Les machines de Turing, ou comment découvrir la puissance des ordinateurs en jouant avec des briques de construction

... une idée farfelue de Marie Duflot-Kremer
créée en mars 2013

dernière modification le 17 août 2018

Le document que vous êtes en train de consulter n'est pas une référence très finalisée, ni un guide strict à suivre. Il regroupe différents éléments pour pratiquer l'activité autour de la machine de Turing. Vous pouvez suggérer des améliorations/enrichissements à `marie.duflot-kremer@loria.fr`. Si ce document vous a été utile, vous pouvez également me le signaler. La page `https://members.loria.fr/MDuflot/` permet de trouver (section médiation/activités) une liste d'autres activités pour faire découvrir différents aspects de l'informatique, réalisables en très grande majorité sans ordinateur.

Je tiens à remercier chaleureusement Jean-Marc Vincent pour avoir montré un grand enthousiasme en découvrant l'activité, et m'avoir suggéré une partie des défis présents dans ce document. C'est sans doute grâce à lui que j'ai bouclé la rédaction. Par contre il ne l'a pas (encore) lu et ne peut être tenu pour responsable des coquilles présentes ici ou là.

Étape 1 — Machine de Turing ? Quezaco ?

Vous avez peut-être déjà entendu parler d'Alan Turing, à la tête de l'équipe qui a réussi à déchiffrer des messages de la machine Enigma pendant la deuxième guerre mondiale, donnant ainsi un net avantage aux alliés ce qui a nettement réduit la durée du conflit.

Tout juste avant l'arrivée des premiers ordinateurs, ce même Alan Turing a eu l'idée en 1936 d'une machine abstraite (donc pas une vraie qu'on peut toucher, mais juste une description théorique), assez rudimentaire, composée d'un état interne, d'un ruban sur lequel on peut écrire/lire des lettres et de règles (le programme) disant comment faire évoluer cette machine. A priori rien de bien révolutionnaire, sauf qu'en fait si ! Pour comprendre pourquoi ces machines abstraites ont intéressé et intéressent toujours les informaticiens malgré les progrès énormes en termes de puissance d'ordinateurs, vous pouvez aller lire l'Etape 6.

Dans cette activité nous allons découvrir comment, avec ce modèle a priori très simple, on peut manipuler, voir travailler une machine de Turing précise, et même concevoir ses propres machines de Turing pour résoudre un problème donné. Et tout ça avec de petites briques de construction colorées.

Étape 2 — Le matériel

Pour réaliser l'activité, j'utilise le matériel suivant, à fournir pour chaque groupe de 2 (c'est mieux, chacun•e manipule plus) à 4 personnes. Pour info le modèle de briques Lego est tombé dans le domaine public. Il existe donc maintenant d'autres marques qui vendent des briques parfaitement compatibles.

- Une bande découpée dans une plaque support Lego de 4 plots de large et aussi longue que notre plaque (30 à 40cm ?),
- des briques de Lego 2 x 2 de plusieurs couleurs différentes (au moins 4), et une quinzaine de briques par couleur,

— une "tête de lecture", c'est à dire un objet que l'on mettra devant une brique pour indiquer où on en est. On peut utiliser une brique Lego sur laquelle on trace une flèche, ou un morceau de carton d'environ 3cm de long sur 1cm de large, taillé en pointe.
— un support permettant d'écrire et d'effacer. A minima du papier un crayon et une gomme, et au mieux une ardoise + craie/marqueur suivant le type.

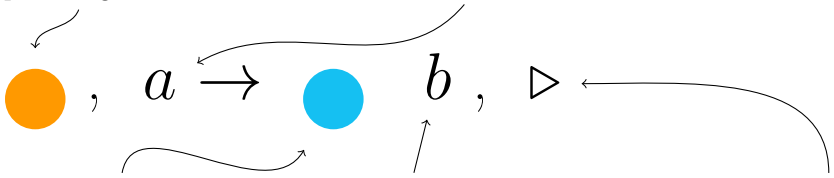
Au besoin si notre plaque n'est pas assez grande on peut scotcher ensemble deux bandes pour avoir notre "ruban".

Étape 3 — Fonctionnement d'une machine de Turing

Comme annoncé plus haut, notre machine va avoir plusieurs composantes. Tout d'abord elle a un **ruban** sur lequel on va écrire un "mot" en alignant côte à côte nos briques de couleur, et une **tête de lecture** qui va pointer sur une des briques (au départ la plus à gauche, la première de notre mot) et va pouvoir se décaler à gauche et à droite au fur et à mesure de l'exécution. Ensuite elle va avoir un **état de contrôle** qui va permettre de varier les comportements et va changer au cours du temps suivant nos besoins (cela sera plus clair sur un exemple).

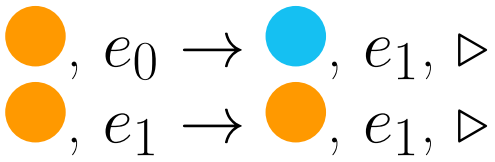
La machine de Turing va disposer d'un ensemble de règles, qui disent, en fonction de l'état de contrôle et de la couleur de la brique qui est en face de la tête de lecture, par quelle couleur il faut remplacer la brique, dans quel état de contrôle on passe, et de quel côté on décale la tête de lecture (à gauche $\triangleleft$ ou à droite $\triangleright$). Et voici comment lire une règle d'une machine de Turing.

Si on pointe sur une brique orange et la machine est dans l'état a ...



.. alors on la remplace par une bleue, on passe dans l'état b, et la tête de lecture va vers la droite.

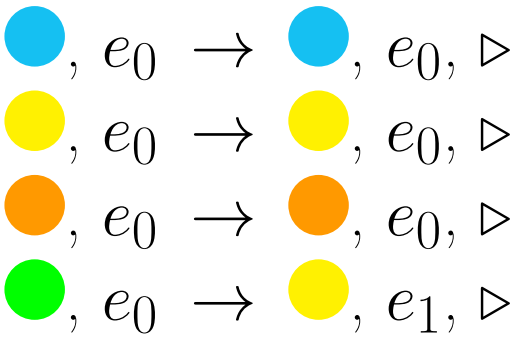
L'état de contrôle (on dit souvent juste "état") est ici nécessaire car sans lui, à chaque fois que l'on verrait une brique de la même couleur, par exemple orange, on ferait la même chose. Avec notre état on peut par exemple changer la première brique orange en une bleue mais ne pas changer les suivantes, comme on le voit avec les deux règles ci-dessous :



En effet, si l'état de départ est e_0, quand on voit une brique orange on la change en une bleue et on passe dans l'état e_1 par contre les briques oranges suivantes ne seront pas changées car maintenant notre machine est dans l'état e_1 et va appliquer la deuxième règle.

Étape 4 — Jouer avec des machines

On commence tout d'abord à expliquer le fonctionnement d'une machine, comme décrit dans la section précédente, en l'illustrant sur un exemple. Voici donc un ensemble de règles d'une machine de Turing qui va prendre en entrée un "mot" (= une suite de briques clipsées les unes à côté des autres sur le ruban) composé de briques bleues, jaunes, oranges et vertes.



On remarque que, pour cette machine, il n'y a aucune règle qui s'applique quand on est dans l'état e_1, et donc si en suivant les règles on finit par arriver dans l'état e_1, la machine ne pourra plus appliquer de règle et s'arrêtera. De même, j'ai fait le choix de donner des noms génériques aux états de contrôle puis d'expliquer le sens qu'ils ont dans une machine. Cela permet de voir que pour la machine, le nom n'a aucune importance. Je peux appeler du même nom des états dans deux programmes différents qui font des choses totalement différentes. Et c'est du coup une partie du travail pour les participants, en voyant une machine, de comprendre à quoi correspond chaque état.

