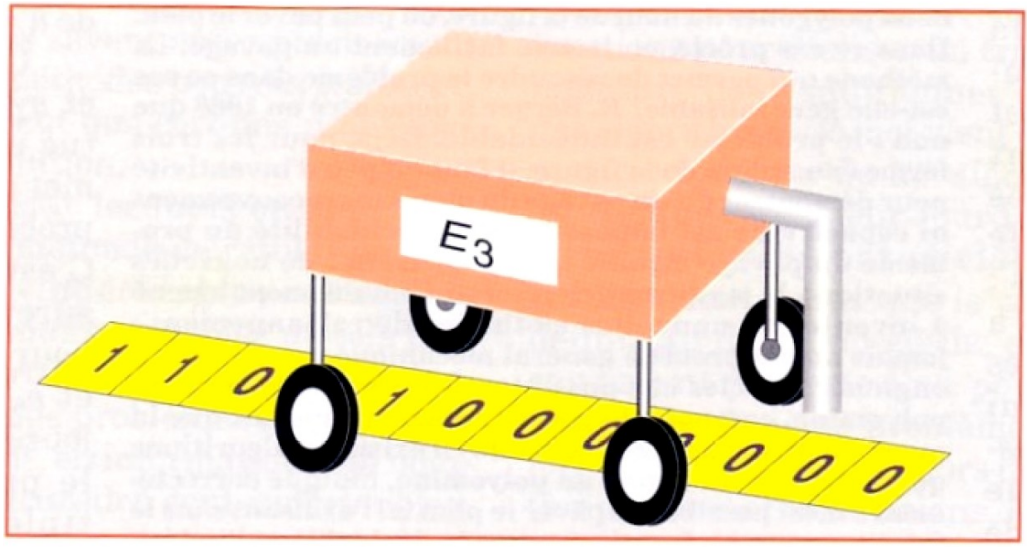
	**Algorithmique**	**NSI T^{ale}**
	Machine de Turing	cours/TD

*

1. La notion de machine de Turing

Il nous faut bien en arriver au cœur du problème : comment les mathématiciens ont-ils réussi à formuler la définition de procédé de calcul ?
Nous nous limiterons à la méthode dite "des machines de Turing", du nom du mathématicien anglais **Alan Turing** qui introduisit cette notion en **1936**.
Le concept de machine de Turing est à la fois simple et puissant. C'est en fait le plus simple des mécanismes universels de calcul qu'on puisse envisager.
Qu'il soit simple, cela va apparaître dans la définition.
Qu'il soit universel, c'est-à-dire qu'il permette effectivement de programmer tout algorithme cela ne fait plus aucun doute.

Une ***machine de Turing*** `M` est un mécanisme idéalisé destiné à effectuer des calculs, comme par exemple calculer `n+2` ou `3n` quand on lui donne `n`.
Pour effectuer ses calculs la machine de Turing `M` utilise **un ruban illimité découpé en cases**, et **une tête de lecture-écriture** qui lui permet de **lire**, d'**effacer** et d'**écrire** sur le ruban.
Cette tête de lecture-écriture peut se déplacer sur le ruban et c'est la machine elle-même qui commande les déplacements.



UNE MACHINE DE TURING est un mécanisme qui possède un nombre fini d'états intérieurs et qui, selon l'état où il se trouve et selon ce qu'il lit sur le ruban, efface la case du ruban qui est sous sa tête de lecture-écriture, y écrit un symbole et se déplace vers la gauche ou vers la droite. Le programme de la machine est une suite finie d'instructions du type : «Si je suis dans l'état E_3 et si je lis un 0 sur le ruban, alors je le remplace par un 1, je me déplace d'une case vers la droite et je passe dans l'état E_2 ; en abrégé, on note une telle instruction (E_3 0 → E_2 1 D). Tout calcul peut être exécuté par une machine de Turing.

