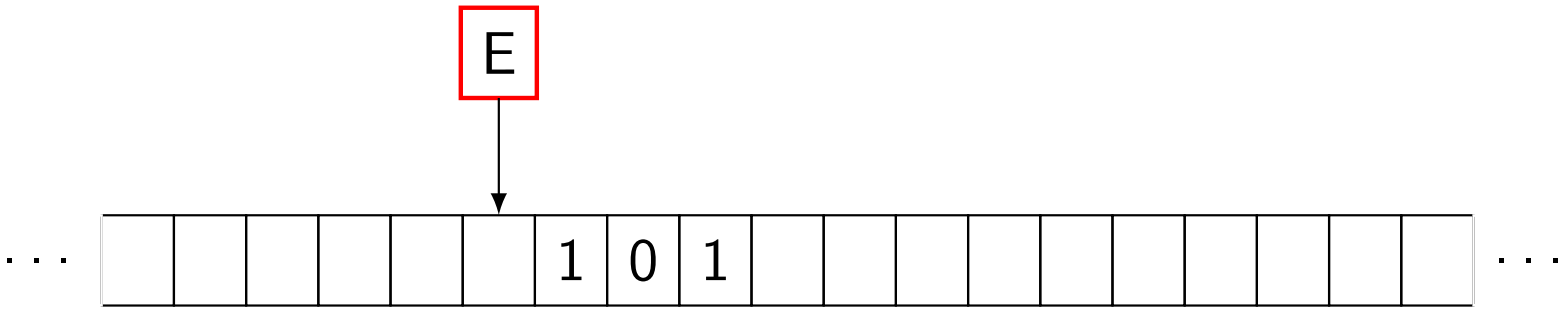
Alan Turing (1912 – 1954)

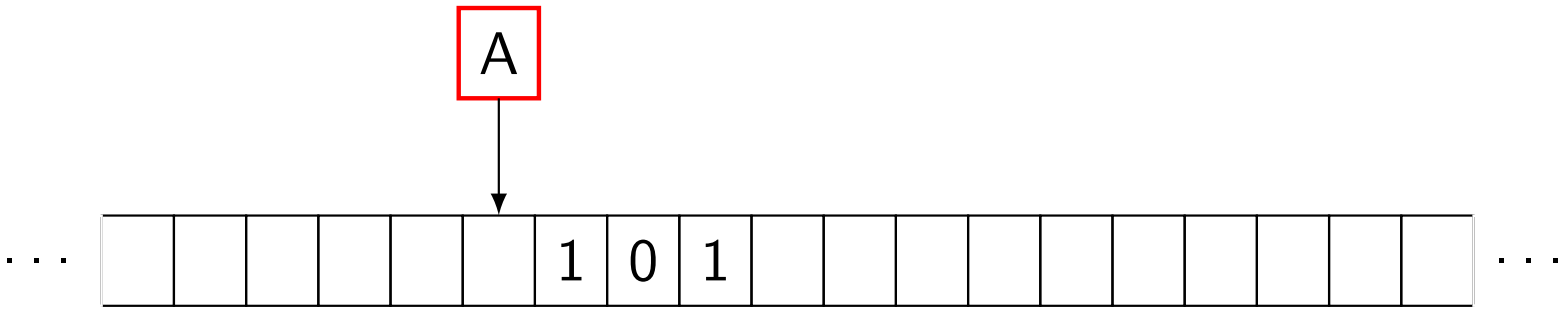
Une machine de Turing Déterministe

- Un ruban infini
- Une tête de lecture et d'écriture qui se trouve dans un état E.;
- Un alphabet R contenant le caractére blanc (ø);
- On peut déplacer le ruban ou la tête de lecture vers la droite ou la gauche;
- Une fonction de transition qui à partir de l'état et de la lecture va écrire, passer ou non à un autre état et avancer à droite ou à gauche le ruban (ou la tête de lecture).



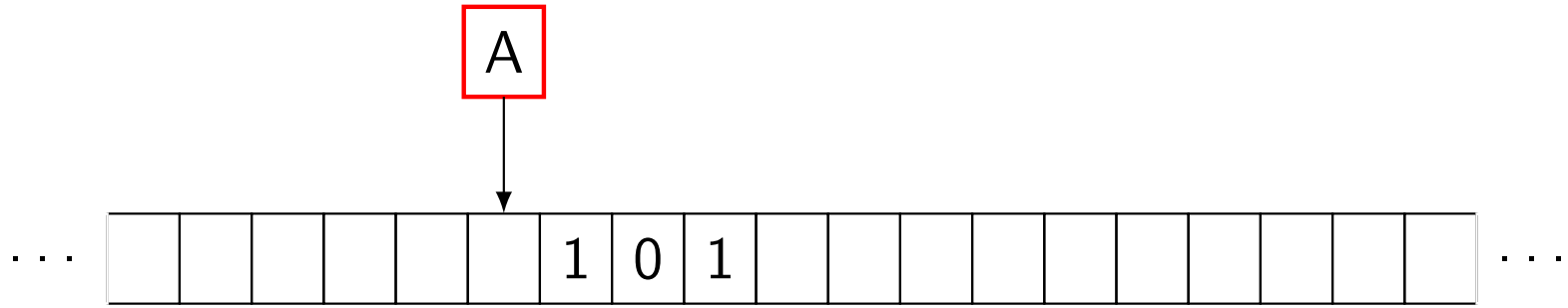
Une machine de Turing Déterministe : Un exemple

- Un ruban infini avec 0 et 1;
- Trois états A (initial), B et C (finale).



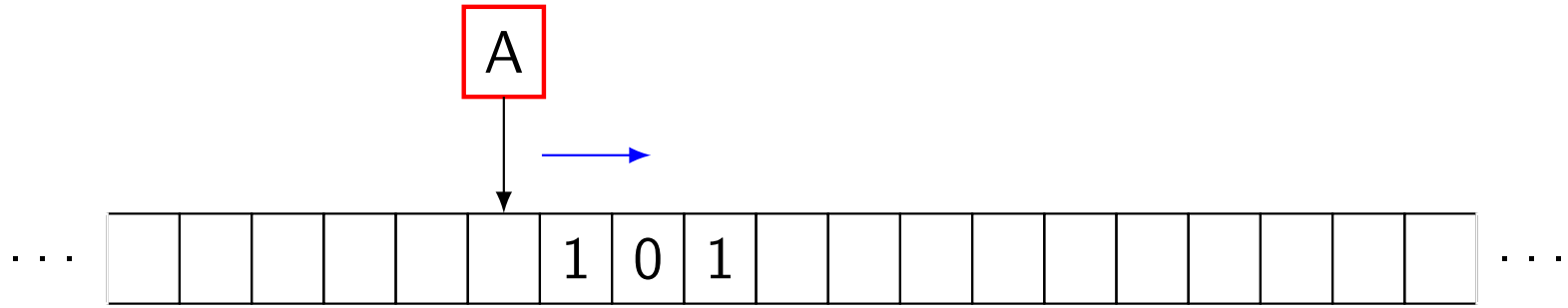
Une machine de Turing Déterministe : Un exemple

