Théorie des Catégories et Critère de Contrôlabilité pour Atari 2600

Lionel Cordesses, Omar Bentahar, Kevin Poulet, Aude Laurent, Sarah Amar, Thomas Ehrmann, Ju Page

Théorie des Catégories et Critère de Contrôlabilité pour Atari 2600

L. Cordesses[1] O. Bentahar[1] K. Poulet[1] A. Laurent[1] S. Amar[1] T. Ehrmann[1] J. Page[2]

[1] Renault Innovation Silicon Valley, 1215 Bordeaux Drive, Sunnyvale, 94089 CA, USA
[2] Université Paris Diderot-CNRS, Laboratoire SPHERE, 5 rue Thomas Mann, 75205 Paris Cedex 13, France

{lionel.cordesses, kevin.poulet, thomas.ehrmann}@renault.com,
{aude.laurent, sarah.amar, omar.bentahar}@nissan-usa.com,
ju.page@hotmail.fr

Résumé

L'objectif de l'apprentissage par renforcement est de contrôler un agent ou système dynamique dans le but de tirer la meilleure récompense de son environnement. Toutefois, dès lors que l'on prend en compte des contraintes réalistes telles qu'un nombre restreint d'expériences ou une puissance de calcul limitée, les approches classiques perdent de leur efficacité.
Nous introduisons des éléments de la théorie du contrôle pour diminuer le nombre d'essais requis. En parallèle, des outils mathématiques issus de la théorie des catégories nous permettent de formaliser une approche d'apprentissage automatique innovante reposant sur la formalisation d'analogies entre problèmes. Cette méthode suit une approche scientifique en remettant en cause – si besoin est – ses propres conclusions. Enfin, nous présentons une première application de la théorie des catégories sur les jeux Atari 2600, et illustrons les avantages d'utiliser des analogies pour transférer des connaissances entre jeux vidéo.

Mots Clef

Apprentissage Automatique, Théorie des Catégories, Atari 2600.

Abstract

Reinforcement Learning aims at controlling a dynamic system or agent to get the best reward from its environment. However, when adding real-world constraints such as a limited number of trials and limited computing power, classic and state-of-the-art approaches may perform suboptimally.
We introduce mathematical concepts from control theory related to dynamic systems to alleviate the need for many trials, while formalizing a Machine Learning approach relying on mathematical analogies from category theory. We argue that our system is able to refute its own findings, as it is the case with any scientific approach. We then present what we believe is the first application of category theory to play Atari 2600 games, and we illustrate the use and benefits of analogies to transfer knowledge between games.

Keywords

Machine Learning, Category Theory, Atari 2600.

1 Introduction

Bien qu'elles soient excellentes lorsqu'il s'agit d'apporter des solutions, les méthodes à la pointe de l'apprentissage automatique n'atteignent pas d'aussi bons résultats dès lors que l'on ajoute des contraintes réalistes telles qu'un budget restreint ou des délais limités. Aussi, plus la recherche en Intelligence Artificielle (IA) progresse sur les problématiques du quotidien telles que la conduite automobile, plus la capacité à transférer des connaissances passées à des situations inconnues devient cruciale, puisque cela réduit considérablement le nombre d'essais requis. En effet, bien que les algorithmes les plus récents aient surpassé les humains sur des tâches spécifiques telles que le jeu de Go [26] ou les échecs [4], les humains restent supérieurs aux machines quand il s'agit d'évoluer et de prendre des décisions pertinentes dans un environnement inconnu. De manière générale, les humains savent généraliser les connaissances et expériences passées.
Un autre aspect prépondérant des algorithmes d'IA est leur robustesse. Certains algorithmes d'apprentissage par renforcement, par exemple, perdent leur efficacité dès que les données changent légèrement (un objet déplacé vers le haut de quelques pixels [10], ou une variation légère de la luminance de l'image [14]). Même si entraîner de nouveau le système résout ce genre de problème, cela met en évidence le besoin d'algorithmes d'IA capables de remettre en cause ce qu'ils ont appris, en particulier si les observations contredisent le modèle établi.
Cela soulève aussi la question de l'échelle à laquelle le raisonnement doit se faire, puisque cette remise en cause nécessite une capacité de déduction. Par exemple, si les données d'entrée sont des images, certains considèrent que le pixel est une échelle trop réduite, et que se focaliser sur des amas de pixels est plus pertinent [25]. De la même façon, nous considérons qu'un cadre mathématique fondé sur la *théorie des catégories* apporte des outils rigoureux pour formaliser ces situations où l'utilisation de connaissances

passées donne un avantage. Ce cadre permet de détailler l'exploitation des analogies entre différentes situations, et définit aussi le processus de transfert.

Afin de réduire le nombre d'échantillons, et donc la quantité de données utilisées, nous proposons un concept d'entités. Nos travaux ont commencé dans le but d'obtenir des performances similaires à l'état de l'art, mais en n'utilisant qu'un centième des ressources et du temps habituellement consacrés. Nous validons cette approche sur les jeux Atari 2600. Nous utilisons aussi ces jeux pour valider la capacité à transférer des connaissances non seulement au sein d'un même jeu, mais aussi entre des jeux tactiquement ou visuellement similaires.

2 Travaux antérieurs

Les jeux proposent un éventail de problèmes qui peuvent être modélisés par un environnement fermé et fini. Les jeux vidéo ont eux la particularité de disposer d'un environnement qui évolue en temps réel. De plus, ils obéissent à un ensemble de règles et constituent donc une représentation d'un monde virtuel mais rationnel. Ce faisant, les jeux vidéo sont devenus une référence pour tester les algorithmes. En effet, l'essor de benchmarks vidéoludiques tels que Arcade Learning Environment (ALE) [2] a permis des progrès phénoménaux des méthodes d'apprentissage en temps réel, telles que l'apprentissage par renforcement [8][18]. Ce benchmark comprend une myriade de problèmes différents qui impliquent tous une prise de décision et ont pu être modélisés par de larges processus de décision Markoviens [19]. Aussi, la versatilité de ces problèmes permet de tester différentes méthodes. Les premiers travaux, à l'instar du Deep-Q-Network (DQN) décrit dans [20], utilisent des réseaux de neurones profonds sur des images pré-traitées de l'aire de jeu. Des algorithmes de recherche ou de planification classiques tels que la recherche arborescente Monte-Carlo [22] ou « Iterated Width Algorithm » [15] peuvent aussi être implémentés pour guider le système.