Pour la machine destinée à calculer `3n` on écrira `n` sur le ruban, puis une fois le calcul terminé on devra y lire `3n`.
Pour une machine destinée à résoudre un autre problème comme celui des pavages par polygones composés de carrés juxtaposés, on écrirait sous une forme codée les données géométriques du problème, et on devrait lire après le calcul la réponse `OUI` ou `NON`.
Une machine `M` possède un certain nombre d'états `E1 E2 ... Ei` et en fonction de l'état où elle se trouve et de ce qu'elle lit sur le ruban elle passe d'un état à un autre, écrit sur le ruban et déplace sa tête de lecture-écriture.
Décrire une machine de Turing, c'est décrire précisément comment l'état de la machine évolue et quels sont les déplacements que la tête de lecture-écriture effectue.

Une instruction élémentaire de machine de Turing est par exemple :

```
"si je suis dans l'état E1 et que je lis le symbole 0 sur le ruban,
alors je dois passer dans l'état E2,
écrire 1, puis me déplacer d'une case vers la droite".
```

(E1 0 -> E2 1 D)

Définir la liste des instructions élémentaires d'une machine de Turing détermine complètement son comportement pour tout jeu de données.
La programmation des machines de Turing est difficile et peu agréable, l'intérêt de la notion tient nous l'avons dit, à la très grande simplicité des machines obtenues, pas à leur convivialité !

En général on précise quel est l'état de départ dans lequel la machine doit être placée avant un calcul, pour nous ce sera `E1`.
On peut aussi exiger et nous le ferons, d'un programme de machine de Turing qu'il ne soit pas ambigu, c'est-à-dire qu'il n'y ait jamais deux instructions élémentaires pouvant s'appliquer en même temps, comme par exemple :

(E1 0 -> E2 1 D) et (E1 0 -> E3 1 G)

Quand une machine de Turing est précisée et qu'un ruban lui est présenté, le déroulement de ses calculs est rigoureusement déterminé et unique.
On l'obtient en suivant scrupuleusement les instructions élémentaires de son programme tant qu'il y en a d'applicables.

(E1 0 -> E1 0 D) (E1 1 -> E2 0 D)

Cette machine de Turing `M` à chaque fois qu'on la placera sur un ruban ne comportant que des `0` et des `1` aura pour effet de se déplacer vers la droite jusqu'à ce qu'elle trouve un `1`, qu'elle transformera en 0 et s'arrêtera.

(E1 0 -> E1 1 D) (E1 1 -> E1 0 D)

transforme tous les `1` en `0` et tous les `0` en `1`, en se déplaçant vers la droite.

L'importance du concept de machines de Turing tient dans le fait que le type de calculs qu'elles permettent de faire est absolument général.
Toutes les tentatives de définition d'algorithme (souvent plus compliquées et en apparence beaucoup plus riches) en fait conduisent à une définition exactement équivalente en puissance.

Ces démonstrations d'équivalence sont souvent longues et pénibles, mais elles sont relativement faciles pour un spécialiste, et comme depuis maintenant cinquante ans qu'on en fait, on a toujours réussi à ramener toutes les propositions de définitions d'algorithmes à celle obtenue avec les machines de Turing, personne ne doute que le concept vague d'algorithme est bien cerné par la notion précise de machine de Turing, et, plus important encore que ce concept est unique : **il y a une notion et une seule d'algorithme**.

C'est ce qu'on appelle :

Thèse de Church (ou Church-Turing)
faisable par algorithme = faisable par machine de Turing

