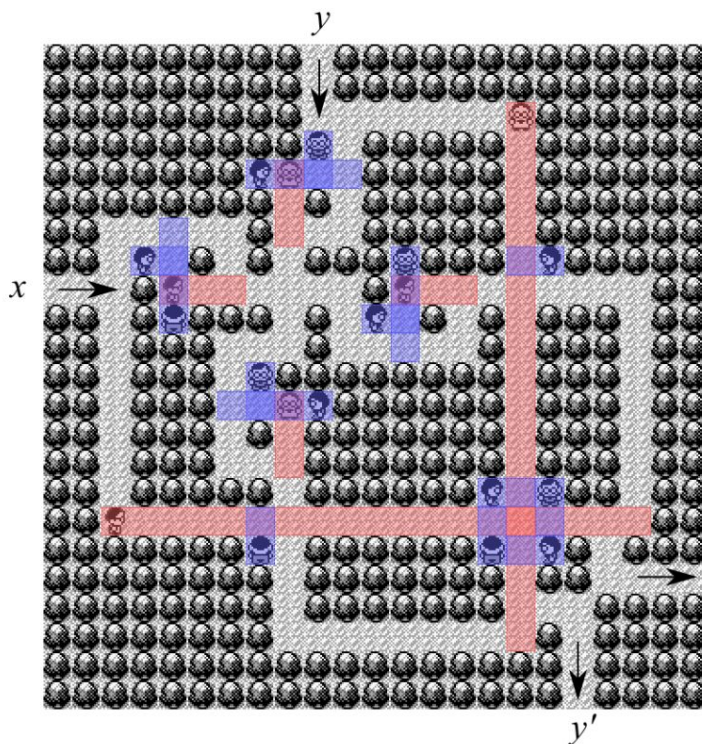
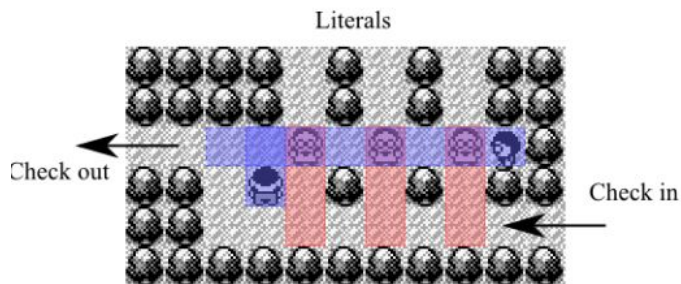
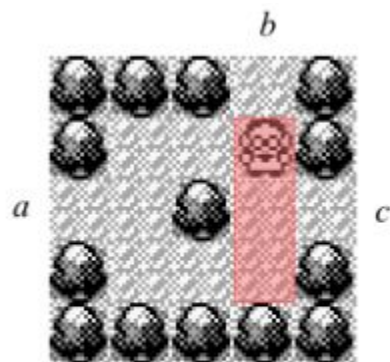
A blank coordinate system with a vertical y-axis labeled 1 to 15 and a horizontal x-axis labeled 0 to 8. The y-axis is on the left, and the x-axis is at the bottom. There are no data points or lines plotted.



$(A \text{ ou } \neg B \text{ ou } C)$
et $(\neg A \text{ ou } B \text{ ou } C)$
et $(A \text{ ou } \neg B \text{ ou } \neg C)$