On va maintenant regarder ce que donne l'exécution de cette machine sur un mot donné en entrée. En pratique avec des enfants il faut absolument qu'elles et ils aient un espace pour écrire l'état de contrôle. Une ardoise s'y prête bien, et si elle est assez grande elle permet aussi d'écrire les petits programmes.

Chaque ligne correspond à la configuration de la machine à un moment donné. On représente une configuration en notant d'abord l'état de contrôle en début de ligne, puis la suite de couleurs sur le ruban (avec un disque souligné pour désigner celui pointé par la tête de lecture). La figure 1 donne donc deux exemples d'exécution, pour deux mots différents.

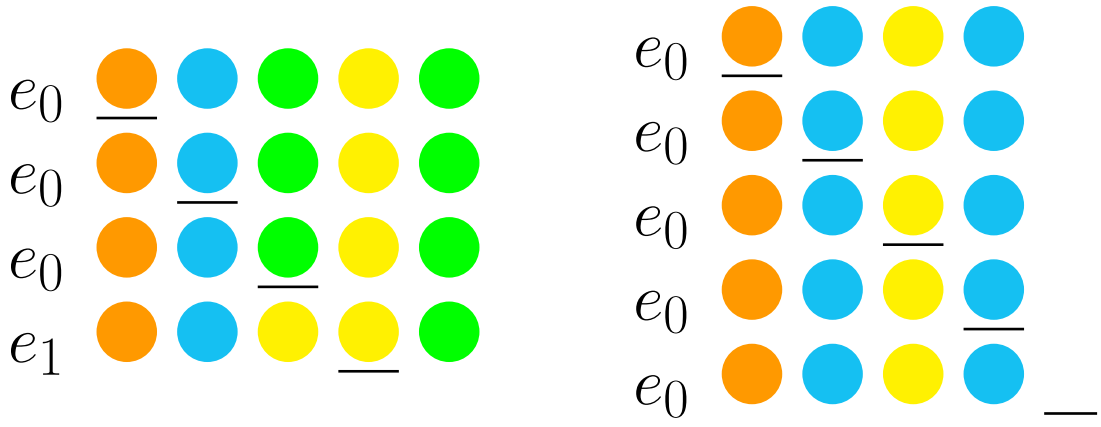


FIGURE 1 – Deux exécutions de notre machine de Turing.

Revenons sur les exécutions de la figure 1. Sur celle de gauche la tête de lecture commence sur une brique orange et dans l'état initial e_0. On applique donc la troisième règle qui nous dit de ne pas changer la couleur de brique, ni l'état, mais de se décaler à droite. On passe ainsi les deux premières briques sans changer leur couleur jusqu'à tomber sur une verte, qui en suivant la quatrième règle est remplacée par une brique jaune pendant que la machine passe dans l'état e_1. Comme toutes les règles de la machine ne s'appliquent que quand l'état est e_0, la machine ne peut plus exécuter de règle et s'arrête après trois transitions.

Pour l'exécution de droite, on commence avec la tête de lecture sur une brique orange également et dans l'état e_0. Comme le ruban ne contient que des briques oranges, bleues et jaunes on va appliquer les règles 1 à 3 de la machine qui ne changent ni l'état ni la couleur des briques. La tête de lecture se décale d'un rang à droite à chaque fois, et finit par dépasser la dernière brique et pointer sur un endroit

vide. Il est en théorie possible de définir une règle à appliquer quand on est sur une case vide du ruban, comme le montre la figure 2 (dans mon cas le ruban est gris). Comme dans notre exemple aucune règle ne s'applique quand on n'a plus de brique en face de la tête de lecture, la machine s'arrête.

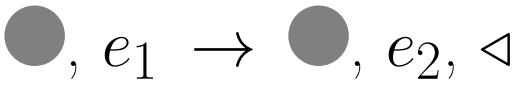


FIGURE 2 – Exemple de règle qui s'applique quand la tête de lecture ne pointe pas sur une brique.

Une fois le principe expliqué, et après avoir montré un exemple d'exécution aux enfants, l'idée est de leur donner le matériel et de les laisser jouer avec un ou plusieurs exemples de machines. Pour chacune d'entre elles une version grand format est donnée dans l'étape 7, d'abord en couleurs puis sans couleurs pour, au choix :

— pouvoir l'imprimer sur une imprimante noir et blanc puis colorier,
— changer les couleurs que j'ai choisies pour les remplacer par les vôtres.

Pour chacune des machines ci-dessous, distribuez-la aux élèves , en précisant le type de mot en entrée (= les couleurs autorisées, et éventuellement le type de mot (par exemple la machine ne travaille que sur des mots ayant au maximum une brique jaune)), et la liste des états de contrôle autorisés, et demandez-leur à quoi correspond selon eux chaque état de contrôle et ce que fait la machine.

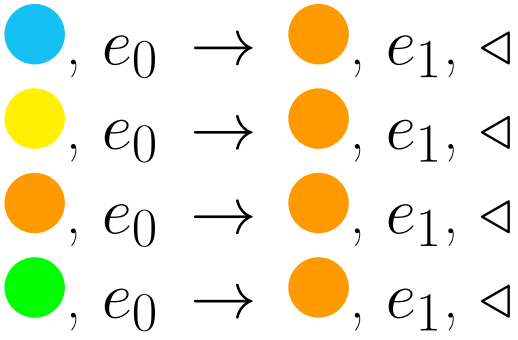


FIGURE 3 – Machine 1

La machine 1 Figure 3 prend en entrée un mot écrit avec les 4 couleurs. La description des états est la suivante :

— e_0 est l'état initial,
— e_1 est l'état final, dans lequel la machine va s'arrêter

La machine va remplacer la première brique (quelle que soit sa couleur) par une brique orange puis s'arrêter, car aucune règle ne s'applique quand elle est dans l'état e_1.

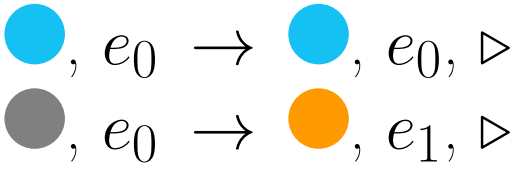


FIGURE 4 – Machine 2

La machine 2 Figure 4 prend en entrée un mot écrit avec uniquement la couleur bleue. La description des états est la suivante :

— e_0 est l'état initial,
— e_1 est l'état final, dans lequel la machine va s'arrêter

Notre machine va passer une à une les briques du mot (on rappelle qu'elles sont toutes bleues) puis quand elle arrive au bout du mot elle ajoute une brique orange et s'arrête. La machine va donc juste rajouter une brique orange en fin de mot.

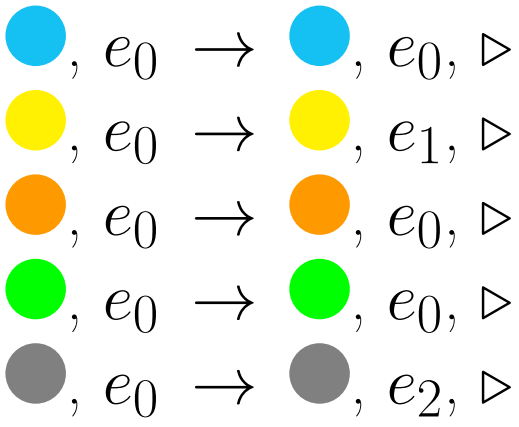


FIGURE 5 – Machine 3

La machine 3 Figure 5 prend en entrée un mot écrit avec nos 4 couleurs. La description des états est la suivante :

- e_0 est l'état initial,
- e_1 signifie que l'on a rencontré une brique de couleur jaune
- e_2 signifie qu'on a atteint la fin du mot (sans rencontrer de brique jaune sinon l'état aurait changé et la machine se serait arrêtée)

On a donc une machine dont le travail est de détecter si un mot contient au moins une brique jaune. La réponse (oui : e_1 ou non : e_2) se voit dans l'état de contrôle à la fin de l'exécution.