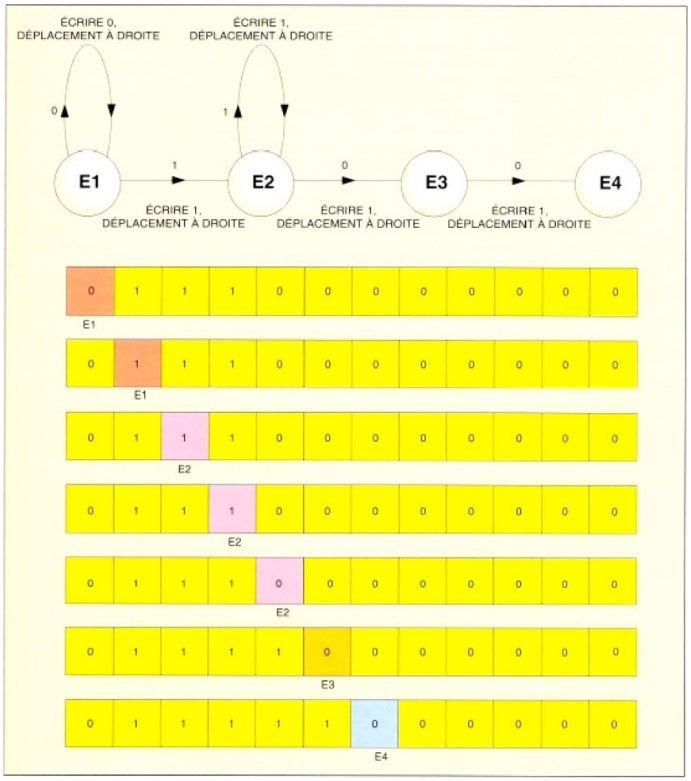
2. *Détail du programme et du fonctionnement de la machine de Turing calculant la fonction n -> n + 2*

Au départ le nombre `n` est écrit sur le ruban codé de la manière la plus simple possible par `n fois 1`, le reste du ruban étant recouvert de `zéro`.
Le résultat se trouvera sur le ruban à la fin du calcul (c'est-à-dire quand aucune instruction ne pourra plus s'appliquer) il sera codé de la même façon, avec des `1`.

```
(E1 0 -> E1 0 D) (E1 1 -> E2 1 D)
(E2 1 -> E2 1 D) (E2 0 -> E3 1 D)
(E3 0 -> E4 1 D)
```

Au départ la machine de Turing se trouve dans l'état `E1`.
Dans un premier temps la tête de lecture-écriture de la machine de Turing recherche un `1`.
Dès qu'elle l'a trouvé passe de l'état `E1` à l'état `E2`.
Alors elle se déplace vers la droite en recopiant chaque `1` qu'elle trouve, et en restant dans l'état `E2`, cela jusqu'à ce qu'elle rencontre un `0`.
Elle passe alors dans l'état `E3`, et remplace le `0` par un `1`.
Puis elle passe dans l'état `E4` en remplaçant encore le `0` qu'elle lit par un `1`.
Une fois dans l'état `E4`, elle s'arrête car aucune instruction ne peut plus la faire changer d'état.

Écrivez l'état du ruban pour un nombre quelconque entre 1 et 10.
Puis suivez les instructions de la machine de Turing en partant d'un zéro à gauche de votre nombre.
Le résultat final vaut-il votre nombre + 2 ?
Comparer avec l'image suivante.



LA MACHINE DE TURING qui calcule la fonction $f(n) = n + 2$ peut être représentée par un graphe *(en haut)* qui résume la liste des instructions. Chaque instruction, déterminée par un état et par une valeur lue sur la bande, est représentée par une flèche joignant l'état de départ à l'état d'arrivée, avec des indications d'écriture et de déplacement. En bas, on a indiqué le détail des états successifs de la machine et de son ruban pour la donnée initiale $n = 3$. Partie de l'état E_1, la machine passe dans l'état E_2 dès qu'elle rencontre un 1. Puis elle parcourt les n cases portant des 1, en restant dans l'état E_2 et, dès qu'elle trouve un 0, elle le remplace par un 1, passe dans l'état E_3, remplace encore le 0 suivant par un 1, passe dans l'état E_4 et s'arrête. Le bilan de ce travail est que deux 1 supplémentaires ont été ajoutés. Les n symboles 1 sont devenus $n + 2$ symboles 1.