Néanmoins, entraîner un algorithme d'apprentissage profond est très chronophage et demande une quantité importante de données. Par exemple, les conditions expérimentales de [20] demandent 200 millions d'images de jeu. Des méthodes plus exotiques ont été développées afin de diminuer le nombre d'essais nécessaires, par exemple « Inverse Reinforcement Learning » [11] ou l'apprentissage par imitation [3]. Un autre aspect modulable en vue d'une diminution du nombre d'essais requis est l'architecture des réseaux utilisés et *a fortiori* la procédure d'entraînement choisie. Des architectures moins profondes [13] ou hybrides [12] ont été mises en oeuvre pour contourner ces problèmes.

L'humain et la machine comprennent et représentent le monde différemment puisque, contrairement à un algorithme, une personne est capable de transférer et généraliser ses connaissances, ce qui lui permet de s'adapter rapidement à un environnement inconnu. On peut considérer que, dans un premier temps, chaque nouvelle situation est traitée en établissant des analogies entre problèmes. C'est pourquoi on peut prétendre que des algorithmes, qui ne peuvent transférer leurs connaissances de manière aussi intuitive, sont incapables de prendre des décisions pertinentes dès lors que l'environnement change. Toutefois, les techniques d'apprentissage par transfert cherchent à reproduire une caractéristique similaire et à abolir cette frontière, et ont récemment fait des progrès colossaux [1] [7] [16]. Un avantage évident de ces techniques demeure la possibilité de contourner les besoins en expériences en important des connaissances et raisonnements d'autres situations.

Cependant, transférer ses connaissances depuis un autre domaine peut ne pas suffire à atteindre les niveaux de versatilité et d'adaptation de l'être humain. En effet, la façon dont les gens utilisent leurs intuitions pour établir des liens de causalité pourrait être une raison expliquant cette aptitude à transférer des connaissances d'un domaine à un autre pourvu qu'ils partagent des causalités similaires. En plus de ce concept, les « Schema Networks » [10] s'appuient sur des entités plutôt que des images brutes, et font l'hypothèse qu'ils disposent d'un système capable de détecter et traquer ces entités sur l'écran. Dès lors, ils parviennent à réutiliser ce qu'ils ont appris pendant $100\,000$ frames d'entraînement sur le jeu Breakout à des variations de ce même jeu.

3 La Théorie des Catégories comme cadre

Bien que les méthodes présentées en Partie 2 parviennent à contrôler avec succès un agent dans les jeux Atari 2600, ces méthodes nécessitent un grand nombre d'images d'apprentissage. Notre objectif reste d'aboutir à une méthode d'apprentissage automatique qui pourrait apprendre avec peu de ressources, à savoir environ un centième des données et puissance de calcul habituellement requises par les meilleures solutions présentes dans la littérature.

À notre connaissance, de tels prérequis ne peuvent être validés à partir de méthodes existantes. C'est pourquoi nous nous sommes lancés dans la conception d'une nouvelle approche d'apprentissage automatique.

3.1 Construction des entités

Comparablement aux « Schemas Networks » [10], notre travail implique un concept « mésoscopique » d'entités. Ces entités sont en général nos objets du quotidien, ou bien dans le cas des jeux vidéo, tous les objets que l'on rencontre au cours de notre parcours dans cet environnement fermé. Par exemple, dans le jeu Breakout, on peut considérer que la raquette constitue une telle entité puisqu'elle se situe, en terme d'échelle, entre le « microscopique » pixel et la « macroscopique » image. Travailler à cette échelle peut s'apparenter à travailler à l'échelle du morphème en linguistique, *i.e.* le plus petit élément significatif d'un langage [6].

Par construction, à partir d'une analyse des signaux échan-

tillonnés, notre système est capable de détecter et traquer ces entités. Toutefois, le monde qui nous entoure n'est pas seulement constitué d'objets, mais aussi de sujets. Dans les problèmes que nous considérons, nous appelons « Moi » l'entité contrôlée par les actions du joueur. Nous supposons qu'une IA familière avec ces concepts sera plus à même de prendre des décisions pertinentes, et de transférer ses connaissances et raisonnements à tout un panel de situations qui ne partagent qu'une vague similarité.
Le « stade du miroir » est une étape clef du développement de l'identité de soi durant l'enfance. Elle correspond au moment où l'enfant commence à prendre conscience de son propre corps et à le distinguer de celui d'autrui. Nous concevons notre Intelligence Artificielle à partir de ce stade, en distinguant d'abord le « Moi » parmi toutes les entités présentes. Dès lors, les autres entités de notre environnement sont regroupées en deux grandes classes en fonction de leurs intentions, selon qu'elles sont plutôt hostiles (les « ennemis ») ou adjuvantes (les « amis »). Le parallèle est facilement établi avec la construction d'un modèle de récompense lors d'un algorithme classique d'apprentissage par renforcement : les amis apportent des récompenses positives, alors que les ennemis produisent des récompenses négatives.

3.2 Remise en cause des modèles

Les modèles obtenus ne sont pas éternellement valides puisque le « Moi », mais aussi l'environnement évoluent et peuvent changer. Ainsi, nous choisissons de nous orienter vers une approche scientifique telle que décrite par [23], c'est-à-dire qu'il doit être possible pour un système empirique d'être réfuté par le résultat de ses expériences, et concevons notre logiciel de sorte à ce qu'il puisse remettre en cause ses propres résultats, puis reconstruire ses modèles. Cette remise en cause se produit dès lors que l'erreur mesurée entre les observations du système et les prédictions du modèle dépasse un certain seuil.

3.3 Effet mémoire

Au fil des parties d'un même jeu, il est évident qu'un joueur humain retient les enseignements tirés des parties précédentes et sait toujours comment jouer. Notre logiciel mémorise les connaissances qu'il découvre (entités, modèles dynamiques, etc.) et les utilise lors des parties successives.

3.4 Utilisation d'analogies pour le transfert de connaissances entre problèmes

L'un des outils les plus efficaces à notre disposition pour évoluer dans une nouvelle situation est la capacité à faire des analogies entre cette situation et nos expériences passées. C'est pourquoi nous pressentons qu'une IA capable d'établir des analogies et sachant comment jouer au jeu Breakout sera capable de transposer ses capacités au jeu Pong, même si les aires de jeu et les règles ne sont pas identiques, à l'instar d'un joueur de tennis qui saurait comment manier une raquette de badminton même s'il n'a jamais pratiqué le sport.