Étape 5 — Concevoir ses propres machines

Après avoir expérimenté le fonctionnement d'une machine de Turing sur plusieurs ensembles de règles, le but est de faire construire aux participants leurs propres machines de Turing. Voici ci-dessous des exemples de défis à réaliser. Libre à vous d'en inventer d'autres (et de me les envoyer pour que j'étoffe cette liste :o)).

Attention : si on précise le genre de mot qui est accepté en entrée (pas de vert par exemple) et qu'on fait tourner notre machine sur un mot contenant une brique verte, alors elle n'est pas obligée de bien se comporter, vu qu'on n'a pas respecté les règles. L'étape 9 (à ne PAS lire tant qu'on n'a pas bien cherché) donne par exemple pour le premier défi une machine qui n'a pas de règle qui s'applique à la vue d'une brique verte. Elle va donc s'arrêter sans rien faire dans ce cas.

1. détecteur d'orange :
   Entrée : un mot composé de couleurs parmi orange, jaune et bleu,
   Objectif : ajoute une brique verte en fin de mot (= après la dernière brique) si le mot ne contient pas de brique orange, ne change rien sinon
2. détecteur d'orange (2) :
   Entrée : un mot composé de couleurs parmi orange, jaune et bleu,
   Objectif : ajoute une brique verte en fin de mot (= après la dernière brique) si le mot contient une brique orange, ne change rien sinon
3. détecteur d'orange (3) :
   Entrée : un mot non vide composé de couleurs parmi orange, jaune et bleu,
   Objectif : remplace la dernière brique par une verte s'il y a une brique orange dans le mot, ne change rien sinon
4. détecteur d'orange (4) :
   Entrée : un mot composé de couleurs parmi orange, jaune et bleu,

Objectif : ajoute une brique verte en début de mot (= avant la première brique) s'il y a une brique orange dans le mot, ne change rien sinon. Attention, dans les machines de Turing il y a toujours de l'espace libre (illimité) avant et après le mot. Il faut donc bien mettre sa tête de lecture sur la première brique du mot mais laisser de l'espace à gauche et à droite du mot sur votre ruban.

5. parité :
   Entrée : un mot composé de couleurs parmi orange, vert, jaune et bleu,
   Objectif : l'état final doit être différent selon si le nombre de briques du mot est pair ou impair.
6. parité verte :
   Entrée : un mot composé de couleurs parmi orange, vert, jaune et bleu,
   Objectif : idem que ci-dessus mais on doit avec l'état de contrôle à la fin savoir si le nombre de briques vertes du mot est pair ou non (les autres couleurs importent peu),
7. pas deux de suite :
   Entrée : un mot composé de couleurs parmi orange, jaune et bleu,
   Objectif : si le mot contient deux briques de suite de la même couleur, la machine ajoute une brique verte en fin de mot
8. pas deux de suite (2) :
   Entrée : un mot composé de couleurs parmi orange, jaune et bleu,
   Objectif : si le mot contient deux briques de suite de la même couleur, la machine remplace la deuxième brique par une verte et continue
9. pas deux de suite (3) :
   Entrée : un mot composé de couleurs parmi orange, vert, jaune et bleu,
   Objectif : avoir en fin d'exécution un mot qui ressemble au maximum au mot d'entrée mais n'a plus deux briques de suite de la même couleur (indice, quand on remplace une brique par une autre, attention à la nouvelle couleur choisie)
10. incrémenteur :
    Entrée : un mot composé de couleurs parmi jaune et bleu
    Objectif : on considère qu'une brique jaune vaut 0 et une brique bleue vaut 1. Un mot sur le ruban est ainsi l'écriture binaire d'un nombre, par exemple 0110 est l'écriture binaire de 6 et 10111 l'écriture binaire de 23. Réaliser un incrémenteur (qui ajoute 1 à un nombre donné). L'algorithme est comme en décimal, sauf qu'on n'a que deux chiffres : $0 + 1 = 1$ et $1 + 1 = 0$ et je retiens 1. Par exemple $0100 + 1 = 0101$, et $0101 + 1 = 0110$ (on a propagé une fois la retenue).

Je ne suis pas une bone cobaye pour juger de la difficulté des défis, mais ils ne sont pas rangés par ordre croissant de difficulté.

Étape 6 — Et hop... c'est de l'informatique !

Avec cette activité le lien avec l'informatique est clair : on exécute et on écrit des programmes, OK. Mais la question qui reste est de savoir quel est l'intérêt de continuer à utiliser ce modèle rudimentaire inventé au milieu des années 30.

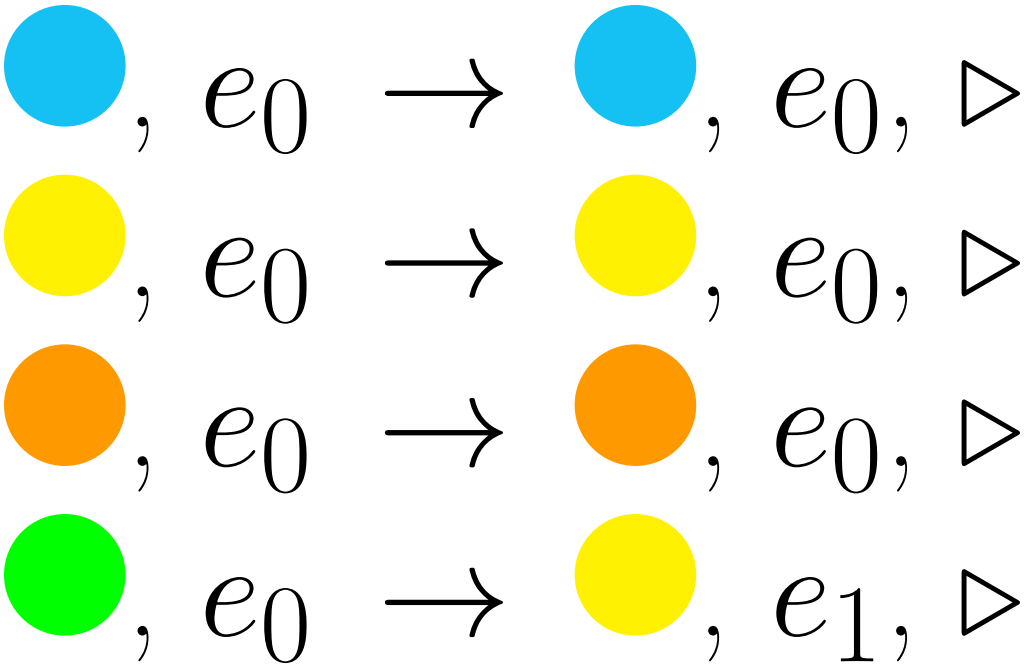
Pourquoi s'en souvenir plus de 80 ans plus tard alors que nous avons des ordinateurs très puissants que nous programmons dans des langages dits "évolués", bien plus proche du langage naturel ? Parce que ce modèle, si rudimentaire soit-il, a la même capacité de calcul que n'importe quel ordinateur. Oui nos ordinateurs modernes sont bien plus rapides, efficaces, agréables à programmer, mais ce qui est incroyable c'est que pour tout problème qu'un ordinateur sait résoudre, si complexe soit-il, on peut concevoir une machine de Turing (sans doute énorme et peu efficace) qui fait la même chose.