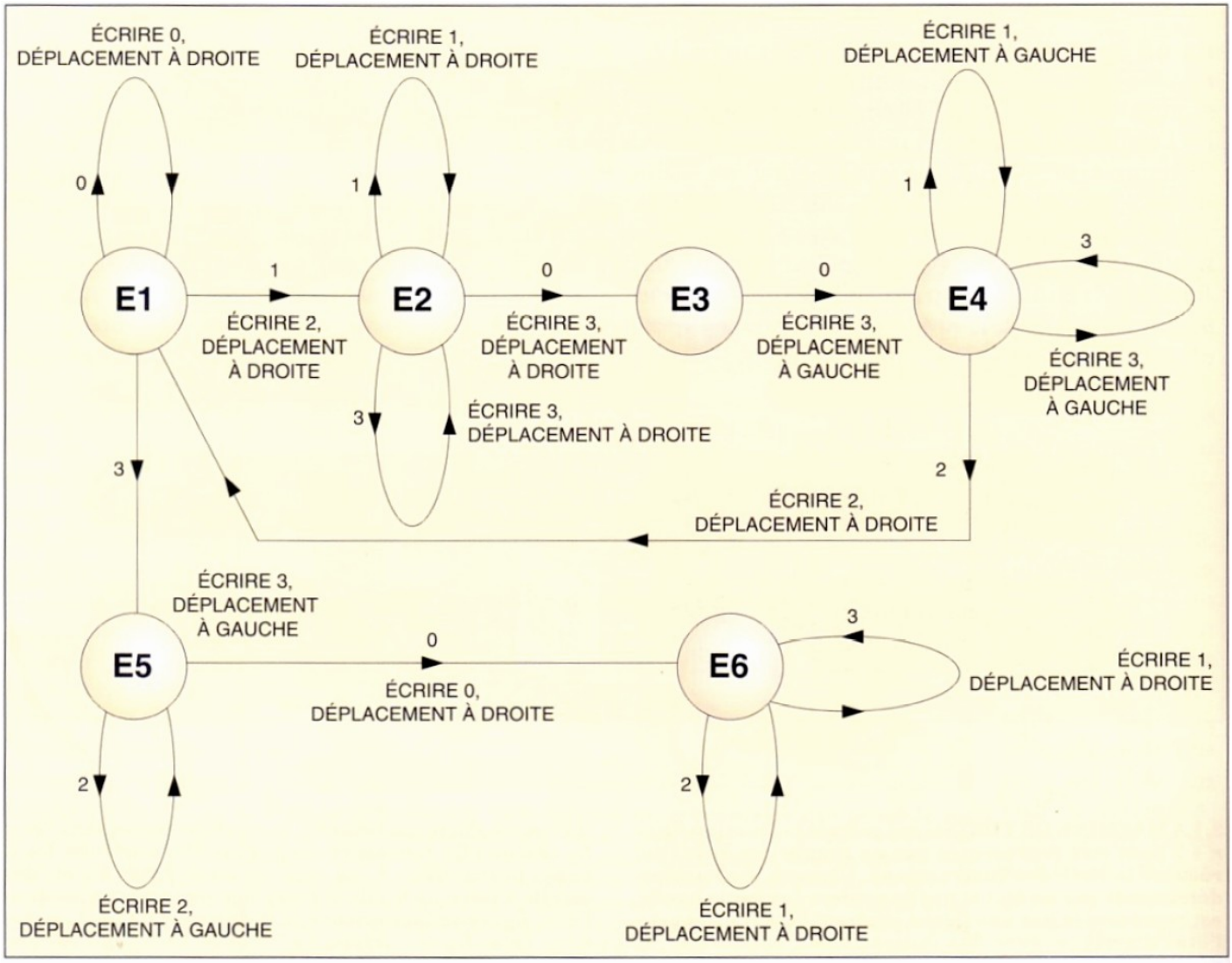
3. Machine de Turing calculant la fonction n -> 3n

```
(E1 0 -> E1 0 D)  (E1 1 -> E2 2 D)
(E1 3 -> E5 3 G)  (E2 1 -> E2 1 D)
(E2 3 -> E2 3 D)  (E2 0 -> E3 3 D)
(E3 0 -> E4 3 G)  (E4 3 -> E4 3 G)
(E4 1 -> E4 1 G)  (E4 2 -> E1 2 D)
(E5 2 -> E5 2 G)  (E5 0 -> E6 0 D)
(E6 2 -> E6 1 D)  (E6 3 -> E6 1 D)
```