Les situations où deux problèmes ont exactement le même nombre d'états et des structures isomorphes sont rares. Néanmoins, il existe des outils mathématiques pour identifier des structures non isomorphes tels que l'équivalence de catégories en théorie des catégories [17] [1]. La théorie des catégories est un outil puissant et moderne qui est apparu au milieu du XX$^{\text{ème}}$ siècle pour permettre le transfert de théorèmes entre différentes branches des mathématiques.
Dans les problèmes qui reposent sur la théorie des ensembles, l'identification se réduit aux relations d'identité et aux cas bijectifs, tandis que la théorie des catégories apporte des descriptions plus riches des objets au travers de l'introduction de flèches (ou morphismes) entre objets, ce qui permet de nouvelles sortes d'identifications. La plupart des outils d'apprentissage automatique s'appuyant sur la théorie des ensembles, et non sur la théorie des catégories, nous illustrons ci-dessous la plus-value d'un tel cadre mathématique.

Théorie des catégories. Une catégorie $\mathcal{C}$ est une collection d'objets et de morphismes entre certains de ces objets, munie d'une composition de morphismes, et peut ainsi être assimilée à un graphe orienté. Si A et B sont des objets de $\mathcal{C}$, la flèche $a : A \to B$ est un isomorphisme si elle est inversible, *i.e* s'il existe une flèche $b : B \to A$, telle que $ba = Id_A$ et $ab = Id_B$. Dans ce cas, les objets A et B sont isomorphes. Les isomorphismes définissent une relation d'équivalence sur la classe des objets de $\mathcal{C}$. On note $\mathcal{C}/\simeq$ son quotient. Aussi, si $F : \mathcal{C} \to \mathcal{C}'$ est ce qu'on appelle une équivalence de catégories, celle-ci induit une bijection $F' : (\mathcal{C}/\simeq) \to (\mathcal{C}'/\simeq)$ entre les classes d'objets isomorphes même si F elle-même n'est pas bijective. Dès lors, on n'identifie plus les objets (ou états) un à un entre deux situations, mais les types (ou classes d'isomorphisme) de ces états.
Ce procédé peut être utilisé dans le cas de problèmes observables. En effet, considérons deux ensembles d'états non vides $\mathcal{C}$ et $\mathcal{C}'$ sans autre hypothèse sur leur cardinalité. Supposons aussi l'existence de deux fonctions d'observations $f : \mathcal{C} \to O$ et $f' : \mathcal{C}' \to O'$. Quitte à restreindre O et O', supposons que ces deux fonctions sont surjectives. Pour tout $o \in O$, on dit que les états $x \in \mathcal{C}$ dont l'observation associée est o (*i.e* $f(x) = o$) sont de type T_o. Cela définit une relation d'équivalence R_f sur l'ensemble $\mathcal{C}$: $\forall x, y \in \mathcal{C}, x\ R_f\ y$ si et seulement si (ssi) $f(x) = f(y)$. Transposé dans le champ lexical de la théorie des catégories, cela revient à placer une flèche inversible entre deux objets x et y de $\mathcal{C}$ ssi $x\ R_f\ y$. $\mathcal{C}$ devient alors une catégorie, où toutes les flèches sont inversibles et telle que $\mathcal{C}/\simeq$ est exactement le quotient $\mathcal{C}/R_f$, quotient qui définit aussi l'ensemble des types d'états de $\mathcal{C}$. Puisque ces types ont été définis via les observations, la surjection $f : \mathcal{C} \to O$ induit une bijection $\tilde{f} : (\mathcal{C}/\simeq) \to O$ entre l'ensemble de types et l'ensemble d'observations. $\tilde{f}$ est donc l'inverse de la fonc-

1. Une autre approche pourrait être l'utilisation de la théorie des modèles, plus spécifiquement du concept d'équivalence élémentaire [5] dans la perspective de doter l'IA de raisonnements logiques.

tion $T : O \to (\mathcal{C}/\simeq)$, $o \mapsto T_o$ qui définit les types T_o. Le même raisonnement peut être reproduit à partir de $\mathcal{C}'$ et de f'.

Enfin, supposons qu'il existe une bijection $G : O \to O'$ entre les ensembles d'observation et que O et O' sont donc de même cardinalité. Ainsi, on peut définir une autre bijection $F' = \tilde{f}'^{-1} \circ G \circ \tilde{f} : (\mathcal{C}/\simeq) \to (\mathcal{C}'/\simeq)$ entre les ensembles de types d'objets, bijection en réalité induite par l'équivalence de catégories $F : \mathcal{C} \to \mathcal{C}'$ définie comme suit : pour tout $x \in \mathcal{C}$, soit $o = f(x)$ et soit $x' \in f'^{-1}(G(o))$, définissons F telle que $F(x) = x'$. Si $\mathcal{C}$ et $\mathcal{C}'$ ont des cardinaux différents, F ne peut être bijective. Par construction, F envoie chaque état x vers un état x' du même type (modulo G). Cela permet de relier les catégories $\mathcal{C}$ et $\mathcal{C}'$, de telle sorte que si l'on dispose d'une stratégie applicable dans $\mathcal{C}$, elle peut être transposée dans $\mathcal{C}'$ par la fonction F.

Une présentation plus détaillée et approfondie de la théorie des catégories peut être trouvée dans [17]. Un exemple de transfert de stratégie grâce à la théorie des catégories peut être trouvé en annexe de cet article.

Similarités entre paires d'entités mesurées grâce à la théorie des catégories. Les humains n'ont aucune difficulté à remarquer les similarités entre différentes situations. Nous montrons dans cette partie comment formaliser cette recherche de similarités, même lorsque les jeux sont complètement différents. Cela nous permettra de transférer des connaissances d'un jeu à un autre.

Nous avons identifié chaque situation par son espace d'états. D'autres paramètres auraient pu être pris en compte, comme le nombre d'entités et leurs interactions possibles. Considérons par exemple un jeu $\mathcal{G}$ et son espace d'états $\mathcal{S}$. Si $\mathcal{G}$ contient n entités visibles, $\mathcal{S}$ pourrait être un sous-ensemble d'un certain produit $\prod_{i=1}^{n} \mathcal{S}_i$, où chaque $\mathcal{S}_i$ serait l'espace d'états d'une entité E_i — ce serait un sous-ensemble strict car, en général, deux entités distinctes ne peuvent pas être au même endroit au même moment. Nous pouvons aussi associer à $\mathcal{G}$ l'ensemble de ses entités $\mathcal{E}_1 = \{E_1, E_2, ...E_n\}$, l'ensemble des paires de ses entités $\mathcal{E}_2 = \{\{E_1, E_2\}; \{E_1, E_3\}...\{E_{n-1}, E_n\}\}$. Nous pourrions alors décrire l'état de $\mathcal{G}$ avec l'information contenue dans ces deux ensembles $\mathcal{E}_1$ et $\mathcal{E}_2$. L'état d'un autre jeu $\mathcal{G}'$ serait décrit par l'ensemble de ses entités $\mathcal{E}'_1 = \{E'_1, E'_2, ...E'_{n'}\}$ et l'ensemble des paires de ses entités $\mathcal{E}'_2 = \{\{E'_1, E'_2\}; \{E'_1, E'_3\}...\{E'_{n'-1}, E'_{n'}\}\}$, n et n' pouvant être égaux ou différents.