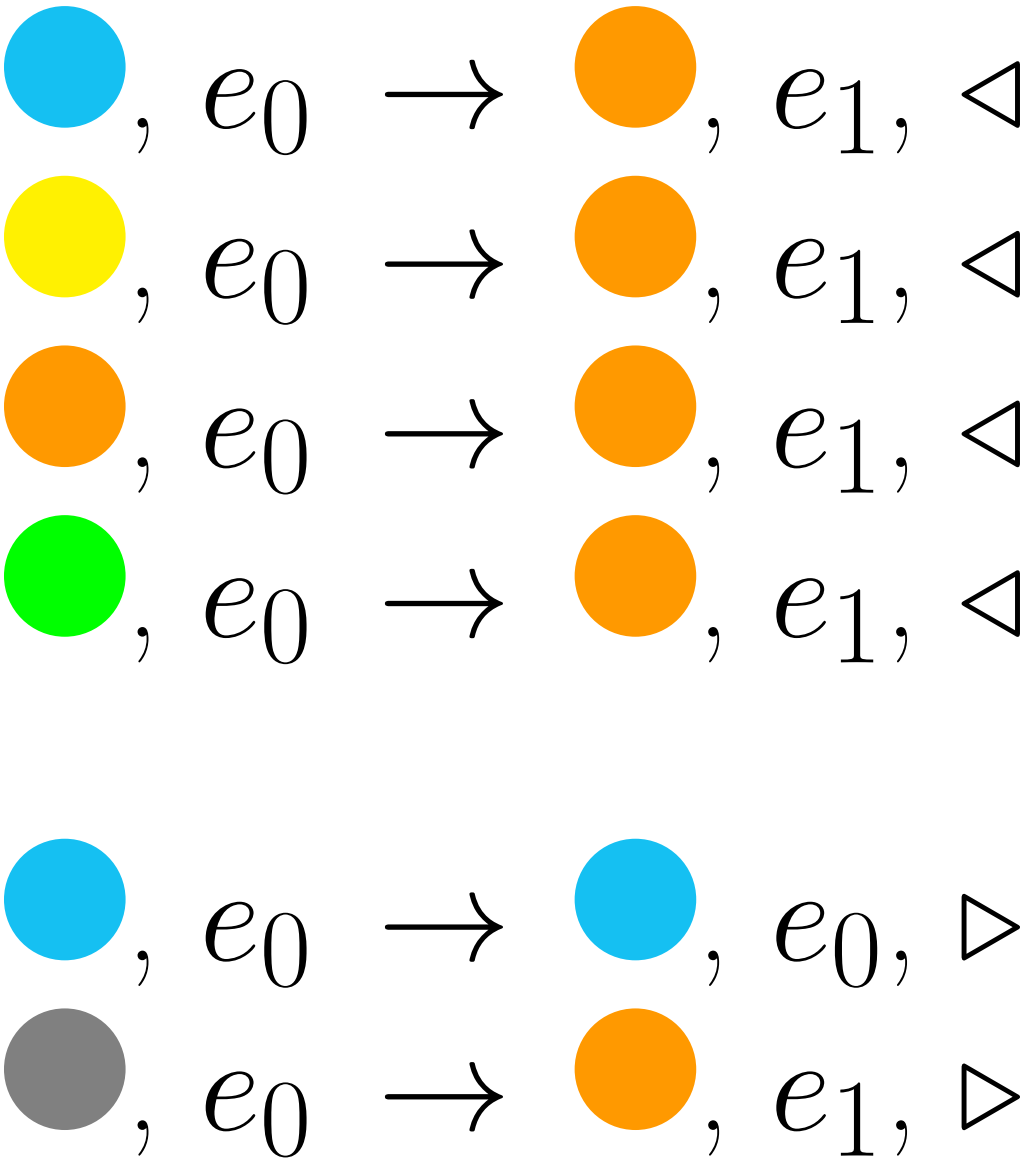
Si vous ne voyez toujours pas l'intérêt cela va venir. Imaginez maintenant que vous trouvez un problème pour lequel vous réussissez à prouver qu'aucune machine de Turing n'est capable de le résoudre. Avec cette preuve et le fait que la machine de Turing sait faire la même chose que les ordinateurs, on obtient automatiquement le résultat suivant : aucun ordinateur, même si dans 100 ans ils sont bien plus puissants, ne pourra résoudre ce problème. Et faire cette preuve sur les machines de Turing est rendu plus simple car le modèle de ces machines est très précis et les "instructions" ou règles sont d'un type très restreint.

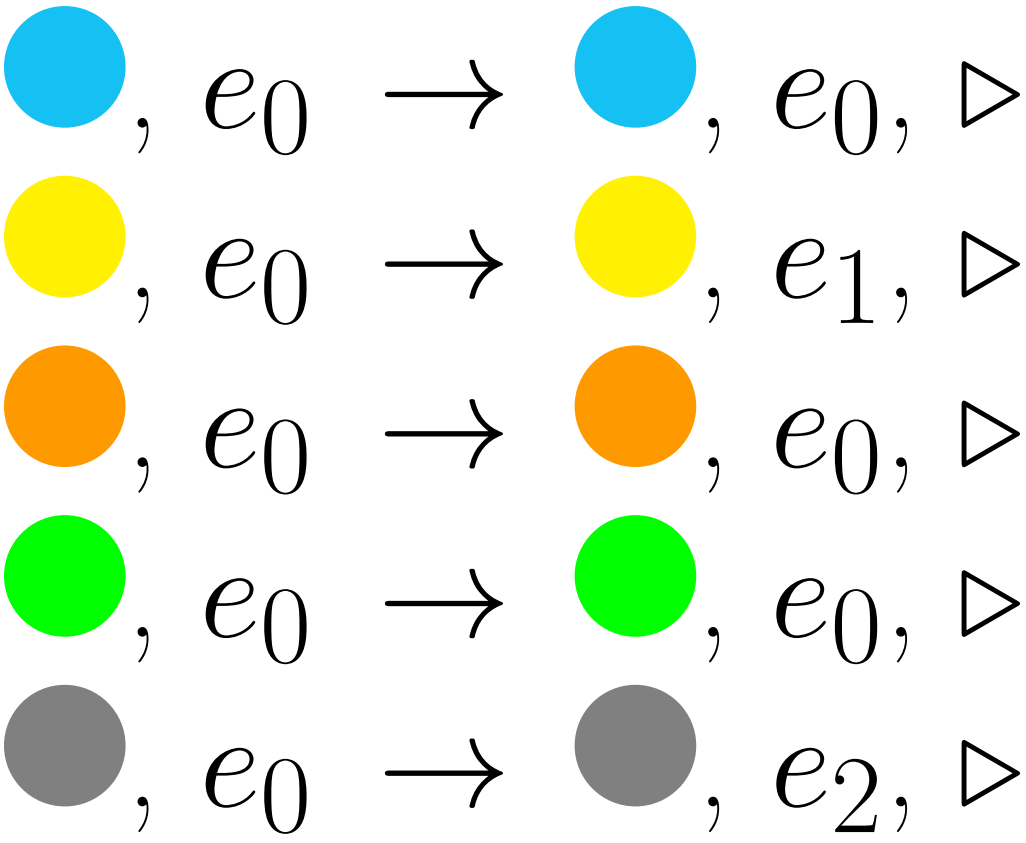
Il existe donc des problèmes, qui n'ont pas forcément l'air très compliqués quand on les décrit, pour lesquels on sait qu'il n'est pas la peine d'essayer d'écrire un programme pour les résoudre, on sait qu'un tel programme n'existera pas. Et tout ça grâce aux machines de Turing !

