# Théories philosophiques de l'IA

PHI 3330

Séance 1

# Plan du cours

- Tout est sur le site web:
- https://jonsimon.net/theories_IA/

- Vos notes seront basée sur :
- Une dissertation de 1000-1500 mots 38%  (3 mars)
  - Complétez la « liste de contrôle du résumé »  2% (24 fév)
- Une dissertation finale de 1000 – 1500 mots 48%  (24 avril)
  - Complétez la « liste de contrôle du résumé »  2% (17 avril)

# Apercu

- 0) Marr : niveaux d'explication dans les sciences cognitives
- 1) Turing:  Qu’est-ce qu’un ordinateur?
- 2) Chalmers: Qu'est-ce qu’un algorithme / qu’est-ce que de mettre en œuvre un algorithme?

# David Marr

Vision 1982

# Marr

- 3 Niveaux :

- Le niveau computationnel
- Le niveau algorithmique
- Le niveau d'implémentation

# Marr

- Le niveau computationnel

- Le niveau général, en précisant le problème à résoudre : p. ex.: «faire un gâteau»

# Marr

- Le niveau algorithmique

- Le niveau spécifiant les éléments cruciaux du plan pour résoudre le problème, par exemple : «La recette de ma grand-mère pour faire un gâteau». (Mélange une poignée de farine, une cuillère de sucre...).

# Marr

- Le niveau d'implémentation

- Le niveau spécifiant tous les détails de la façon dont on suit exactement la recette. Quelle quantité de farine exactement, et quelle sorte (raffinée ou non) ? Quelle quantité de sucre exactement, et quelle sorte (brun, blanc, canne) ? Quel type de bol à mélanger, en bois ou en métal ? Etc...

# Marr

Note que nous pouvons faire des subdivisions et qu'il s'agit en fait plus d’une gamme que d'une stricte distinction en trois parties

(Comparez la distinction: stratégique vs tactique vs opérationnel).

# Niveau d’explication

1) du problème qu'elle résout (peu importe comment), (**behaviorisme**)

2) en vertu de la stratégie générale qu'elle utilise pour résoudre un problème, (**fonctionnalisme**)

3) en raison de détails très spécifiques de la façon dont elle met en œuvre cette stratégie (**physicalisme**)

# Niveau d’explication

Le niveau informatique : classe cet ensemble en nombres pairs et impairs.

Niveau algorithmique : bubble sort vs merge sort

Le niveau de l'implémentation : Mac ou PC ou personne avec un crayon et du papier...

# Niveau d’explication

il est utile de réfléchir aux efforts déployés pour donner une définition rigoureuse de la notion informelle de *méthode efficace* - la notion intuitive d'une méthode permettant de résoudre un problème en un nombre fini d'étapes à l'aide d'un ensemble fixe de règles ne nécessitant pas d'intuition.

# Niveau d’explication

- Cet effort a culminé en 1936 avec les articles d'Alonzo Church et d'Alan Turing, qui ont chacun proposé des caractérisations formelles de ce qu'est un problème qui peut être résolu par une telle méthode.
- Turing, dans une annexe à son article de 1936, a prouvé l'équivalence extensionnelle de ses deux caractérisations formelles et de celles de Church.
- D'autres approches, comme celle de Von Neumann, ont également prouvé qu'elles étaient équivalentes à ces deux caractérisations, c'est-à-dire qu'elles permettaient d'obtenir la même classe de fonctions.

# Niveau d’explication

- La thèse Church-Turing est une autre affirmation (philosophique) selon laquelle la classe de fonctions identifiée par les définitions de Church, Turing (et Von Neumann) capture l'extension de la conception intuitive de la méthode efficace [Copeland, SEP].

# Niveau d’explication

- Note, de manière cruciale, que l'équivalence discutée dans ces travaux fondateurs est extensionnelle plutôt qu'intensionnelle.
- L'annexe de Turing (1936) montre comment construire une méthode efficace pour qu'une machine de Turing calcule une fonction, étant donné une méthode efficace pour la résoudre dans le lambda calcul (et vice versa),
- …et non pas que les deux méthodes efficaces sont identiques.