Nous voulons comparer $(\mathcal{E}_1, \mathcal{E}_2)$ et $(\mathcal{E}'_1, \mathcal{E}'_2)$. Si $n = n'$, nous pouvons définir des bijections $\phi : \mathcal{E}_1 \to \mathcal{E}'_1$ et $\psi : \mathcal{E}_2 \to \mathcal{E}'_2$. Nous sommes alors dans le cas bijectif. Mais si $n \neq n'$ nous pouvons définir des fonctions $\phi : \mathcal{E}_1 \to \mathcal{E}'_1$ et $\psi : \mathcal{E}_2 \to \mathcal{E}'_2$ qui sont des équivalences de catégories. Pour ce faire, comme expliqué précédemment, nous définissons des flèches inversibles entre les éléments de $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}'_1$ et $\mathcal{E}'_2$, transformant ces quatre ensembles en catégories. Ces flèches sont définies en utilisant des fonctions d'observation $f_i : \mathcal{E}_i \to \mathcal{O}_i$ $(i = 1, 2)$ et $f'_i : \mathcal{E}'_i \to \mathcal{O}'_i$ $(i = 1, 2)$, qui définissent des relations d'équivalence $\mathcal{R}_{f_1}, \mathcal{R}_{f_2}, \mathcal{R}_{f'_1}$ et $\mathcal{R}_{f'_2}$ sur les ensembles $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}'_1$ et $\mathcal{E}'_2$ respectivement. Nous pouvons alors en déduire des bijections $\Phi : (\mathcal{E}_1/\mathcal{R}_{f_1}) \to (\mathcal{E}'_1/\mathcal{R}_{f'_1})$ et $\Psi : (\mathcal{E}_2/\mathcal{R}_{f_2}) \to (\mathcal{E}'_2/\mathcal{R}_{f'_2})$. Φ (resp. Ψ) peuvent être utilisées pour identifier les entités (resp. paires d'entités) entre des états de jeu différents.

Nous avons expérimenté le cas particulier où $f_1 : \mathcal{E}_1 \to \{0, 1\}$ et $f'_1 : \mathcal{E}'_1 \to \{0, 1\}$ valent 1 si l'entité est un « Moi » et 0 sinon ; et $f_2 : \mathcal{E}_2 \to \{0, 1\}$ et $f'_2 : \mathcal{E}'_2 \to \{0, 1\}$ valent 1 si les trajectoires des deux entités d'une paire s'intersectent, et 0 sinon. Ces fonctions d'observation nous permettent de transférer la connaissance de l'ami dans Breakout — la balle — pour identifier sans apprentissage l'ami dans Pong : la balle.

4 Implémentation et Validation

4.1 Détection et suivi d'entités par analyse d'images

ALE permet de valider cette approche sur des configurations complexes en traitant des signaux directement échantillonnés par l'émulateur. Cette validation de principe repose sur le fait que notre système est capable de détecter puis traquer les entités. Pour vérifier ce prérequis, nous concevons notre algorithme à partir de la librairie open-source OpenCV [21]. Puisque la meilleure description d'une entité correspond à un groupe de pixels identiques ou similaires, la première étape consiste en une détection de contours via un filtre de Sobel sur chacun des trois canaux de l'image en TSV, *i.e.* Teinte, Saturation, Valeur (deuxième image en partant de la gauche en figure 1), ce modèle étant privilégié car plus proche de la perception humaine des couleurs. Ensuite, des boîtes englobantes sont calculées et utilisées en guise d'entités (troisième image de la figure 1).

Pour suivre ces entités, nous les apparions grâce à leur description TSV à des objets identiques rencontrés dans les images passées. De plus, la librairie ORB décrite dans [24] permet aussi de regrouper certains objets similaires, ce qui résout partiellement le problème des objets déformables. Ensuite, dès lors que l'on dispose d'un objet dans l'image présente ainsi que de son « ancêtre » d'une image passée, nous en déduisons son vecteur vitesse, information qui est incluse au résultat de l'algorithme de traitement d'images et apparaît à droite en figure 1.

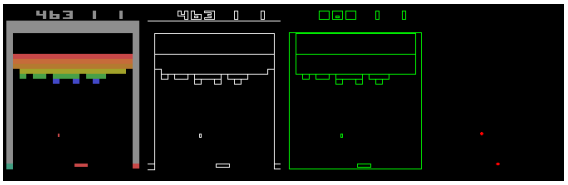

FIGURE 1 – Traitement d'images sur Breakout. De gauche à droite : Image de départ, Contours, Boîtes englobantes, Vitesses mesurées

4.2 Critère de contrôlabilité

Pour trouver l'entité qui est le « Moi » parmi le nombre important d'entités présentes (le nombre d'entités dans le cas des jeux Breakout et Space Invaders est représenté en figures 2 et 3 — l'histogramme serait très similaire pour le jeu Pong), nous utilisons une méthode d'identification de systèmes. La première étape est d'envoyer une séquence d'actions pseudo-aléatoires à ALE pour obtenir les entités affectées par ces signaux, ce qui permet ensuite de construire leur modèle dynamique.