Étape 7 _____ Quelques exemples de machines à imprimer

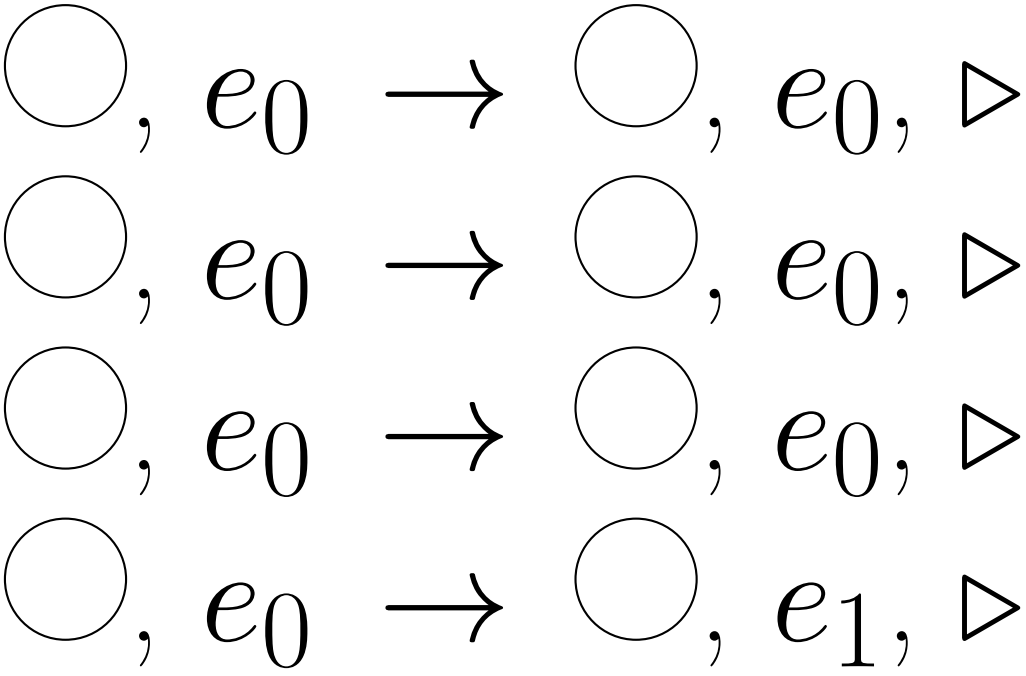
Dans l'ordre, leur rôle est de :
- remplacer la première brique verte du mot par une jaune
- remplacer la première brique du mot par une orange (quelle que soit sa couleur)
- si le mot ne contient que des briques bleues, on ajoute une brique orange à la fin (sinon on ne fait rien)
- si le mot contient une brique orange, la machine passe dans l'état e_1, si il n'en contient pas la machine passe dans l'état e_2

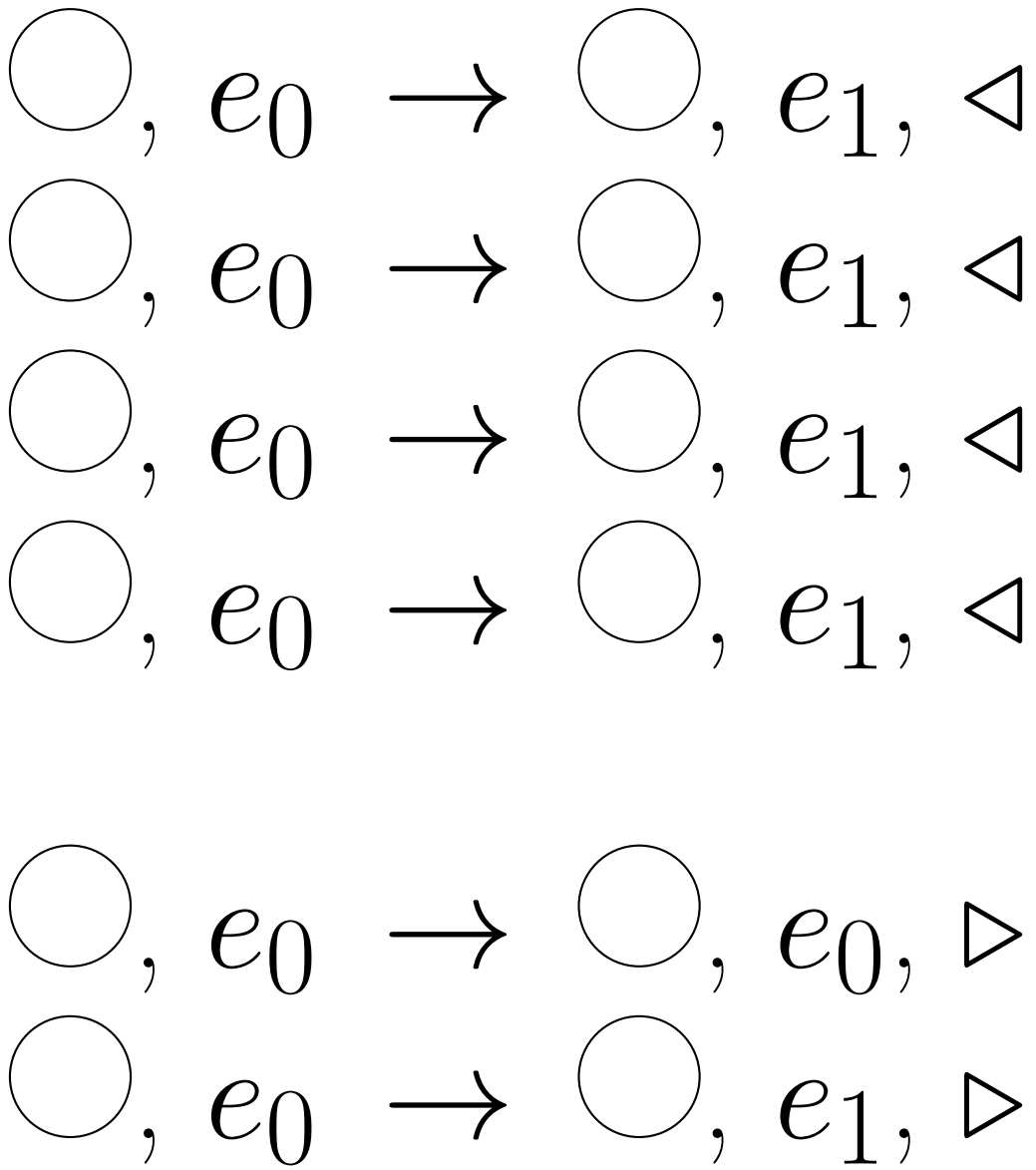


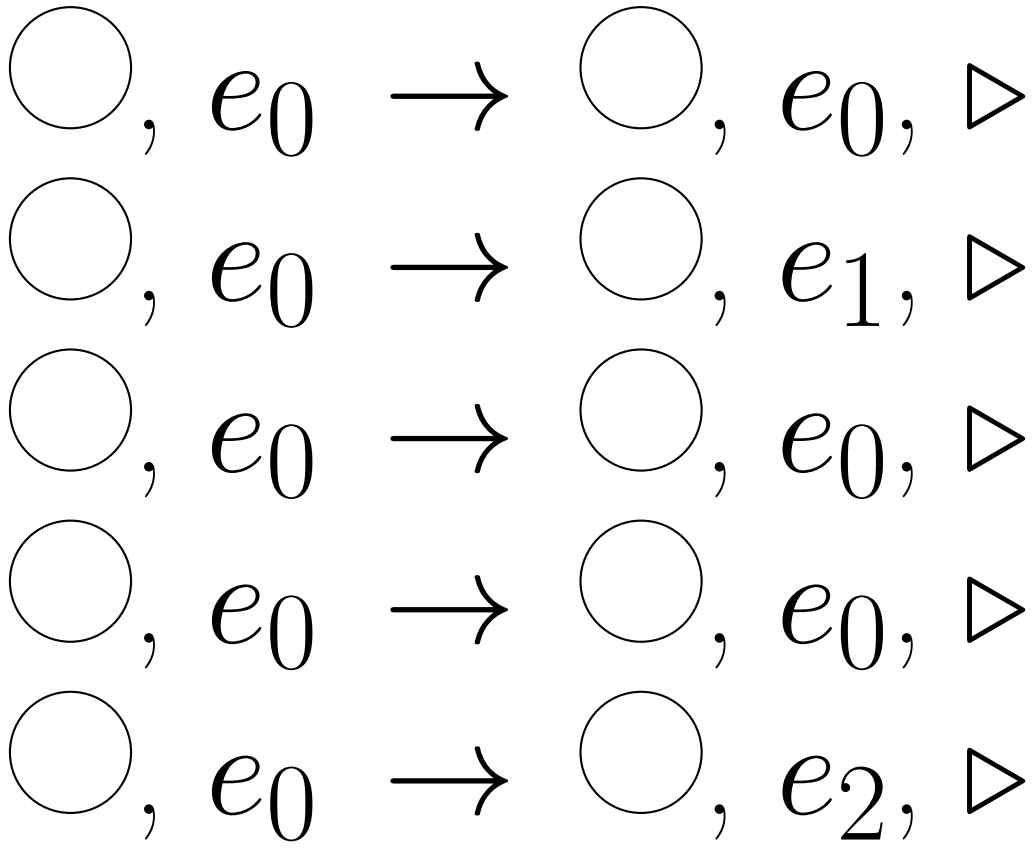

, e_0 → , e_1, ◁
, e_0 → , e_1, ◁
, e_0 → , e_1, ◁
, e_0 → , e_1, ◁
, e_0 → , e_0, ▷
, e_0 → , e_1, ▷