Pokémon

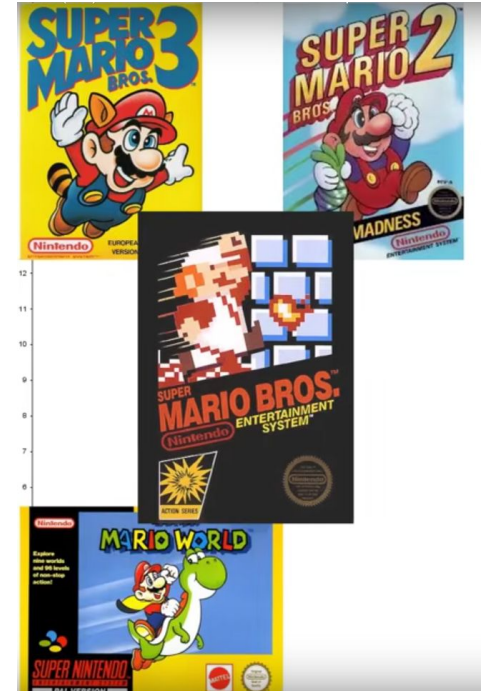
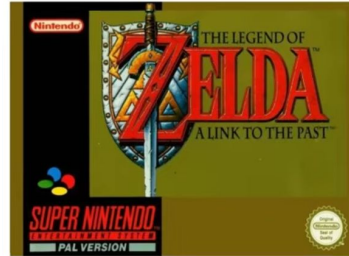


Bleu = dresseur fort
(jamais battu)
Rouge = dresseur
faible (toujours
battu).

Parler à un
dresseur change
son orientation et
permet le passage.

Figure 34: Clause gadget for Pokémon

Conclusion : les jeux nintendo sont NP-difficiles



Les jeux nintendo sont PSPACE-difficile ?

Même idée, mais il faut un module en plus:

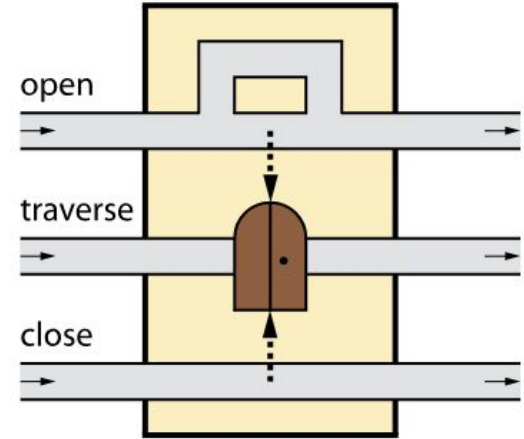
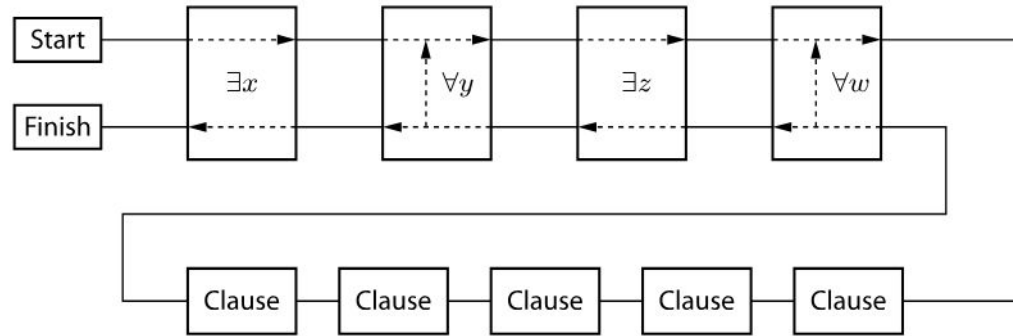


Fig. 2. Door gadget.

Pour les quantifieurs universels: on les mets à true premier passage, puis on force le joueur à les prendre et les mettre à false au deuxième passage

Mario Kart:

Mario Kart is Hard

Jeffrey Bosboom

Erik D. Demaine

Adam Hesterberg

Jayson Lynch

Erik Waingarten

Massachusetts Institute of Technology

Abstract

Nintendo's Mario Kart is perhaps the most popular racing video game franchise. Players race alone or against opponents to finish in the fastest time possible. Players can also use items to attack and defend from other racers. We prove two hardness results for generalized Mario Kart: deciding whether a driver can finish a course alone in some given time is NP-hard, and deciding whether a player can beat an opponent in a race is PSPACE-hard.

NP-hard: même chose, on utilise les carapaces et des bananes

PSpace-hard: Un joueur s'occupe des variables existentielles, l'autre des universelles et lance des bananes pour bloquer des chemins.

FIN