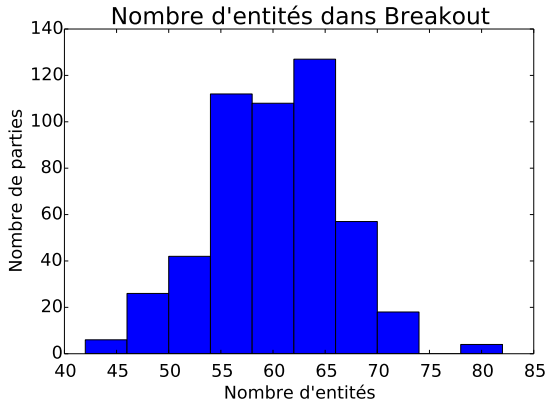

FIGURE 2 – Nombre d'entités détectées dans Breakout

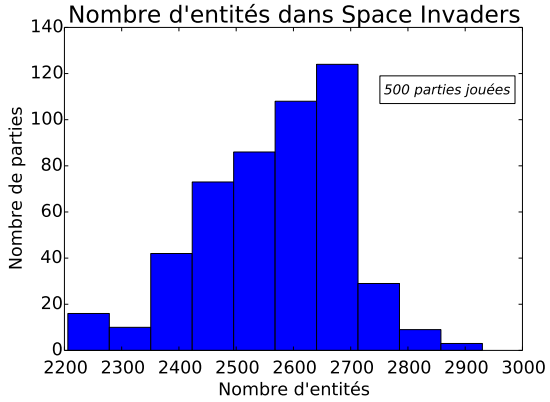

FIGURE 3 – Nombre d'entités détectées dans Space Invaders

Le mouvement de chacune de ces entités est modélisé par les équations d'état discrètes (1) qui correspondent dans notre cas à un système d'ordre $N_d = 2$, où x est le vecteur d'état, y la mesure, u la commande, A la matrice d'état, B la matrice de commande, C la matrice d'observation, D la matrice d'action directe, et k l'indice de l'échantillon.

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned} \tag{1}$$

Pour trouver le « Moi », nous nous restreignons aux entités dont l'erreur entre le modèle dynamique et les mesures est faible (inférieure à cinq fois la hauteur de l'image). Ce faisant, nous utilisons des résultats de *théorie du contrôle*, à savoir le critère de contrôlabilité d'un système dynamique qui stipule qu'un système est complètement contrôlable ssi la matrice de contrôlabilité $C_{N_d} \triangleq [B|AB|\dots|A^{N_d-1}B]$ est de rang N_d [9].
Cependant, les approximations numériques dans les calculs successifs peuvent créer des faux positifs, tels que des matrices C_{N_d} de rang complet alors qu'un des vecteurs est presque nul (par exemple, la norme d'un des deux vecteurs peut être 10^{20} fois plus grande que celle de l'autre, et il devient absurde de parler de colinéarité pour des questions d'échelle). Pour éliminer ces faux positifs, nous passons par le calcul des déterminants de ces matrices de contrôlabilité. En effet, le déterminant étant un accès direct à l'angle entre les deux vecteurs à travers le produit vectoriel, il permet une évaluation plus fine de la colinéarité de nos vecteurs. Les déterminants les plus élevés sont favorisés. Le « Moi » est choisi en fonction de ce critère, et est le système (l'entité) avec le déterminant le plus élevé.

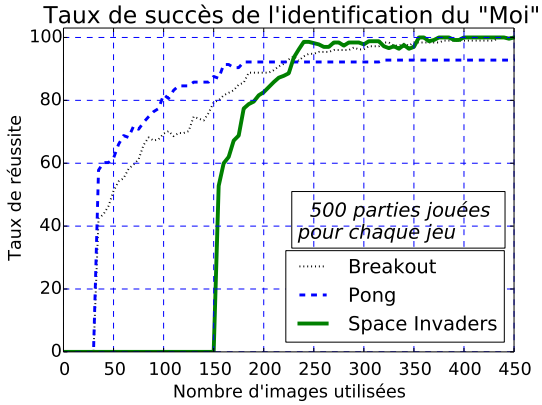

FIGURE 4 – Taux de réussite de l'identification dans trois jeux

Les résultats de cette procédure d'identification sont présentés en figure 4. Pour les trois jeux, notre algorithme cherche à identifier le « Moi » parmi des dizaines (Breakout et Pong) voire des milliers d'entités (Space Invaders) à partir de quelques centaines d'images (soit environ quelques secondes de jeu).
Étant donnée la robustesse de cette méthode et le nombre limité d'images requises, elle peut être appliquée à n'importe quel moment, y compris en cours de partie.

5 Résultats Expérimentaux

5.1 Procédure d'entraînement

L'entraînement de l'agent se fait en 2 étapes. Tout d'abord, il est nécessaire d'identifier le « Moi » parmi toutes les entités, grâce au procédé décrit en partie 4.2. Le système détermine aussi l'existence d'amis ou d'ennemis dans l'environnement en fonction des récompenses qu'ils apportent à l'agent. Dans cet exemple, la seule récompense positive est

l'augmentation du score, et les récompenses négatives sont la diminution du score et la perte de vies.
La stratégie de contrôle du « Moi » implémentée est très basique, et sera améliorée dans des travaux ultérieurs. Le « Moi » est contrôlé de manière à aller vers ses amis, et à fuir ses ennemis.
Les tests sont conduits avec les mêmes paramètres que [20] : l'IA joue des parties de 18 000 images (5 minutes à 60 images/seconde). Nous utilisons le DQN de [20] en guise de référence, car cette publication est l'une des publications sur les jeux Atari les plus citées. Ces travaux ont été reproduits à partir du code mis à disposition par les auteurs. Afin de permettre une comparaison avec notre approche dans des conditions similaires, le score atteint par le DQN suite à un apprentissage fait avec peu d'images d'entraînement est extrait.

5.2 Expériences

Remise en cause des modèles. Comme stipulé en partie 3.2, notre système a été conçu pour être capable de remettre en cause ses propres résultats. C'est pourquoi nous avons inclus une fonctionnalité permettant au système de remettre en cause son identification du « Moi » si celui-ci vient à changer. Par exemple, dans le jeu Breakout, la taille de la raquette diminue la première fois que la balle frappe le mur du haut. Dès lors, l'algorithme détecte que le système auparavant contrôlé est absent, et cherche ce nouveau « Moi » en temps réel par la procédure d'identification présentée en partie 4.2.
Nous évaluons les bénéfices de cette fonctionnalité en comparant le gain de score dû à cette procédure, c'est-à-dire que nous comparons les résultats dans le cas où cette réidentification est possible et le cas où elle ne l'est pas. Chaque expérience est constituée d'une procédure d'entraînement à nombre d'images donné (entre 0 et 12 000) puis d'un test sur une partie de Breakout.

Effet mémoire. Comme décrit en section 3.3, l'algorithme mémorise les connaissances qu'il génère pour les réutiliser lors des parties successives. La procédure transfère tous les « Moi » rencontrés, ainsi que leurs modèles dynamiques, et transfère aussi la liste des amis et ennemis rencontrés au cours de parties précédentes. Chaque expérience inclut 5 parties consécutives et est conduite indépendamment sans, puis avec cette fonctionnalité.