Étape 8 . Les mêmes machines mais sans couleurs (attention à ne pas se planter au coloriage)




, e_0 → , e_1, ◁
, e_0 → , e_1, ◁
, e_0 → , e_1, ◁
, e_0 → , e_1, ◁
, e_0 → , e_0, ▷
, e_0 → , e_1, ▷

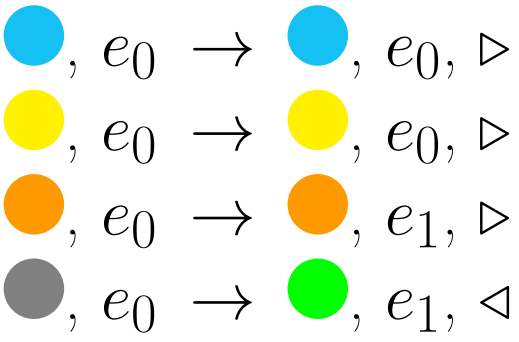


Étape 9 — Solutions des défis

A ne pas lire tant qu'on ne s'est pas bien creusé la tête dessus

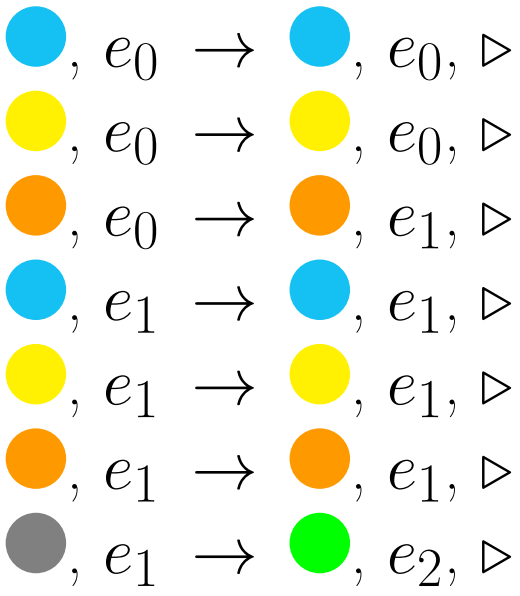
Rappel : en programmation (d'ordinateur ou de machine de Turing), il y a toujours plusieurs solutions à un même problème. Vous pouvez donc avoir d'autres solutions que les miennes, et pourquoi pas plus simples/élégantes/courtes. Si vous pensez que c'est le cas vous pouvez me les envoyer, je pourrais pourquoi pas les ajouter ou remplacer les miennes par les vôtres.

1. détecteur d'orange :



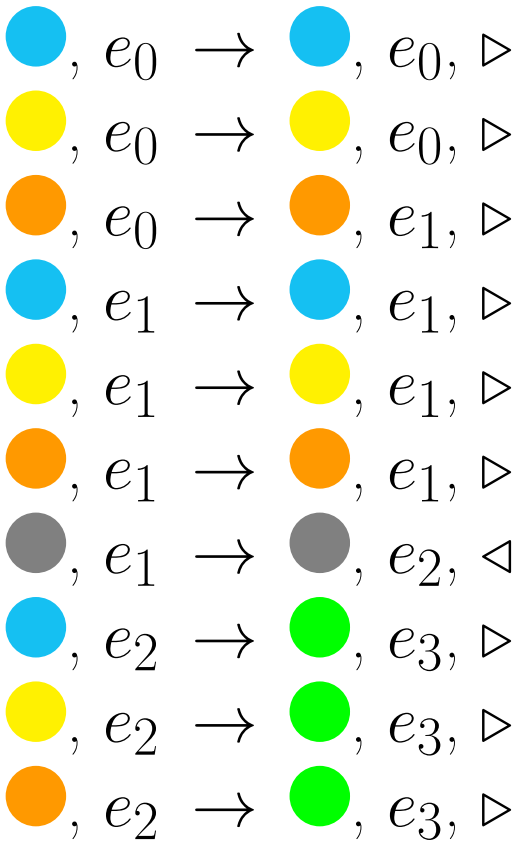
Signification des états :
— e_0 : pas encore vu de brique orange
— e_1 : fini
Tant qu'elle est dans l'état e_0, la machine va passer toutes les briques de couleur sans les changer, et passer dans l'état e_1 si elle voit une brique orange. Si on finit de lire le mot (on arrive sur le ruban, donc couleur grise) et donc si on n'a pas rencontré de brique orange, on ajoute une brique verte, et on passe dans l'état e_1 où plus aucune action n'est possible.

2. détecteur d'orange (2) :



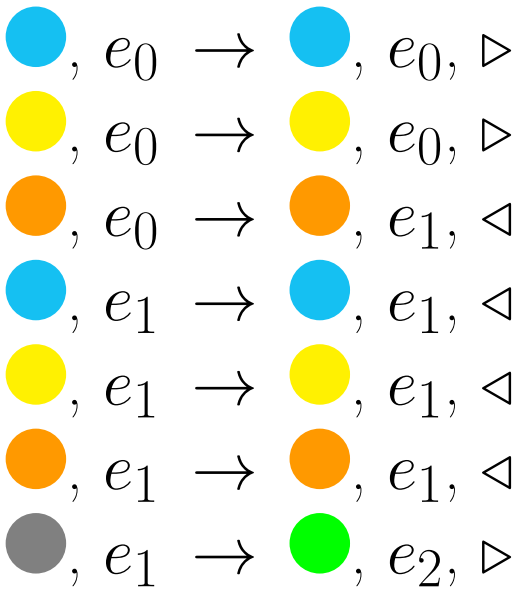
Signification des états :
— e_0 : pas encore vu de brique orange
— e_1 : vu une brique orange
— e_2 : fini
La machine va passer toutes les briques de couleur sans les changer, et passer dans l'état e_1 si elle voit une brique orange. Comme on veut que, même après avoir croisé une brique orange elle continue, on rajoute des règles pour continuer à avancer sans changer d'état même quand on est dans l'état e_1. Quand on a fini de lire le mot (on arrive sur le ruban, donc couleur grise), si l'état est e_1 (car on a croisé une brique orange) on ajoute une brique verte, et on passe dans l'état e_2 où plus aucune action n'est possible.