Etat	Lecture	Ecriture	Deplacement	Etat suivant
A	blanc	blanc	droite	B
B	0	0	droite	B
	1	1	droite	B
	blanc	0	droite	C (Fin)



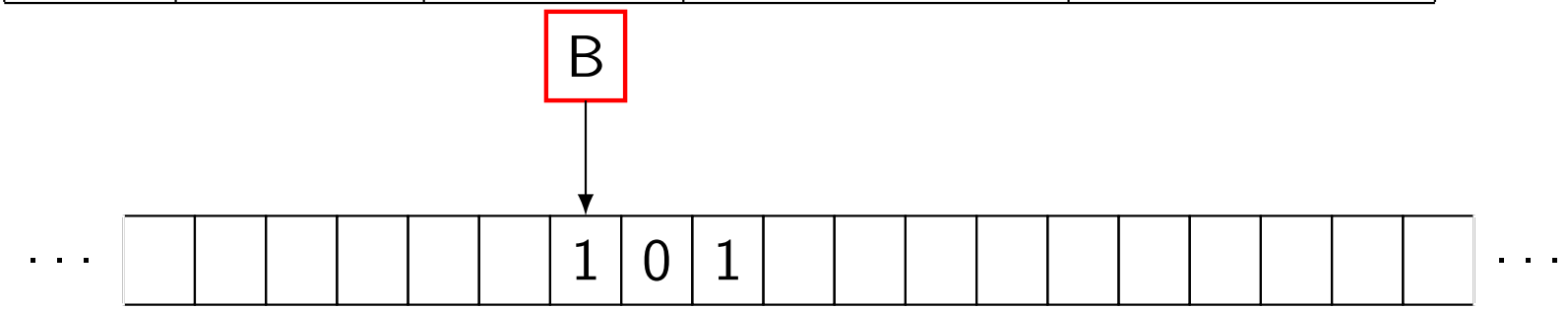
Une machine de Turing Déterministe : Un exemple

Etat	Lecture	Ecriture	Deplacement	Etat suivant
A	blanc	blanc	droite	B
B	0	0	droite	B
	1	1	droite	B
	blanc	0	droite	C (Fin)



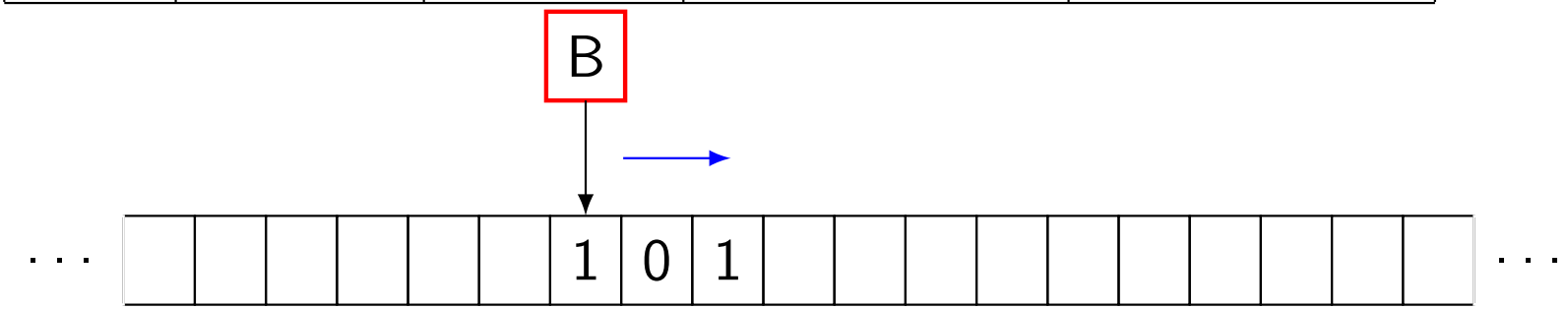
Une machine de Turing Déterministe : Un exemple

Etat	Lecture	Ecriture	Deplacement	Etat suivant
A	blanc	blanc	droite	B
B	0	0	droite	B
	1	1	droite	B
	blanc	0	droite	C (Fin)



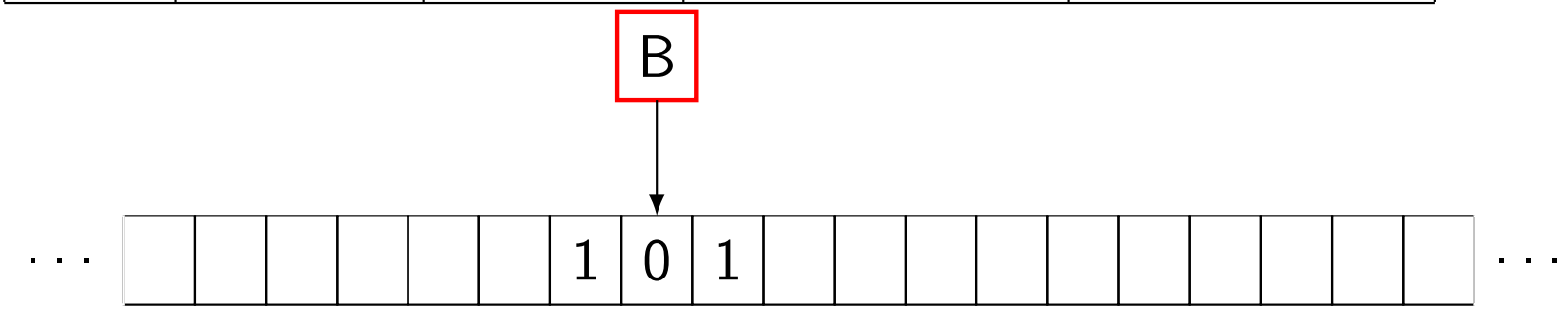
Une machine de Turing Déterministe : Un exemple

Etat	Lecture	Ecriture	Deplacement	Etat suivant
A	blanc	blanc	droite	B
B	0	0	droite	B
	1	1	droite	B
	blanc	0	droite	C (Fin)