# Niveau d’explication

- De plus, en général, il existe plusieurs méthodes efficaces pour calculer une fonction lorsqu'il y en a une.
- Par exemple, on peut utiliser le tri par bulles ou le tri par fusion pour dériver un tableau trié d'un tableau non trié : ce sont des méthodes efficaces différentes (et il peut y avoir des raisons claires d'utiliser l'une plutôt que l'autre), mais elles résolvent la même fonction calculable.
- Cela signifie que nous devons faire la distinction entre la question des fonctions qu'un système peut calculer et la question des méthodes efficaces (c'est-à-dire des algorithmes) qu'il utilise pour le faire.

# Is Bubble Sort efficient?

Bubble sort can be fast if most items are in order. Otherwise it can be slow because it would involve a **lot of swaps**.

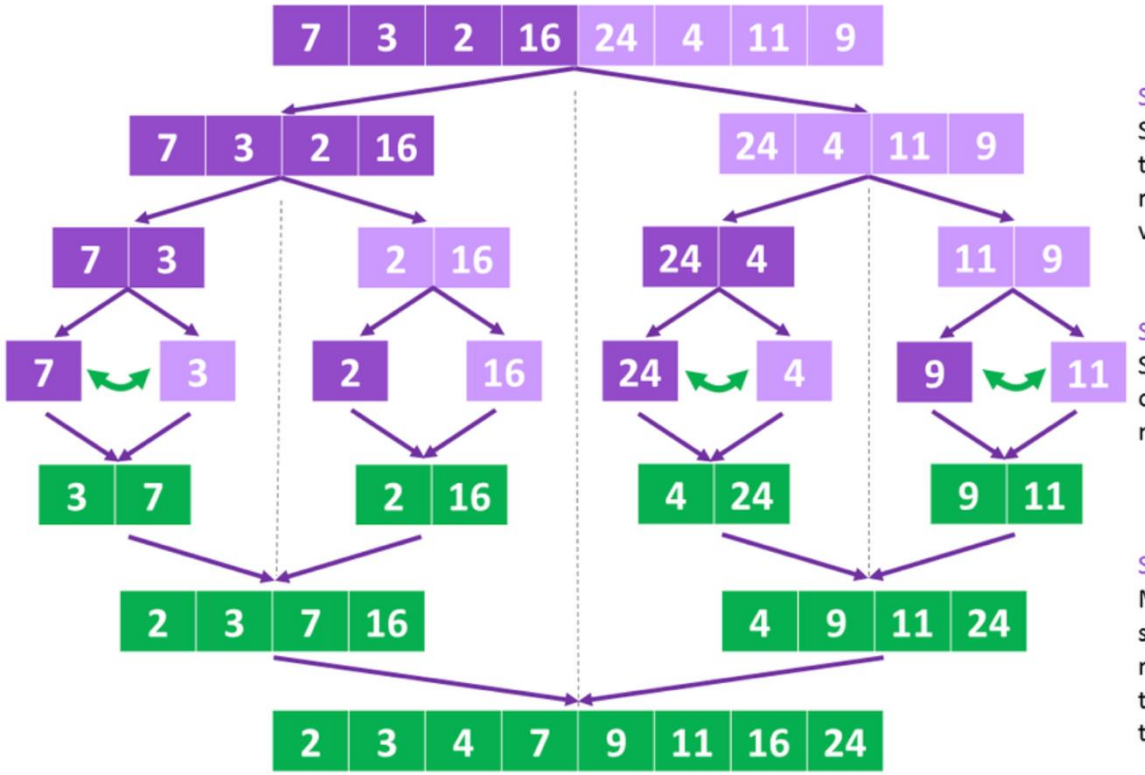
# The Swap Algorithm

```
int temp = nums[p];
nums[p] = nums[p + 1];
nums[p + 1] = temp;
```

A swap involves the following three lines of code.

- **Temporarily store** the first list item ( as the next step wil
  overwrite it).
- Set the first item to the next item.
- Set the next item to the temporary stored item.

# Merge Sort




Step 1:
Split sub-lists in two until you reach pair of values.

Step 3:
Sort/swap pair of values if needed.

Step 4:
Merge and sort sub-lists and repeat process till you merge to the full list.