Au départ le ruban ne comporte que des `0` et `n` fois le `1`.
Le calcul se déroule en plusieurs étapes.
D'abord la machine `M` repère le premier `1` qu'elle trouve et le transforme en `2` (*).
Ensuite elle se déplace jusqu'à rencontrer un `0`, qu'elle transforme en `3` ainsi que le zéro suivant.
Elle revient alors en arrière à la recherche d'un `2` (***).
Dès qu'elle l'a trouvé elle repart en avant dans l'état `E1` à la recherche d'un `1` ou d'un `3` (****).
Si elle trouve un `1` ($) elle va à nouveau aller ajouter deux `3` à droite de ceux déjà écrits ($$), si elle trouve un `3` ($$$), ce qui signifie que tous les `1` présents au départ ont été transformés en `2`, elle termine en transformant tous les symboles non nuls en `1` et s'arrête ($$$$).
Dans ce calcul chaque `1` a donné naissance à deux autres `1`, il y a donc a la fin du calcul exactement `3` fois plus de `1` qu'il y en avait au début.

Écrivez l'état du ruban pour le nombre 2.
Puis suivez les instructions de la machine de Turing en partant d'un zéro à gauche de votre nombre.
Schématisez les états de la machine de Turing comme dans l'image précédente.
Comparer avec les documents suivants.

```
0 1 1 0 0 0 0 0 0 0 ... (*)
E1
0 1 1 0 0 0 0 0 0 0 ... (*)
  E1
0 2 1 0 0 0 0 0 0 0 ... (**)
    E2
0 2 1 0 0 0 0 0 0 0 ... (**)
      E2
0 2 1 3 0 0 0 0 0 0 ... (**)
        E3
0 2 1 3 3 0 0 0 0 0 ... (**)
      E4
0 2 1 3 3 0 0 0 0 0 ... (***)
    E4
0 2 1 3 3 0 0 0 0 0 ... (***)
  E4
0 2 1 3 3 0 0 0 0 0 ... (****)
    E1
0 2 2 3 3 0 0 0 0 0 ... ($)
      E2
0 2 2 3 3 0 0 0 0 0 ... ($$)
        E2
0 2 2 3 3 0 0 0 0 0 ... ($$)
          E2
0 2 2 3 3 3 0 0 0 0 ... ($$)
            E3
0 2 2 3 3 3 3 0 0 0 ... ($$)
          E4
0 2 2 3 3 3 3 0 0 0 ... ($$)
        E4
0 2 2 3 3 3 3 0 0 0 ... ($$)
      E4
0 2 2 3 3 3 3 0 0 0 ... ($$)
    E4
0 2 2 3 3 3 3 0 0 0 ... ($$$)
      E1
0 2 2 3 3 3 3 0 0 0 ... ($$$$)
    E5
0 2 2 3 3 3 3 0 0 0 ... ($$$$)
  E5
0 2 2 3 3 3 3 0 0 0 ... ($$$$)
E5
0 2 2 3 3 3 3 0 0 0 ... ($$$$)
  E6
0 1 2 3 3 3 3 0 0 0 ... ($$$$)
    E6
0 1 1 3 3 3 3 0 0 0 ... ($$$$)
      E6
0 1 1 1 3 3 3 0 0 0 ... ($$$$)
        E6
0 1 1 1 1 3 3 0 0 0 ... ($$$$)
          E6
0 1 1 1 1 1 3 0 0 0 ... ($$$$)
            E6
0 1 1 1 1 1 1 0 0 0 ... ($$$$)
              E6
arrêt.
```



MACHINE DE TURING qui calcule la fonction $f(n) = 3n$.

4. Machine de Turing calculant la fonction n -> 3n + 2

Écrivez et testez la Machine de Turing correspondante.