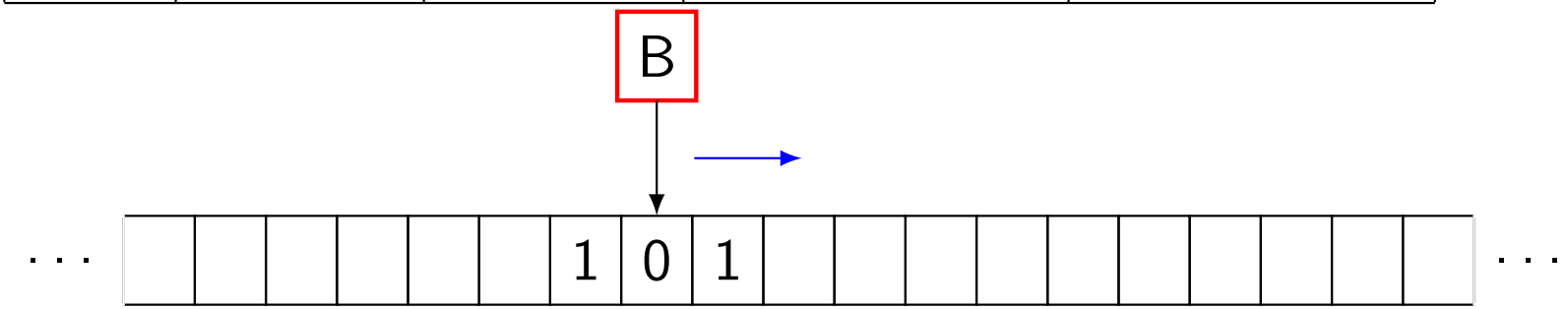
Une machine de Turing Déterministe : Un exemple

Etat	Lecture	Ecriture	Deplacement	Etat suivant
A	blanc	blanc	droite	B
B	0	0	droite	B
	1	1	droite	B
	blanc	0	droite	C (Fin)



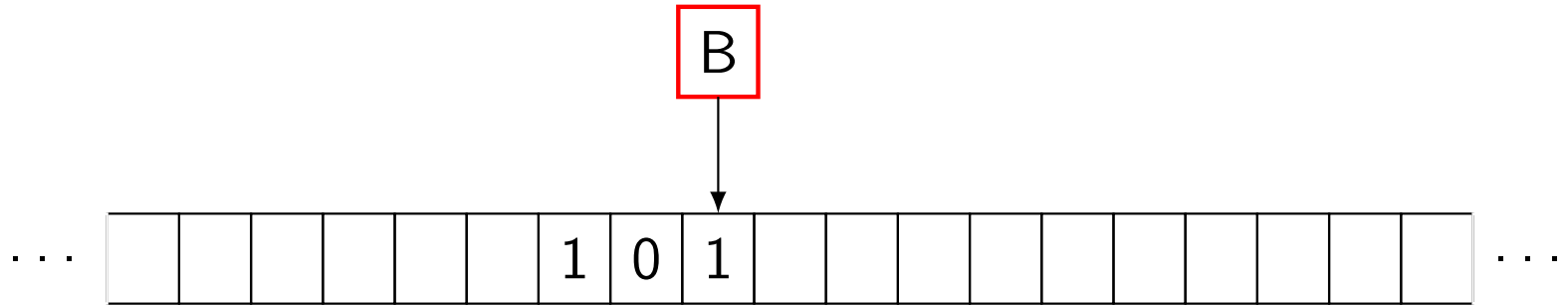
Une machine de Turing Déterministe : Un exemple

Etat	Lecture	Ecriture	Deplacement	Etat suivant
A	blanc	blanc	droite	B
B	0	0	droite	B
	1	1	droite	B
	blanc	0	droite	C (Fin)



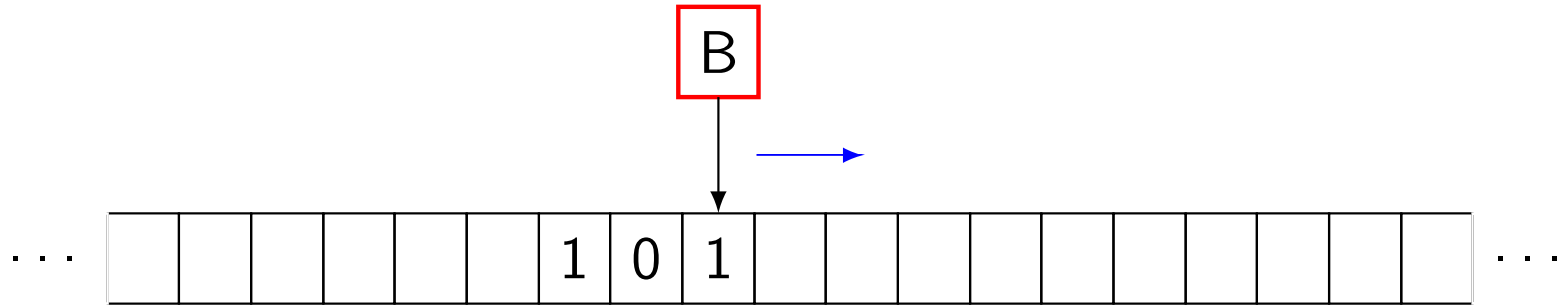
Une machine de Turing Déterministe : Un exemple

Etat	Lecture	Ecriture	Deplacement	Etat suivant
A	blanc	blanc	droite	B
B	0	0	droite	B
	1	1	droite	B
	blanc	0	droite	C (Fin)



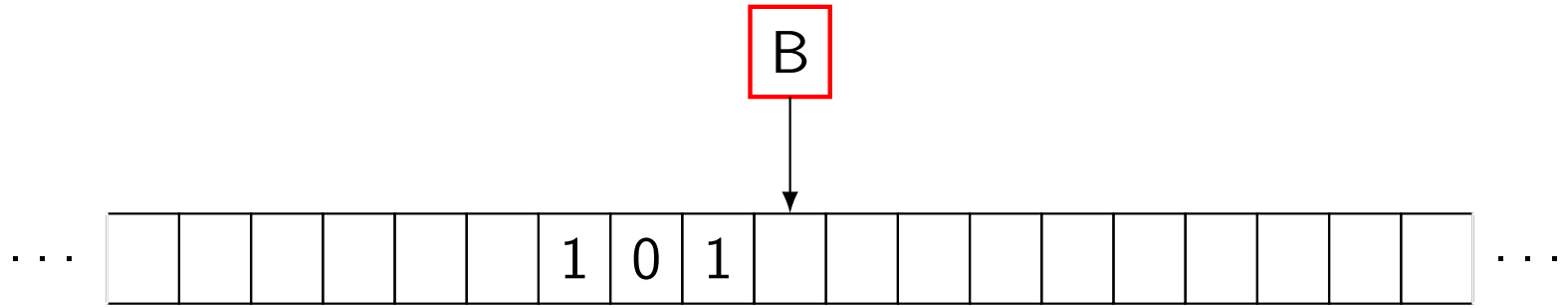
Une machine de Turing Déterministe : Un exemple