Transfert de connaissances par analogies entre problèmes. L'humain est capable de reconnaître deux situations similaires en établissant des analogies. Ce faisant, lorsque confronté à une situation inconnue mais similaire à une expérience passée, il est capable d'importer une stratégie qu'il a appliqué auparavant (cf. section 3.4). C'est pourquoi notre système est conçu de sorte à être capable de se remémorer les connaissances acquises dans un premier jeu et de les transposer dans une nouvelle situation. Cette capacité est testée sur les jeux Breakout et Pong puisque dans ces deux jeux, la stratégie peut se réduire à déplacer le « Moi » vers la seule entité amie, à savoir la balle. L'objectif est de transférer la stratégie de Breakout et de l'utiliser efficacement dans Pong sans avoir à entraîner complètement le système. Le système est entraîné à partir de 10 000 images de Breakout, puis 500 images de Pong sont utilisées pour identifier le « Moi » avant de tester l'agent sur Pong, en transférant la connaissance de l'ami à partir de Breakout en utilisant les fonctions d'observation décrites dans la Section 3.4.

5.3 Résultats

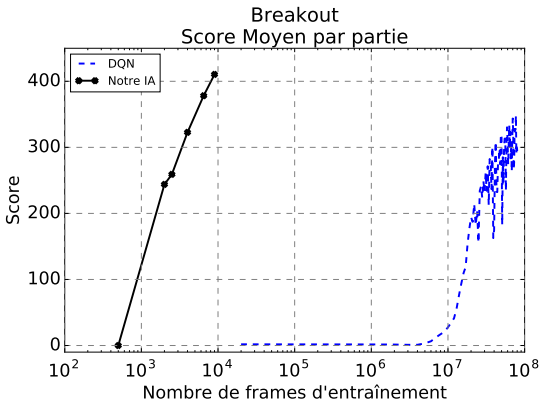

FIGURE 5 – Agent entraîné par DQN vs. notre système. Des scores similaires sont atteints 20 000 fois plus vite

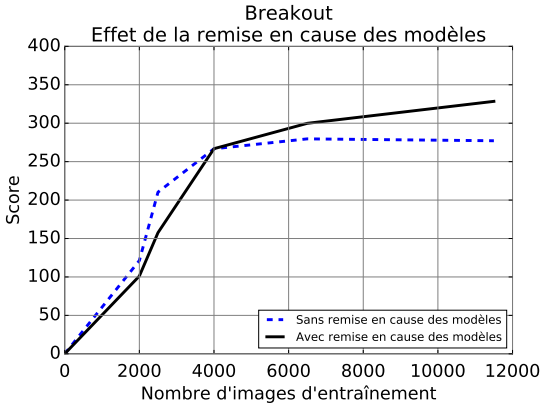

FIGURE 6 – Effet de la remise en cause des modèles. Chaque point (5 par courbe) est une moyenne de 40 expériences

Les résultats [2] sont présentés dans la table 1 et correspondent à 10 000 images d'entraînement (moins de 3 minutes de jeu). La figure 5 reflète le fait que notre système requiert 20 000 fois moins d'images pour atteindre des scores du même niveau que [20].

Remise en cause des modèles. La figure 6 montre les bénéfices apportés sur le jeu Breakout par une remise en

2. Les résultats du joueur humain et des parties aléatoires sont retranscrits depuis [20]. Le joueur s'entraîne préalablement sur chaque jeu pendant 2 heures (432 000 images).

TABLE 1 – Comparaison des scores obtenus après 10 000 images d'entraînement (moyenne ± écart-type sur 8 parties)

JEU	ALÉATOIRE	JOUEUR HUMAIN	DQN	IA
Breakout	1.7	31.8	1.25 ± 1.02	**414.37 ± 88.47**
Pong	−20.7	9.3	−21.00 ± 0.00	**0.20 ± 5.38**

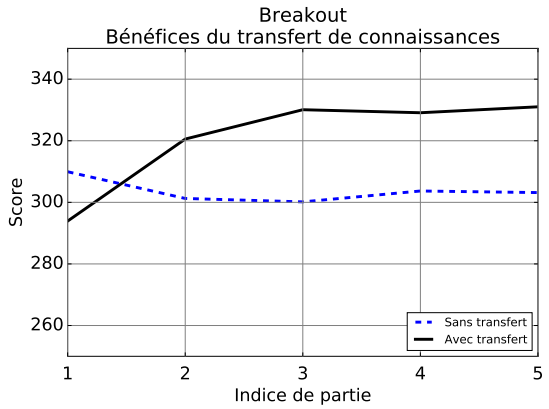

FIGURE 7 – Effet Mémoire. Chaque courbe est une moyenne de 112 expériences

cause des modèles telle que décrite en 5.2. À la fin de la procédure d'entraînement, un meilleur score est atteint si cette remise en cause est possible puisque les modèles sont invalidés dès lors que le « Moi » n'est plus correctement identifié. La procédure de réidentification permet de réactualiser le modèle et de fermer la boucle ouverte. En revanche, sans cette fonctionnalité, l'algorithme joue de façon aléatoire dès que le « Moi » disparaît, ce qui explique la courbe plate au delà de 5000 images d'entraînement. Enfin, pour les entraînements à nombre d'images très réduit, les résultats de l'identification sont trop aléatoires pour avoir une comparaison significative des algorithmes et les scores obtenus sont très similaires (de l'ordre d'une dizaine de points d'écarts).

Effet mémoire. La figure 7 reflète l'évolution du score lorsque la connaissance que l'on a du « Moi » est transférée d'une partie à l'autre. Ce transfert de connaissances fait augmenter le score de façon évidente puisque, dès lors qu'une nouvelle forme du « Moi » a été découverte, le système conserve celle-ci en mémoire et reconnaît cette forme instantanément dès qu'elle réapparaît et l'utilise comme « Moi ». Dans l'autre cas, le score est à peu près constant, car des points peuvent être perdus pendant que la réidentification a lieu au cours de la partie.

Transfert de connaissances par analogies entre problèmes. Cette expérience est réalisée selon les paramètres décrits en partie 5.2. Quand l'IA importe la stratégie du jeu Breakout, les tests sur Pong conduisent aux résultats suivants : -2.53 ± 3.79 ce qui demeure très proche des résultats obtenus lorsque notre système était préalablement entraîné sur Pong (0.20 ± 5.38). Cela prouve que notre cadre devient d'autant plus utile quand il s'agit de transposer des stratégies entre des problèmes analogues, bien que l'on ne dispose d'aucune connaissance du nouveau problème. En effet, cette expérience montre que l'IA est capable de jouer à ces deux jeux à partir de 10 000 images de Breakout et 500 images de Pong (10 500 images au total) aussi bien que si elle s'était entraînée séparément sur 10 000 images de chaque jeu (20 000 images au total).