# Alan Turing

# Alan Turing

# Turing

- Turing est surtout connu pour ses travaux sur la théorie de la calculabilité : il a identifié le problème de l'arrêt (die Entscheidungsproblem), a fourni des définitions concrètes de nombreux termes centraux de la théorie du calcul (comme celui d'un ordinateur universel, alias une machine de Turing). Il a également travaillé à la décodage de la machine Enigma (le système de cryptographie nazi) pendant la Deuxième Guerre mondiale.
- En parallèle, il a rédigé ce document, fixant l'agenda de la recherche sur l'IA pour les 50 prochaines années

# Machines Informatique et Intelligence

- 1) Les machines peuvent-elles penser ?
- propose Turing : il est trop difficile de répondre à cette question. Remplacez-la par :
- 2) Une machine peut-elle être performante (tromper quelqu'un 30 % du temps) au jeu d'imitation ?

# Machines Informatique et Intelligence

- Le jeu de l'imitation :

- Un examinateur, deux joueurs. Un joueur est humain, l'autre est une machine. Les joueurs sont dans des salles séparées, seuls des messages écrits sont échangés. L'examinateur peut demander n'importe quoi à l'un ou l'autre des participants, le but étant de deviner qui est l'humain. L'objet du jeu est de tromper l'examinateur en lui faisant croire que l'autre joueur est la machine.

# Machines Informatique et Intelligence

- Pourquoi propose-t-il ce remplacement ?
- 1) Il évite un débat philosophique insolubles, le remplace par une question qui reste assez substantielle et difficile à répondre
- 2) Comportementalisme / Positivisme ?
- 3) Théorie cartésienne de la primauté de la preuve de la capacité linguistique (rappel de Descartes vs. Romanes et Huxley)

# Machines Informatique et Intelligence

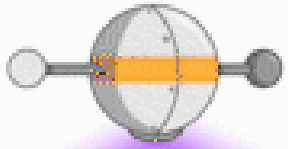
- Le reste du document :
- 1) la formulation de ce que doit être un ordinateur numérique (une machine de Turing) et de ce que doit être l'exécution d'un programme informatique (une machine à états discrets).

- 2) Ensuite, les réponses aux objections prévues pour affirmer qu'un ordinateur numérique sera capable de passer le test 50 ans plus tard (en 2001)

- 3) Enfin une esquisse positive d'une voie à suivre : ici, quelques réflexions classiques sur l'IA. Notez cependant qu'il ne les utilise pas pour répondre à des objections...

# Les machines de Turing

- Le vrai génie de tout cela est qu'il est possible de démontrer que pratiquement toute opération formelle (qui intuitivement est quelque chose qui peut être fait sans nécessiter de perspicacité ou d'imagination, quelque chose que «n'importe quel ordinateur pourrait faire») peut être encodée arithmétiquement de la manière appropriée

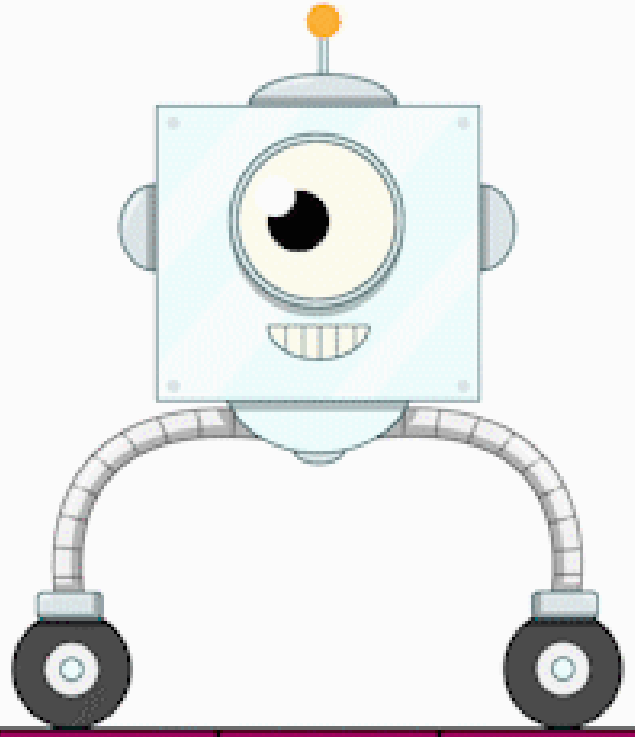


**State 2**

if 0
then erase

write 1
move right

go to state 5



| | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|

# Le fonctionnalisme machinique

- Une machine de Turing : une longueur de bande / papier (aussi longue que nécessaire) divisée en cellules.
- Chaque cellule porte un numéro écrit dessus.
- Un lecteur/enregistreur fonctionne sur une cellule à la fois.
- Il lit le numéro, puis le modifie selon son *état*, et selon un ensemble de règles, puis se déplace vers une autre cellule (et modifie son état) selon un ensemble de règles.