Etat	Lecture	Ecriture	Deplacement	Etat suivant
A	blanc	blanc	droite	B
B	0	0	droite	B
	1	1	droite	B
	blanc	0	droite	C (Fin)



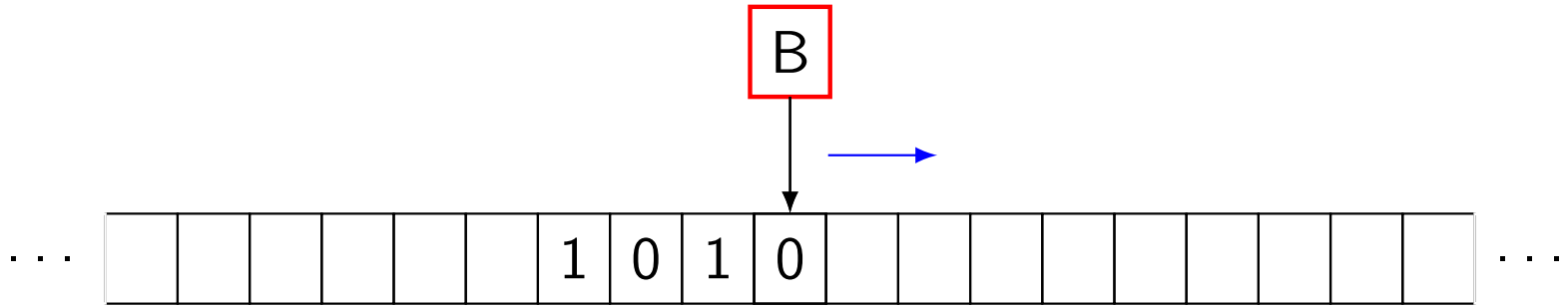
Une machine de Turing Déterministe : Un exemple

Etat	Lecture	Ecriture	Deplacement	Etat suivant
A	blanc	blanc	droite	B
B	0	0	droite	B
	1	1	droite	B
	blanc	0	droite	C (Fin)



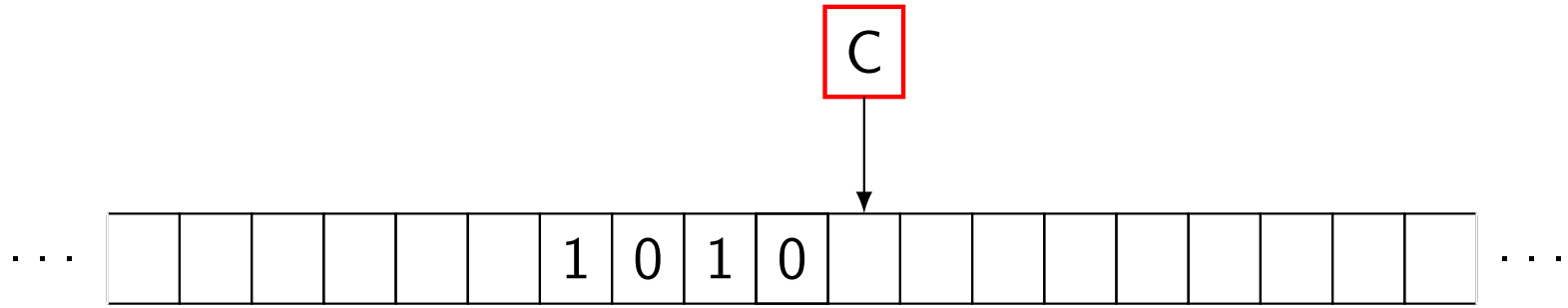
Une machine de Turing Déterministe : Un exemple

Etat	Lecture	Ecriture	Deplacement	Etat suivant
A	blanc	blanc	droite	B
B	0	0	droite	B
	1	1	droite	B
	blanc	0	droite	C (Fin)



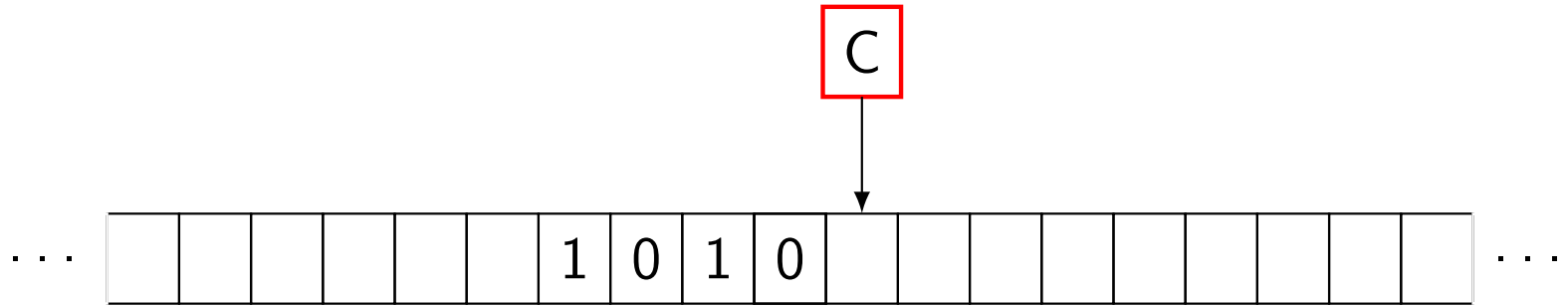
Une machine de Turing Déterministe : Un exemple

Etat	Lecture	Ecriture	Deplacement	Etat suivant
A	blanc	blanc	droite	B
B	0	0	droite	B
	1	1	droite	B
	blanc	0	droite	C (Fin)