6 Conclusion

Ces travaux introduisent un cadre basé sur des concepts élémentaires de la théorie des catégories qui permet de réduire de manière significative la quantité de données nécessaires pour entraîner et contrôler un agent sur Atari 2600. Pour ce faire, il a semblé pertinent d'échantillonner nos données à une échelle mésoscopique entre le pixel et l'image entière. Par ailleurs, il a fallu construire un modèle d'entités qui sont détectées puis suivies grâce à des outils usuels de traitement du signal. Nos expériences montrent qu'il est possible d'atteindre des résultats similaires à l'état de l'art avec seulement quelques minutes de jeu en guise d'entraînement. De plus, ce cadre peut aussi servir de base à des méthodes d'apprentissage par transfert. En effet, nos autres expériences ont mis en exergue la possibilité de transférer une stratégie d'un jeu à un jeu analogue, puis de commencer à jouer avec seulement quelques secondes de transition.

Les prochains travaux viseront à atteindre une meilleure compréhension de l'environnement par l'agent à partir des prédictions obtenues des modèles, ainsi qu'à développer un panel de stratégies applicables, puisqu'à ce jour seule une stratégie de survie basique a été utilisée.

Ces résultats théoriques, validés sur un environnement de simulation de jeu, pourront être appliqués dans l'industrie à des problèmes pour lesquels l'acquisition de données, et le temps ou la puissance de calcul requis sont des facteurs limitants, empêchant l'application des méthodes classiques en Intelligence Artificielle tels que les réseaux de neurones.

References

[1] A. Barreto, W. Dabney, R. Munos, J. Hunt, T. Schaul, D. Silver, and H. van Hasselt, "Successor features for transfer in reinforcement learning," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 4058–4068.

[2] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, June 2013.

[3] M. Bogdanovic, D. Markovikj, M. Denil, and N. de Freitas, "Deep apprenticeship learning for playing video games," in *AAAI Workshop on Learning for General Competency in Video Games*, 2015.

[4] M. Campbell, A. J. Hoane, and F. Hsu, "Deep blue," *Artificial Intelligence*, vol. 134, no. 1, pp. 57 – 83, 2002.

[5] C. Chang and H. J. Keisler, *Model Theory*, 3rd ed. North Holland, 1990.

[6] F. de Saussure, *Cours de linguistique générale*. Payot, 1916.

[7] L. Fei-fei, "Knowledge transfer in learning to recognize visual object classes," in *In: International Conference on Development and Learning (ICDL*, 2006.

[8] H. v. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI'16. AAAI Press, 2016, pp. 2094–2100.

[9] R. E. Kalman, "On the general theory of control systems," in *Proceedings of the First International Congress on Automatic Control*. Butterworth, London, 1960, pp. 481–492.

[10] K. Kansky, T. Silver, D. A. Mély, M. Eldawy, M. Lázaro-Gredilla, X. Lou, N. Dorfman, S. Sidor, D. S. Phoenix, and D. George, "Schema networks: Zero-shot transfer with a generative causal model of intuitive physics," in *ICML 2017*, 8 2017, pp. 1809–1818.

[11] G. Lee, M. Luo, F. Zambetta, and X. Li, "Learning a Super Mario controller from examples of human play," in *2014 IEEE Congress on Evolutionary Computation (CEC)*, July 2014, pp. 1–8.

[12] N. Levine, T. Zahavy, D. Mankowitz, A. Tamar, and S. Mannor, "Shallow updates for deep reinforcement learning," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 3138–3148.

[13] Y. Liang, M. C. Machado, E. Talvitie, and M. Bowling, "State of the art control of Atari games using shallow reinforcement learning," in *AAMAS '16*, 2016, pp. 485–493.

[14] Y. Lin, Z. Hong, Y. Liao, M. Shih, M. Liu, and M. Sun, "Tactics of adversarial attack on deep reinforcement learning agents," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017, pp. 3756–3762.

[15] N. Lipovetzky, M. Ramirez, and H. Geffner, "Classical planning with simulators: Results on the Atari video games," in *IJCAI'15*, 2015, pp. 1610–1616.

[16] Z. Luo, Y. Zou, J. Hoffman, and L. Fei-Fei, "Label efficient learning of transferable representations acrosss domains and tasks," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 164–176.

[17] S. Mac Lane, *Categories for the working mathematician*, 2nd ed. Springer-Verlag, 1998.

[18] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proceedings of The 33rd ICML*, ser. Proceedings of Machine Learning Research, vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1928–1937.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," in *NIPS Deep Learning Workshop*, 2013.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, February 2015.

[21] OpenCV, "Open source computer vision library," https://opencv.org/, 2017.

[22] T. Pepels, M. H. M. Winands, and M. Lanctot, "Real-Time Monte-Carlo Tree Search in Ms. Pac-Man," *IEEE Trans. on Computational Intelligence and AI in Games*, vol. 6, no. 3, pp. 245–257, Sept 2014.

[23] K. Popper, *The Logic of Scientific Discovery*. Hutchinson & Co, London, 1959.

[24] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*, Nov 2011, pp. 2564–2571.

[25] S. Sabour, N. Frosst, and G. Hinton, "Dynamic routing between capsules," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 3859–3869.

[26] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, pp. 354–359, October 2017.

FIGURE 8 – Algorithme utilisé.

Require: $bdc = [entites, modeles, strategies]$
Require: $strategie\_par\_defaut = actions\_pseudo\_aleatoires$
Require: $seuil$, N_{max}
1: **for** $N_{init} = 0$ to N_{max} **do**
2: $entites$.append(boites_englobantes)
3: $action \leftarrow strategie\_par\_defaut$
4: applique($action$)
5: **end for**
6: $jeu\_similaire \leftarrow$ recherche jeu avec des $entites$ similaires
7: $modeles, strategies \leftarrow$ extrait($bdc, jeu\_similaire$)
8: bdc.append($[entites, modeles, strategies]$)
9: $f\_refute\_modeles \leftarrow$ **false**
10: **if** $modeles$ est vide **then**
11: $f\_refute\_modeles \leftarrow$ **true**
12: **end if**
13: **while** experimentation en cours **do**
14: $entites$.append(boites_englobantes)
15: **if** $f\_refute\_modeles$ **then**
16: $action \leftarrow strategie\_par\_defaut$
17: $modeles, strategies, f\_id \leftarrow$ identification($entites$)
18: **if** f_id **then**
19: $f\_refute\_modeles \leftarrow$ **false**
20: bdc.append($[entites, modeles, strategies]$)
21: **end if**
22: **else**
23: $action \leftarrow$ cherche_action($entites, strategies$)
24: $predictions \leftarrow$ cherche_predictions($entites, modeles$)
25: $erreur \leftarrow predictions - entites$
26: **if** $|erreur| > seuil$ **then**
27: $f\_refute\_modeles \leftarrow$ **true**
28: **end if**
29: **end if**
30: applique($action$)
31: **end while**
32: **return** bdc

Annexe 1 : Algorithme

L'algorithme utilisé sur les jeux Atari 2600 est décrit en pseudo-code sur la figure 8. Il repose sur une base de connaissance bdc, une stratégie par défaut $strategie\_par\_defaut$, un seuil $seuil$ de rejet du modèle et un nombre d'itérations N_{max} correspondant à la durée de l'initialisation.