3. détecteur d'orange (3) :



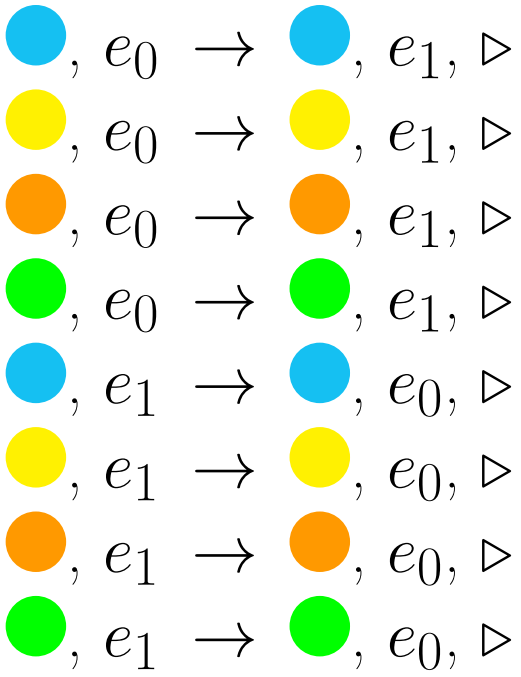
Signification des états :
— e_0 : pas encore vu de brique orange
— e_1 : vu une brique orange
— e_2 : je vais remplacer la prochaine brique par une verte
— e_3 : fini
La machine commence comme la précédente, elle va passer toutes les briques de couleur sans les changer jusqu'à la fin du mot, en passant dans l'état e_1 si elle voit une brique orange. Quand on a fini de lire le mot, si l'état est e_1 (car on a croisé une brique orange), on revient une brique en arrière et on la remplace (quelle que soit sa couleur) par une brique verte, puis on s'arrête.

4. détecteur d'orange (4) :



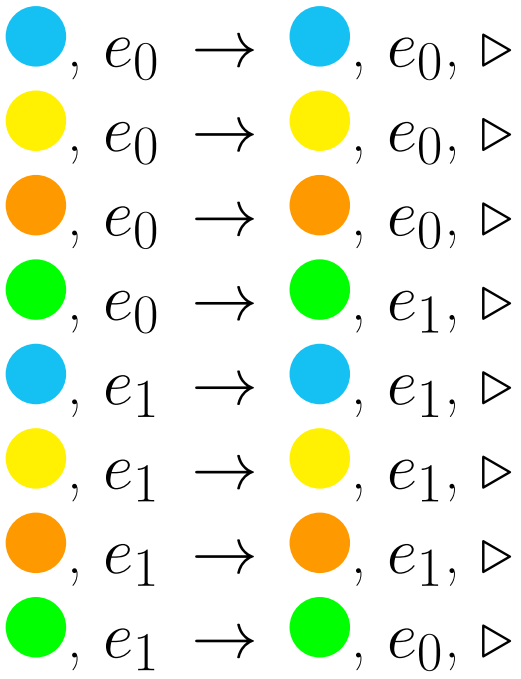
Signification des états :
— e_0 : pas encore vu de brique orange
— e_1 : vu une brique orange (et donc on repart vers la gauche)
— e_2 : fini
La machine commence comme la précédente, elle va passer toutes les briques de couleur sans les changer, jusqu'à la fin du mot ou jusqu'à rencontrer une brique orange, passer dans l'état e_1 et faire demi tour. Quand la tête de lecture arrive sur le ruban, soit on est dans l'état e_0 (et donc à droite du mot) et on ne peut rien faire donc on s'arrête, soit on est dans l'état e_1 (à gauche du mot) et on ajoute une brique verte avant de s'arrêter.

5. parité :



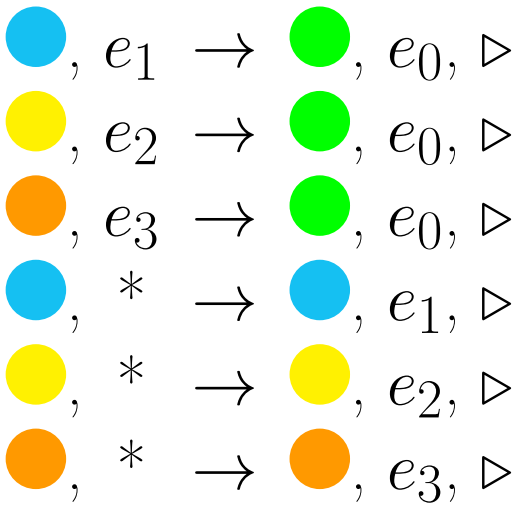
A chaque nouvelle brique on passe à la suivante en alternant entre les états e_0 (pair) et e_1 (impair). Quand on a lu toutes les briques du mot on s'arrête et l'état de contrôle donne la réponse.

6. parité verte :



Comme précédemment, à chaque nouvelle brique on passe à la suivante mais on ne change d'état entre e_0 (pair) et e_1 (impair) qu'à la vue d'une brique verte. Quand on a lu toutes les briques du mot on s'arrête et l'état de contrôle donne la réponse.

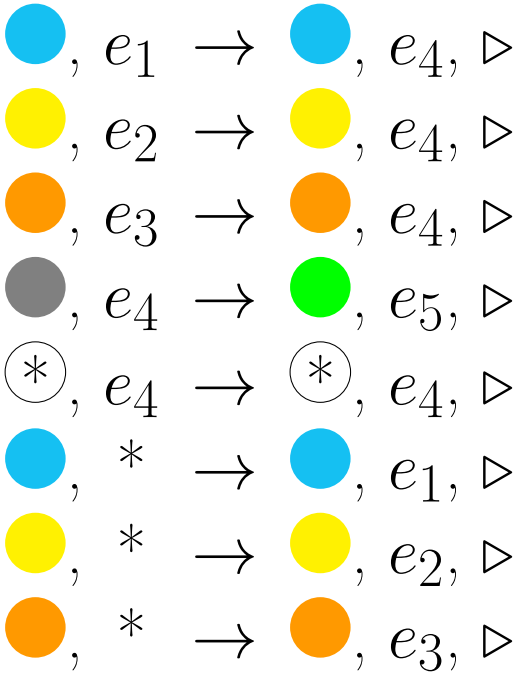
7. pas deux de suite (2) :
   Pour cette machine et les suivantes, si on veut vraiment écrire toutes les règles, c'est très/trop long. Du coup je propose les conventions suivantes. Tout d'abord, à chaque étape, on applique la première dont les conditions sont satisfaites. Cette convention nous permet d'en introduire une deuxième. Le signe "*" représente un état donné, peu importe lequel, et de même le symbole  désigne une couleur donnée, peu importe laquelle. Ainsi dans la machine suivante, la dernière règle s'applique quand on a une brique orange et qu'on est dans n'importe quel état... sauf e_3 sinon on applique la règle numéro 3 qui est avant celle-ci dans la liste.



Signification des états :
— e_0 : pas encore vu de brique ou la précédente est verte
— e_1 : la brique précédente est bleue
— e_2 : la brique précédente est jaune
— e_3 : la brique précédente est orange

Dans cette machine, l'état de contrôle sert à mémoriser la couleur de la dernière brique vue (voir la liste ci-dessous). Comme aucune brique dans le mot n'est verte, on ne traite pas ce cas, et l'état e_0 regroupe le cas où on n'a pas vu de brique ou bien on vient de poser une brique verte. Du coup si on voit une brique bleue sur le ruban, il y a deux cas possibles : soit la précédente était bleue (donc on est dans l'état e_1) et il faut donc remplacer la brique actuelle par une verte, soit elle était d'une autre couleur (état tout sauf e_1) et on peut avancer, en mémorisant la couleur de cette nouvelle brique dans notre état. On fait pareil pour les trois couleurs possibles dans le mot et le tour est joué.