Une machine de Turing Déterministe : Un exemple

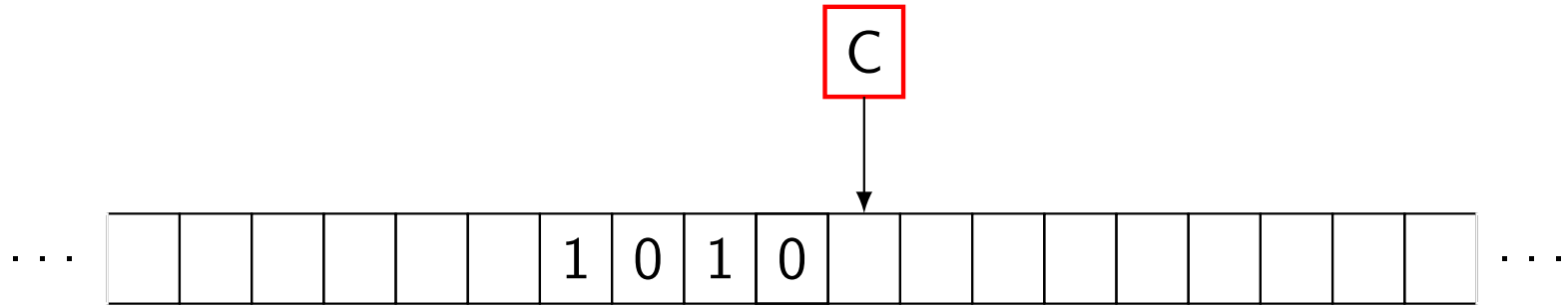
Etat	Lecture	Ecriture	Deplacement	Etat suivant
A	blanc	blanc	droite	B
B	0	0	droite	B
	1	1	droite	B
	blanc	0	droite	C (Fin)



Que fait cette machine ?

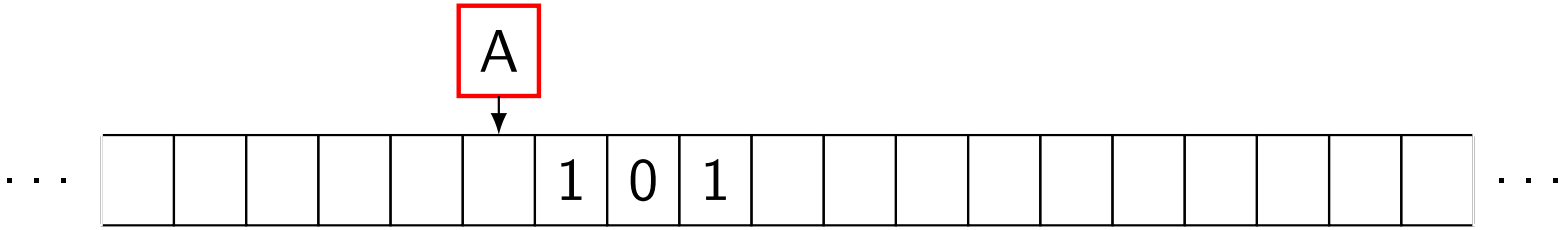
Une machine de Turing Déterministe : Un exemple

Etat	Lecture	Ecriture	Deplacement	Etat suivant
A	blanc	blanc	droite	B
B	0	0	droite	B
	1	1	droite	B
	blanc	0	droite	C (Fin)

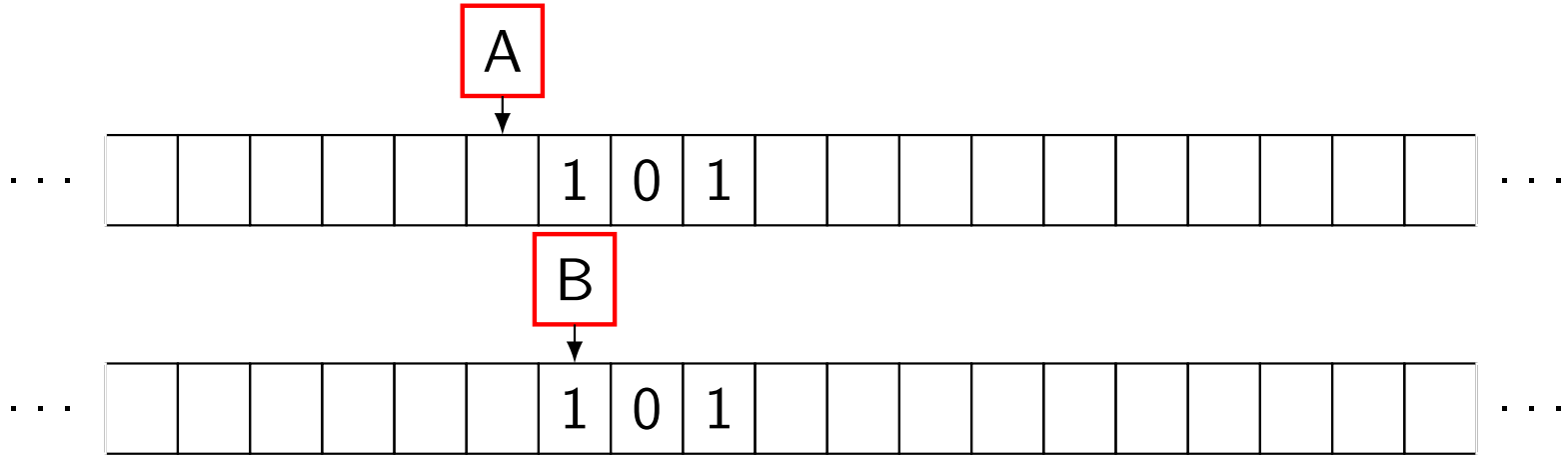


Que fait cette machine ? C'est la multiplication par 2.