La base bdc, initialement vide, se remplit progressivement des stratégies et modèles obtenus au cours des parties jouées. La stratégie par défaut $strategie\_par\_defaut$ est une séquence pseudo-aléatoire d'actions envoyées en entrée du système.

L'algorithme commence par acquérir l'image brute provenant de l'émulateur ALE puis en déduit (en utilisant le procédé décrit en Section 4.1) des boites englobantes correspondant à des types d'entités en ligne 2. À partir de ces entités, l'algorithme détermine la situation qui présente les objets les plus similaires en ligne 6 avant d'extraire en ligne 7 les modèles et stratégies obtenus lors de cette situation. La base de connaissances bdc est ensuite complétée.

Lors de la partie, le système continue à acquérir des échantillons et à les transformer en entités. Si le modèle est valide (lignes 23 à 27), le systéme calcule à partir des entités présentes et stratégies disponibles la meilleure action à effectuer. Avec les entités et les modèles, une prédiction de l'état de l'environnement est aussi calculée. Si ces predictions sont trop éloignées de l'état mesuré, alors la variable f_refute_savoir devient vraie. Dans ce cas, dès l'itération suivante, la procédure d'identification (lignes 15 à 20) se lance, et de nouveaux modèles et stratégies sont calculés puis appliqués dès l'itération suivante.

Notons que l'algorithme est identique en phase d'apprentissage et en phase d'exploitation (test). Il continue de réfuter son savoir, si besoin est, même en phase d'exploitation.

Annexe 2 : Illustration du transfert de stratégie grâce à la théorie des catégories

Considérons un jeu $\mathcal{C}$ (par exemple le jeu Breakout pour Atari 2600) où une raquette horizontale est située en x_0 sur un segment $[0, N]$ d'états initiaux possibles ($0 \leq x_0 \leq N$) et peut se déplacer vers la gauche ou la droite afin d'atteindre une balle devant arriver à la position $x_1 \in [0, N]$. La stratégie π est la suivante : si $x_1 < x_0$, se déplacer vers la gauche, si $x_1 = x_0$, ne pas bouger, mais si $x_1 > x_0$, alors se déplacer vers la droite. Désormais, considérons un autre jeu $\mathcal{C}'$ (tel que Pong sur Atari 2600 par exemple) où cette fois-ci une raquette verticale se trouve en position y_0 sur le segment $[0, N']$ des positions atteignables ($0 \leq y_0 \leq N'$) et peut se déplacer de haut en bas pour toucher une balle qui atteindra la position $y_1 \in [0, N']$.

Nous souhaitons *identifier* $\mathcal{C}$ et $\mathcal{C}'$ afin de transposer la stratégie π de $\mathcal{C}$ à $\mathcal{C}'$. Les états (objets) de $\mathcal{C}$ sont les $N + 1$ positions x_1 de $[0, N]$. De la même façon, les états (objets) de $\mathcal{C}'$ sont les $N' + 1$ positions y_1 de $[0, N']$. Si $N = N'$ et $x_0 = y_0$, alors il est facile de définir une bijection de $\mathcal{C}$ à $\mathcal{C}'$ puis de transposer π, mais si $N \neq N'$ ou $x_0 \neq y_0$ alors il est impossible de définir une telle bijection. Pour résoudre ce problème, nous transformons $\mathcal{C}$ et $\mathcal{C}'$ en catégories à travers la définition suivante de flèches, et ce faisant, nous pourrons définir une équivalence de catégories $F : \mathcal{C} \to \mathcal{C}'$. Pour définir ces flèches, nous utilisons les fonctions d'observation suivantes définies sur les états de $\mathcal{C}$ et $\mathcal{C}' : f : \mathcal{C} \to \{0, 1, 2\}$ et $f' : \mathcal{C}' \to \{0, 1, 2\}$ telles que $f(x_1) = 0$ ssi $x_1 < x_0$, $f(x_1) = 1$ ssi $x_1 = x_0$ et $f(x_1) = 2$ ssi $x_1 > x_0$. De la même façon, $f'(y_1) = 0$ ssi $y_1 < y_0$, $f'(y_1) = 1$ ssi $y_1 = y_0$ et $f'(y_1) = 2$ ssi $y_1 > y_0$. Ensuite, nous définissons une flèche inversible entre deux états de $\mathcal{C}$ ssi ils ont la même image par f, ainsi qu'une flèche inversible entre deux états de $\mathcal{C}'$ ssi ils ont la même image par f'. Soit $F : \mathcal{C} \to \mathcal{C}'$, telle que si $x_1 < x_0$, alors $F(x_1) = 0$; si $x_0 = x_1$, alors $F(x_1) = y_0$; et si $x_1 > x_0$, alors $F(x_1) = N'$. Cette fonction F définit une équivalence de catégories et induit une bijection F' entre les ensembles de classes : $F' : (\mathcal{C}/\simeq) \to (\mathcal{C}'/\simeq)$. $(\mathcal{C}/\simeq)$ correspond (grâce à la stratégie π) aux actions disponibles $\{Gauche, Attendre, Droite\}$ dans $\mathcal{C}$ alors que $(\mathcal{C}'/\simeq)$ correspond aux actions disponibles $\{Haut, Attendre, Bas\}$ dans $\mathcal{C}'$ (en choisissant d'orienter l'axe des y du haut vers le bas). Dans ce cas-ci $F'(Gauche) = Haut$, $F'(Attendre) = Attendre$ et $F'(Droite) = Bas$.

En reproduisant ces raisonnements, la théorie des catégories permet de transférer rigoureusement des stratégies d'un problème à un problème analogue.