# Le fonctionnalisme machinique

- Il est essentiel que les règles indiquent au scanner ce qu'il doit écrire étant donné ce qu'il lit *et dans quel état il se trouve*. Les règles disent aussi *dans quel état il doit aller*.

# Le fonctionnalisme machinique

La liste des règles est appelée *une table de machine*. Voici un exemple de table de machine pour une machine à coke (prochaine page). Notez que la «signification» des états est définie implicitement (les étiquettes ne sont ici qu'à des fins heuristiques)

# Example: Machine à Coke (Coke coute 10c)

TABLE 1

| | $S_1$ | $S_2$ |
|---|---|---|
| Input : pièce de 5 cents | N'émettre aucun output Passer en $S_2$ | Emettre un coca-cola Passer en $S_1$ |
| Input : pièce de 10 cents | Emettre un coca-cola Rester en $S_1$ | Emettre un coca-cola et une pièce de 5 cents Passer en $S_1$ |

# Example: Machine à Coke (coke coute 15c)

| Input | État | Prochain État | Output |
|---|---|---|---|
| 5c | Désire 15c | Désire 10c | rien |
| 5c | Désire 10c | Désire 5c | rien |
| 5c | Désire 5c | Désire 15c | Coke! |
| 10c | Désire 15c | Désire 5c | rien |
| 10c | Désire 10c | Désire 15c | Coke! |
| 10d | Désire 5c | Désire 10c | Coke! |

# Example: Machine à Coke (coke coute 15c)

| Input | État | Prochain État | Output |
|---|---|---|---|
| 5c | $S_0$ | $S_1$ | rien |
| 5c | $S_1$ | $S_2$ | rien |
| 5c | $S_2$ | $S_0$ | Coke! |
| 10c | $S_0$ | $S_2$ | rien |
| 10c | $S_1$ | $S_0$ | Coke! |
| 10d | $S_2$ | $S_1$ | Coke! |

# Le fonctionnalisme machinique

Observations :

1) Seules les entrées et les sorties sont définies explicitement, les états internes sont définis implicitement

2) Holisme : les états ne sont définis qu'implicitement en référence les uns aux autres. Si vous modifiez l'un des états, vous modifiez chacun d'eux

# Turing: Machines Informatique et Intelligence

# Réussir le test

- Quel est exactement son argument selon lequel une telle machine pourrait passer le test ? Il n'a pas vraiment d'argument décisif : il examine plutôt les objections et y répond.

- À noter ici : en particulier, il ne fonde pas explicitement tout sur l'idée que toute pensée humaine est une manipulation formelle de symboles : son objectif principal est de faire valoir qu'un ordinateur numérique pourrait réussir le test, et non qu'une "IA classique" pourrait. Les hypothèses qui conduisent à l'IA classique n'arrivent qu'à la fin

# Objections

- Objection théologique
- La tête dans le sable
- Objection mathématique
- Argument de la conscience
- Divers handicaps
- L'objection de Lady Lovelace
- La continuité du système nerveux
- L'informalité du comportement
- Perception extra-sensorielle

# Objections

- Objection théologique: Nous ne pouvons pas créer des âmes ! Turing : en effet. Mais ici, la question est seulement de savoir si nous pouvons créer des manoirs pour eux.
- La tête dans le sable: C'est effrayant ! Turing : pas d'objection
- Objection mathématique: Limitations formelles sur ce que les ordinateurs peuvent penser / prouver ! Turing : pourquoi de telles limitations ne peuvent-elles pas exister aussi pour les connaissances humaines ?