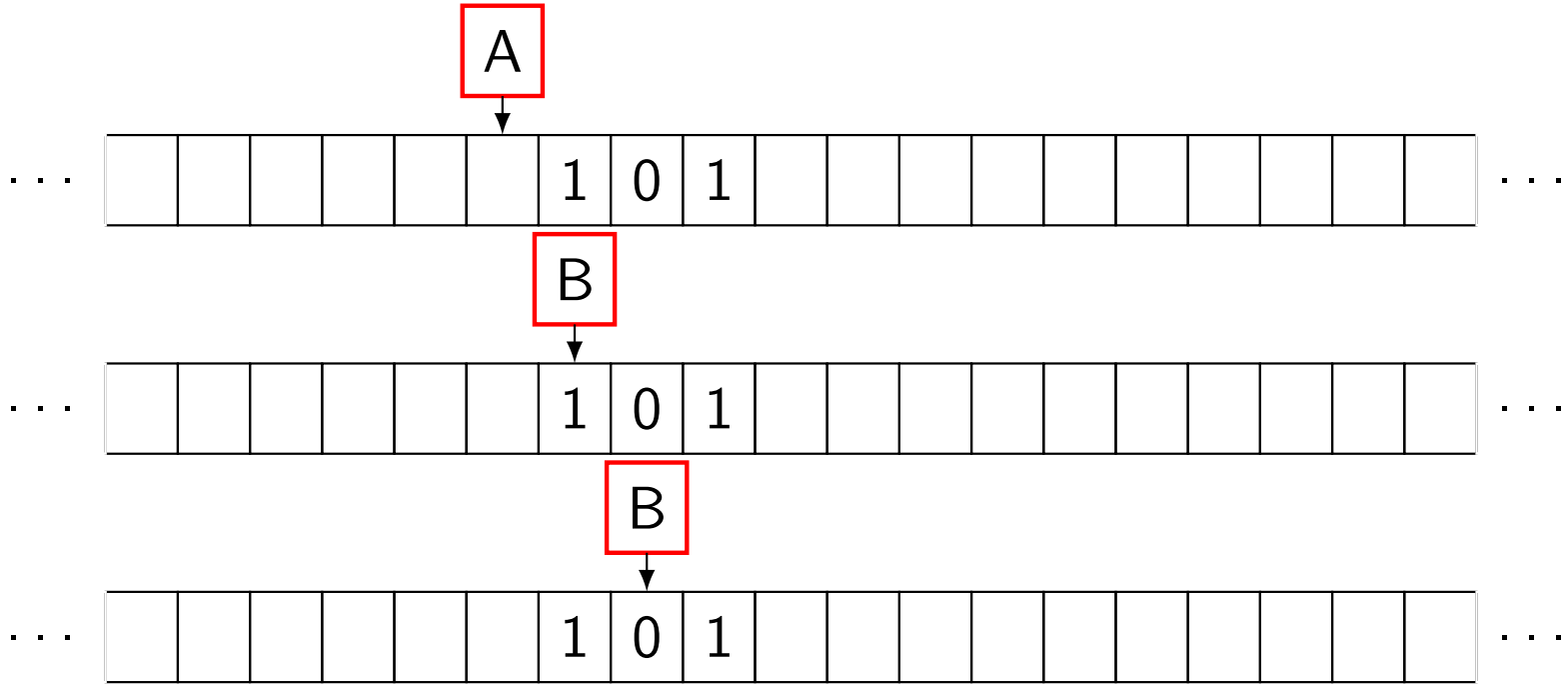
Une machine de Turing Déterministe : Un exemple



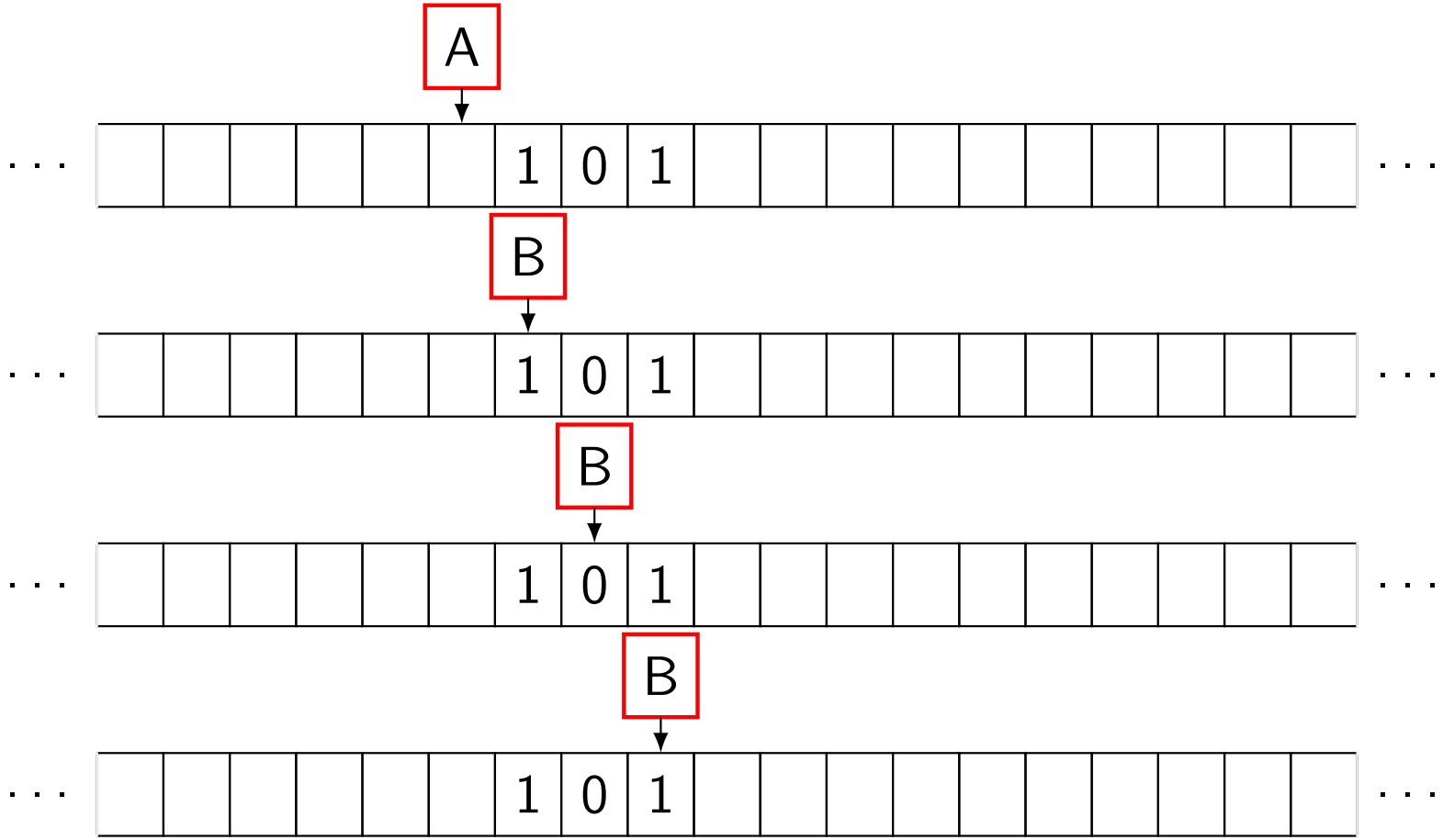
Une machine de Turing Déterministe : Un exemple