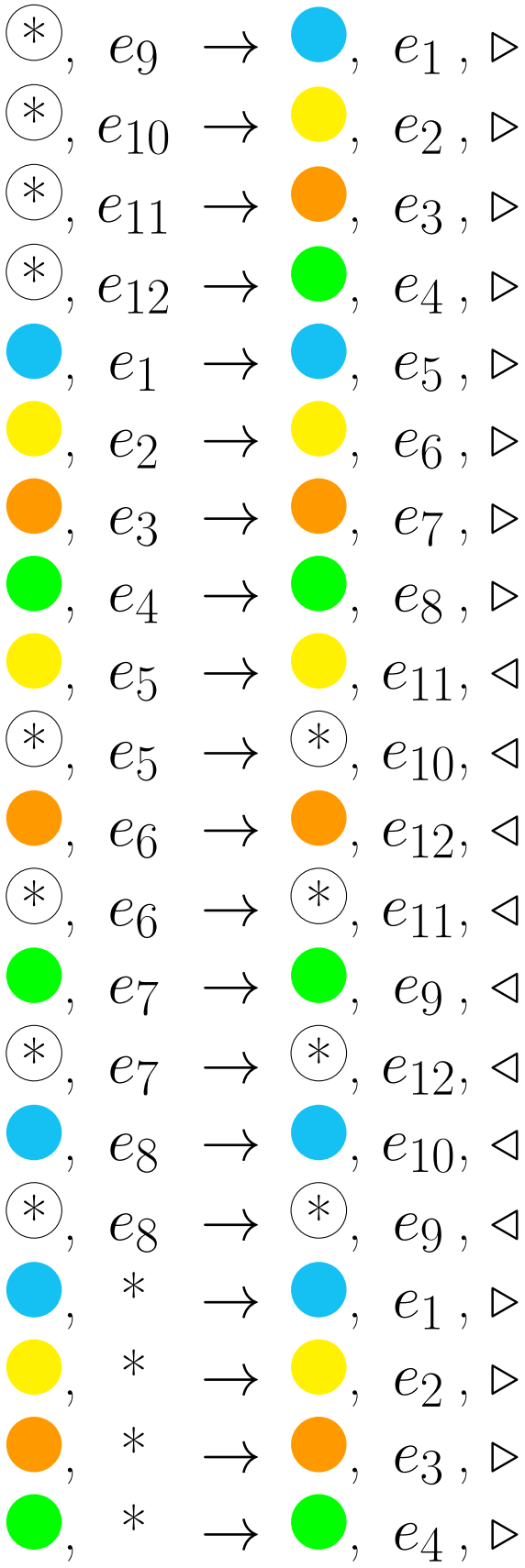
8. pas deux de suite :
   Dans cette machine on utilise les mêmes conventions que dans la précédente. La cinquième règle signifie que quand la machine est dans l'état e_4, quelle que soit la couleur de brique qu'on voit (sauf le gris du ruban vide sinon on applique la quatrième règle) on ne change pas la brique ni l'état et on avance à droite.



   Signification des états :
   — e_0 : pas encore vu de brique ou la précédente est verte
   — e_1 : la brique précédente est bleue
   — e_2 : la brique précédente est jaune
   — e_3 : la brique précédente est orange
   — e_4 : on a détecté deux briques de suite de la même couleur
   — e_5 : fini
   Dans cette machine, l'état de contrôle sert toujours à mémoriser la couleur de la dernière brique vue cela permet de détecter qu'on est en train de voir la deuxième brique de suite de la même couleur et ainsi passer dans un état destiné à signaler cette double couleur. Quand on a fini de lire le mot, soit aucune règle s'applique (états e_0 à e_3) et on s'arrête, soit on a détecté une double couleur (état e_4) et on ajoute une brique verte avant de s'arrêter.

9. pas deux de suite (3) :
   L'idée de cette machine est que quand on voit deux briques de suite de la même couleur, on ne peut pas remplacer la deuxième directement, sinon on risquerait de la mettre de la même couleur que la suivante qu'il faudrait aussi changer. On va donc voir la brique d'après, et en fonction de sa couleur, décider par quoi on remplace la brique en doublon. Dans mon exemple ci-dessous, j'essaie par défaut de remplacer une brique bleue par une jaune, une jaune par une orange, une orange par une verte et une verte par une bleue (dans l'ordre alphabétique des couleurs en fait). Si la brique suivant le doublon est de ma couleur par défaut, alors je prends la suivante dans l'ordre alphabétique. Il y a évidemment plusieurs méthodes possibles.

On utilise toujours la même convention mais malgré cela il y a une vingtaine de règles :



Signification des états :
— e_0 : pas encore vu de brique
— e_1 : la brique précédente est bleue
— e_2 : la brique précédente est jaune
— e_3 : la brique précédente est orange
— e_4 : la brique précédente est verte
— e_5 : les deux briques précédentes sont bleues
— e_6 : les deux briques précédentes sont jaunes
— e_7 : les deux briques précédentes sont oranges
— e_8 : les deux briques précédentes sont vertes
— e_9 : il faut remplacer la brique par une bleue
— e_{10} : il faut remplacer la brique par une jaune
— e_{11} : il faut remplacer la brique par une orange
— e_{12} : il faut remplacer la brique par une verte

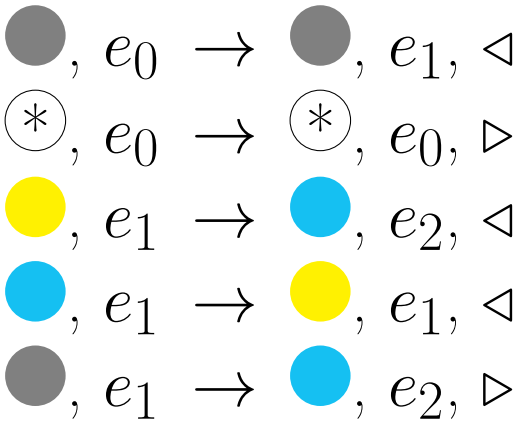
Les quatre premières règles servent quand on sait qu'il faut remplacer une brique (quelle que soit sa couleur) par une brique donnée. Elles font le remplacement, mémorisent la couleur de la nouvelle brique et passent à la suivante.

Les quatre règles qui suivent correspondent au moment où on voit une deuxième brique consécutive de la même couleur. On ne peut pas encore la remplacer (il faut regarder la couleur de la suivante), mais on mémorise dans l'état de contrôle la couleur du doublon. Ensuite les quatre paire de règles de la ligne 9 à la ligne 16 servent, quand on a vu un doublon (donc on est dans un état parmi e_5, e_6, e_7 et e_8) à décider par quelle couleur on va remplacer la deuxième brique. Par exemple dans l'état e_5 (on a vu un doublon bleu), on veut remplacer par défaut le deuxième bleu par un jaune. Si la brique suivante est jaune (règle 9), on décide que la couleur de remplacement est orange (état e_{11}), dans les autres cas, y compris le gris du ruban vide, (règle 10) on choisit le jaune (état e_{10}). Et surtout on n'oublie pas de faire aller la tête de lecture vers la gauche pour la mettre en face de la brique à remplacer.

Les quatre dernière règles s'appliquent dans tous les autres cas, c'est à dire quand les briques consécutives ont des couleurs différentes.

10. incrémenteur :

Pour finir voici la machine qui permet d'imaginer comment la machine de Turing sait faire tout ce que peut faire un ordinateur. Ici on commence à calculer.



Signification des états :
— e_0 : on parcourt le nombre/mot jusqu'au bout (à droite)
— e_1 : il faut ajouter 1 au chiffre/brique
— e_2 : fini

Pour incrémenter un nombre, il faut déjà aller voir son chiffre des unités. Comme il est tout à droite et qu'au départ la tête de lecture pointe sur la brique tout à gauche, on va rester dans l'état e_0 en parcourant tout le mot, sans changer aucune brique. Quand on atteint une case vide, on change d'état (e_1) pour passer en mode calcul.

Dans l'état e_1 si on rencontre un 0/brique jaune, on lui ajoute 1 ce qui donne une brique bleue et

on s'arrête. Si par contre on rencontre un 1/bleu, on le change en 0/jaune, on se décale à gauche pour aller ajouter la retenue 1 au chiffre suivant et on continue. Si notre nombre était de la forme 1111 (que des chiffres à 1), on va tous les changer un par un en 0, et on va arriver sur le ruban vide à gauche de notre nombre. Il faut donc une dernière règle qui écrit un 1 à cette place avant de s'arrêter. Du coup $1111 + 1 = 10000$ (en fait c'est comme en décimal si on veut calculer $99 + 1 = 100$, on doit ajouter un chiffre, le résultat ne tient plus sur deux chiffres).