# Objections

- Argument de la conscience: Une machine ne peut jamais vraiment être consciente ? Turing : vous pourriez changer d'avis si vous en voyiez une passer le test. Nous pouvons également aborder la question de savoir si la machine peut passer le test sans adresser celui-ci
- Divers handicaps: Mais une machine ne sera jamais capable de... Turing : soit non pertinent (manger de la glace), soit suppliant (réussir ce test), soit faux (faire des erreurs)
- L'objection de Lady Lovelace: Les machines ne peuvent pas vraiment créer / engendrer quoi que ce soit. Turing : dans un sens pertinent, il n'est pas évident que nous non plus.

# Objections

- La continuité du système nerveux: Le système nerveux est continu et non discret. Turing : l'examinateur en jeu ne pourrait pas en tirer profit.

- (q : oui, mais cela pourrait-il avoir une importance pour que la machine soit suffisamment performante ?)

# Objections

- L'informalité du comportement:  Le comportement humain ne peut pas être décrit par un système de règles !

- Turing : Peut-être qu'elles peuvent être son juste difficile. Notez combien il serait difficile de déduire les règles qui régissent un ordinateur réel simplement en observant son comportement...
- Perception extra-sensorielle: ?

# Apprentissage machine

- Turing conclut son article par quelques réflexions sur la façon d'amener les machines à apprendre à bien jouer (plutôt que de les programmer explicitement).

- Ses idées ici anticipent certaines idées dans l'apprentissage du renforcement, mais vont aussi clairement dans le sens de l'IA classique.  Mais il ne s'appuie pas sur ces idées dans la première partie de l'article pour défendre ses hypothèses clés...

# Compréhension / consolidation

- Est-il logique d'utiliser un modèle comme celui-ci pour tester la conscience plutôt que l'intelligence ?
- Ce test soutient-il vraiment le computationnisme, même s'il est en fin de compte comportemental ?
- Comment faire la différence entre un ordinateur qui fonctionne mal et quelque chose qui n'est pas du tout un ordinateur ?
- Le fait que le chatgpt puisse passer ce test : comment cela affecte-t-il votre opinion à son sujet ?

# Qu'est-ce que c'est que de mettre en œuvre un calcul?

# Chalmers, Maudlin, Klein, Bostrom:

# Bonnes et mauvaises abstractions

# Bonnes et mauvaises abstractions

- Si les machines de Turing (en tant que descriptions abstraites ou mathématiques) spécifient des algorithmes, comment dire qu'un système donné «est» (ou, implémente / met en ouevre) une machine de Turing ?

# Bonnes et mauvaises abstractions

- **Premier problème** : les descriptions des machines de Turing sont trop abstraites - même un rocher pourrait être «*interprété*» comme une machine de Turing
- **Deuxième problème** : les descriptions des machines de Turing sont trop exigeantes : même les ordinateurs ont des problems, font des erreurs, etc. Cela signifie-t-il que ce ne sont pas vraiment des ordinateurs ?

# Bonnes et mauvaises abstractions

- Problème: les descriptions des machines de Turing sont trop abstraites - même un rocher pourrait être «interprété» comme une machine de Turing

# Example: Machine à Coke (coke coute 15c)

| Input | État | Prochain État | Output |
|---|---|---|---|
| 5c | Désire 15c | Désire 10c | rien |
| 5c | Désire 10c | Désire 5c | rien |
| 5c | Désire 5c | Désire 15c | Coke! |
| 10c | Désire 15c | Désire 5c | rien |
| 10c | Désire 10c | Désire 15c | Coke! |
| 10d | Désire 5c | Désire 10c | Coke! |

# Example: Machine à Coke (coke coute 15c)

| Input | État | Prochain État | Output |
|---|---|---|---|
| 5c | $S_0$ | $S_1$ | rien |
| 5c | $S_1$ | $S_2$ | rien |
| 5c | $S_2$ | $S_0$ | Coke! |
| 10c | $S_0$ | $S_2$ | rien |
| 10c | $S_1$ | $S_0$ | Coke! |
| 10d | $S_2$ | $S_1$ | Coke! |

# Example: Machine à Coke (coke coute 15c)

| Input | État | Prochain État | Output |
|---|---|---|---|
| $I_1$ | $S_0$ | $S_1$ | $O_1$ |
| $I_1$ | $S_1$ | $S_2$ | $O_1$ |
| $I_1$ | $S_2$ | $S_0$ | $O_2$ |
| $I_2$ | $S_0$ | $S_2$ | $O_1$ |
| $I_2$ | $S_1$ | $S_0$ | $O_2$ |
| $I_2$ | $S_2$ | $S_1$ | $O_2$ |

# Bonnes et mauvaises abstractions

- Problème: les descriptions des machines de Turing sont trop abstraites - même un rocher pourrait être «interprété» comme une machine de Turing

# Bonnes et mauvaises abstractions

- Problème:

Étiquette l'environnement initial de la roche comme I1, sa configuration initiale S1, sa configuration un moment plus tard S2, l'environnement comme O1 : alors il correspond à une ligne sur ce tableau !