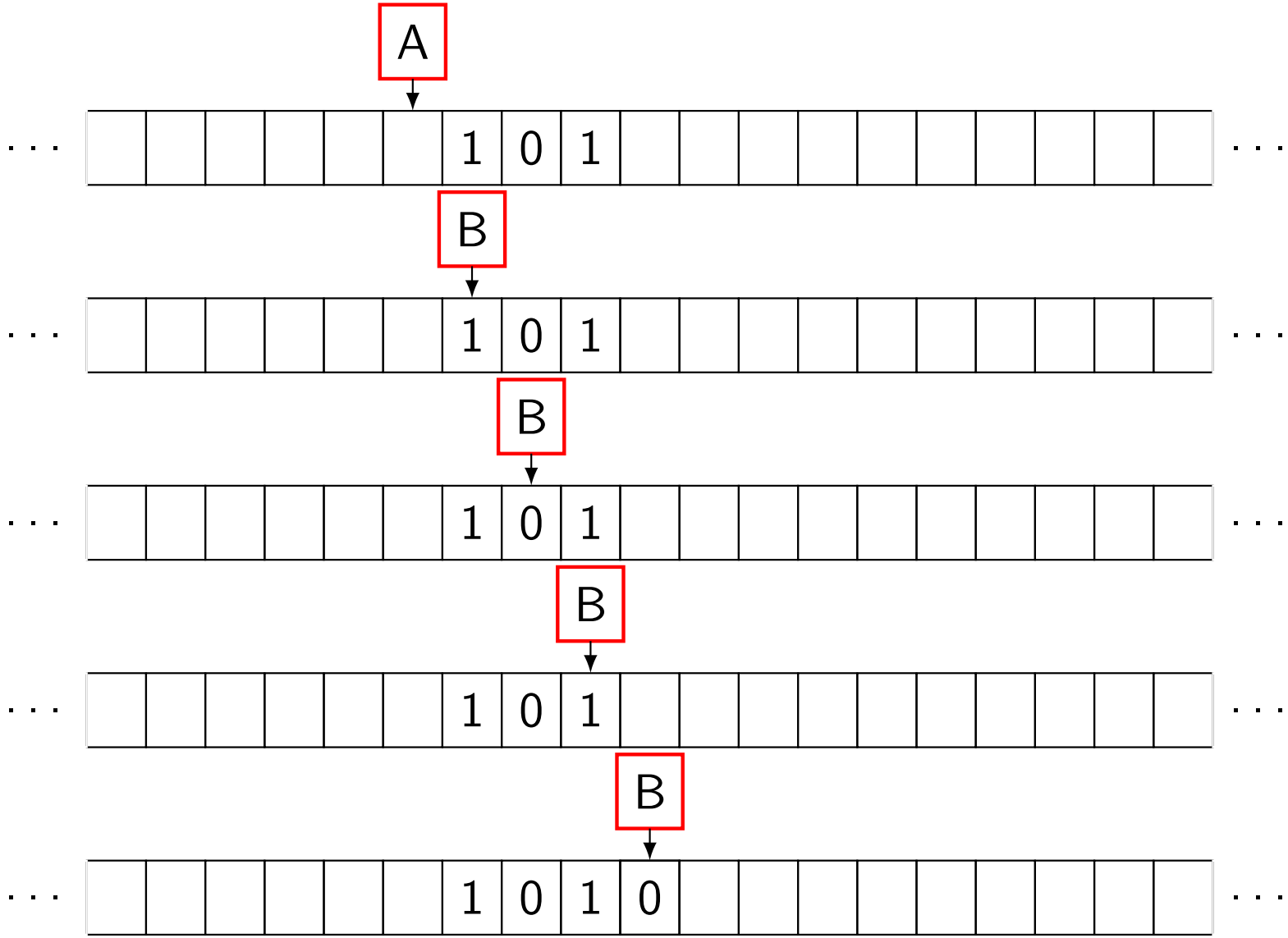
Une machine de Turing Déterministe : Un exemple



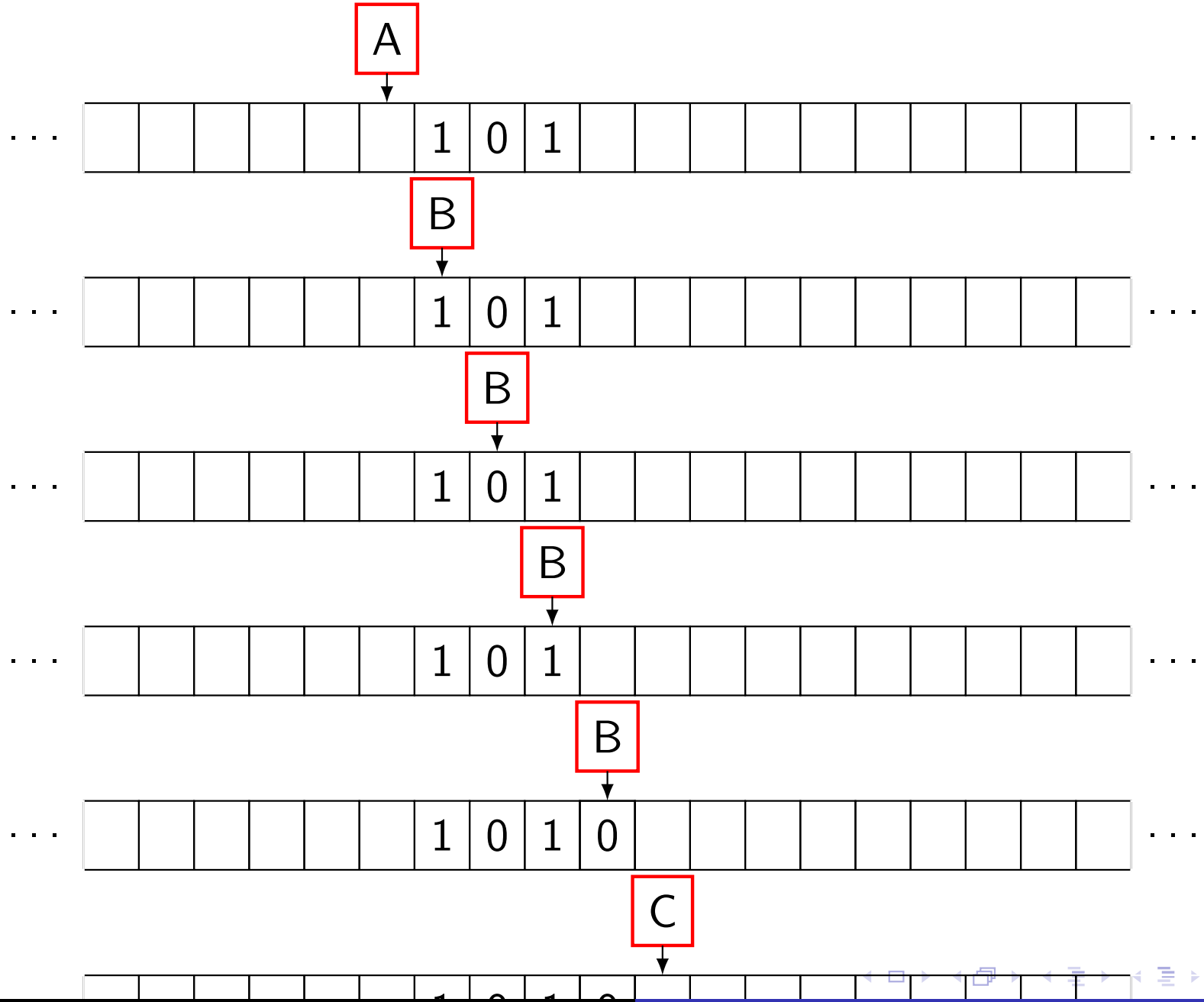
Une machine de Turing Déterministe : Un exemple