# Bonnes et mauvaises abstractions

- Résolution:
- Nous avons besoin d'une sorte de contrainte contrefactuelle, ou d’isomorphism entre table de machine et mécanisme, c'est-à-dire que si le système avait été dans un état initial différent ou avait reçu une entrée différente, il aurait quand même satisfait à la description (de table de machine) en question...

# Bonnes et mauvaises abstractions

- problèmes résiduels :
- 1) Est-ce trop fort ?  Nous devons permettre que, par exemple, les ordinateurs puissent avoir des problèmes (tout en restant des ordinateurs). Quel degré de défaillance pouvons-nous autoriser ?  À quel moment un système passe-t-il du statut d'ordinateur très glitchy ou peu fiable, à celui de ne plus être un ordinateur (Bostrom).

# Bonnes et mauvaises abstractions

- problèmes résiduels :
- 2) Cause par omission : le fait de se fier à un critère contrefactuel peut impliquer que cela fait une différence, pour quelque chose qui ne se produit pas, si cela aurait pu se produire (l'alto silencieux ajoute à l'esthétique de la pièce, Klein).

# Bonnes et mauvaises abstractions

- problèmes résiduels :
- 3) Ce n'est pas assez fort ? Nous pouvons décrire des systèmes qui respectent la condition contrefactuelle, mais qui intuitivement ne semblent toujours pas mettre en ouvre l'algo (Maudlin, Klein).

# Bonnes et mauvaises abstractions

- problèmes résiduels :
- 3) Ce n'est pas assez fort ?
- Simon vs Theodore vs Alvin:

# Bonnes et mauvaises abstractions

- Simon est une machine de Turing qui, étant donné un nombre, produit ses facteurs. Pour le nombre 20, elle produit (1,2,4,5). Étant donné le nombre 21, elle produit (1,3,7)...

# Bonnes et mauvaises abstractions

- Théodore ne peut résoudre qu'un seul problème (par exemple, trouver les facteurs de 20 : 1,2,4,5). Quel que soit le nombre que tu lui donnes, il exécute la recette pour factoriser 20 et obtient (1,2,4,5).

# Bonnes et mauvaises abstractions

- Alvin est composé de Théodore et de Simon : par défaut, il exécute le programme le plus simple de Théodore. Mais il possède un interrupteur qui vérifie si l'entrée est 20 ou non. Si l'entrée n'est pas 20, l'interrupteur désactive le mécanisme de Théodore et active celui de Simon.

- Alvin est contrefactuellement sensible.... mais si on lui donne l'entrée 20, il fait la même chose que Simon.

# Bonnes et mauvaises abstractions

- Argument de Maudlin:
- 1) Si Alvin est conscient et que Théodore ne l'est pas, deux systèmes peuvent différer en matière de conscience, uniquement en raison de différences dans leurs tendances inertes (non manifestées)..
- 2) Mais la conscience est déterminée par ce qui est manifeste

# Bonnes et mauvaises abstractions

- Argument de Maudlin:
- 3) Donc, Alvin n’est pas conscient. (conclusion initiel)
- 4) Si Alvin n’est pas conscient, le computatonnalisme est faux
- 5) Donc, le computationnalisme est faux

# Bonnes et mauvaises abstractions

- Réponses?
- A) Acceptez qu'Alvin soit conscient mais pas Théodore : les absences sont importantes pour la conscience. (le son du silence?)
- B) Appel à la réponse aux objections de trivialité. Il ne s'agit pas simplement de calculer cette fonction d'entrée/sortie (ce que n'importe quel UTM peut faire)... le mécanisme est important. (Alvin n'est pas conscient).