Une machine de Turing Déterministe : Un exemple



Une machine de Turing Déterministe : Un exemple



Programmer en python: la table de transition

```
def transition(ruban,i,etat):
    if etat=="A":
        if ruban[i]==None:
            i=i+1; etat="B"
    elif etat=="B":
        if ruban[i]==0:
            i=i+1; etat="B"
        elif ruban[i]==1:
            i=i+1; etat="B"
        elif ruban[i]==None:
            ruban[i]=0
            i=i+1; etat="C"
    return ruban,i,etat
```

Programmer en python: la machine

```
def machine(ruban,i,etat):
    """
    In: Liste de 0, 1 ou None (le ruban de Turing),
        int (position de la tête de lecture,
        str (état initial)
    Out: Liste de 0, 1 ou None (le ruban de Turing),
    """
    while etat!="C":
        ruban,i,etat=transition(ruban,i,etat)
    return ruban

monruban=[None,None,1,0,1,None,None]
depart=1
etat="A"
print(machine(monruban,depart,etat))
```

Avec la définition d'une machine de Turing, on peut maintenant définir ce qu'est un algorithme.

Definition 1 (algorithme)

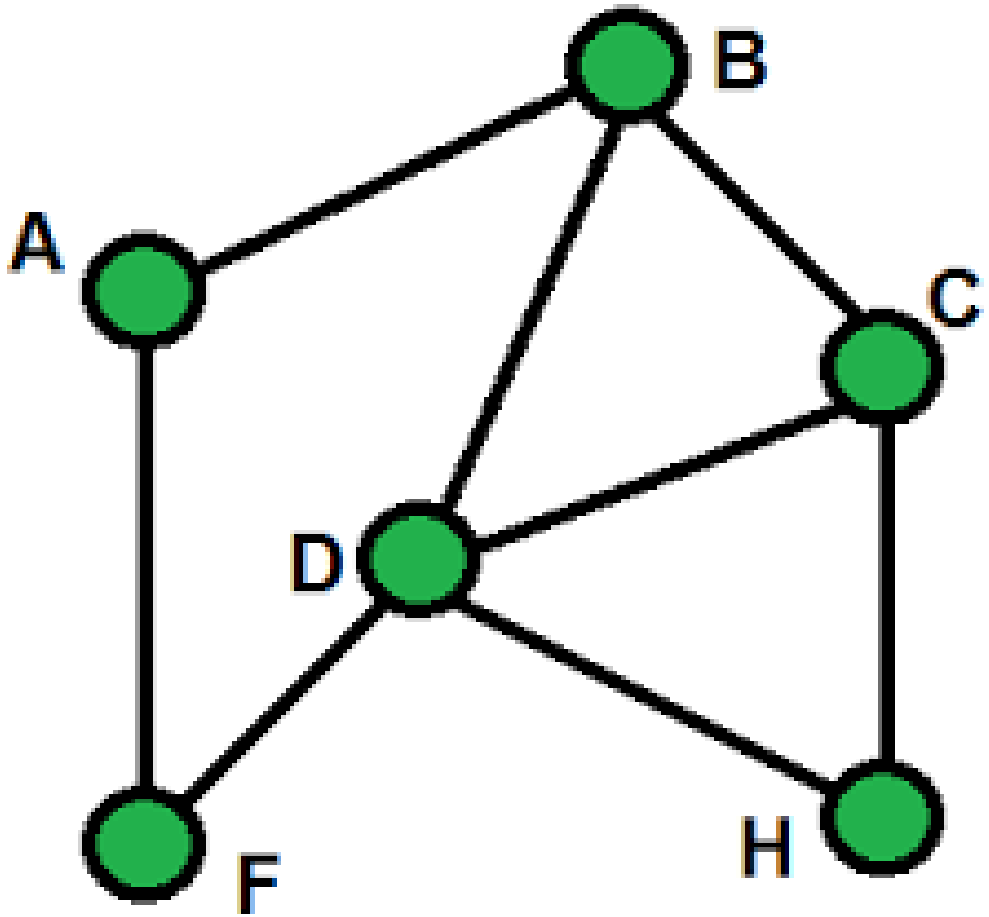
Un algorithme est une table de transition d'une machine de Turing.

Definition 2 (calculabilité)

On dit qu'un problème est calculable s'il existe une machine de Turing déterministe qui le résout.

Exemple : graphe et chemin hamiltonien

Un chemin hamiltonien d'un graphe orienté ou non orienté est un chemin qui passe par tous les sommets une fois et une seul.



Existe-t-il un chemin hamiltonien et si oui lequel ?

Machine de Turing universelle

Qu'est ce qu'un ordinateur moderne ?

Definition 3

Une machine de Turing Universelle est une machine de Turing qui peut simuler n'importe quelle machine de Turing.

- Une machine de Turing est définie par sa table de transition
- La machine de Turing universelle est seulement une machine de Turing qui a comme paramètre une table de transition.

Tous les ordinateurs actuelles sont basées sur le principe de la machine de Turing universelle.

Machine de Turing universelle

On peut simuler cette machine en python de la maniére suivante:

```
def machineUniverselle(MaTuring,ruban,i,etat):
    """
    In: Une fonction de transition
        Liste de 0, 1 ou None (le ruban de Turing),
        int (position de la t^ete de lecture,
        str ( etat initial)
    Out: Liste de 0, 1 ou None (le ruban de Turing)
    """
    while etat!="C":
        ruban,i,etat=MaTuring(ruban,i,etat)
    return ruban
```

Definition 4 (Problème de décision)

Un problème de décision est un problème fermé. C'est à dire une question à laquelle on répond par oui ou par non.

Exemples

- Soit n un entier, est il premier ?
- Soit P un polynôme , $P(x) = 0$ a-t-il une solution dans $\mathbb{R}$?
- Un voyageur se déplace sur Terre avec, à chaque étape de son voyage, l'indication seule de sa prochaine destination.
  " le voyageur atteindra-t-il Rome en cinq étapes ? "

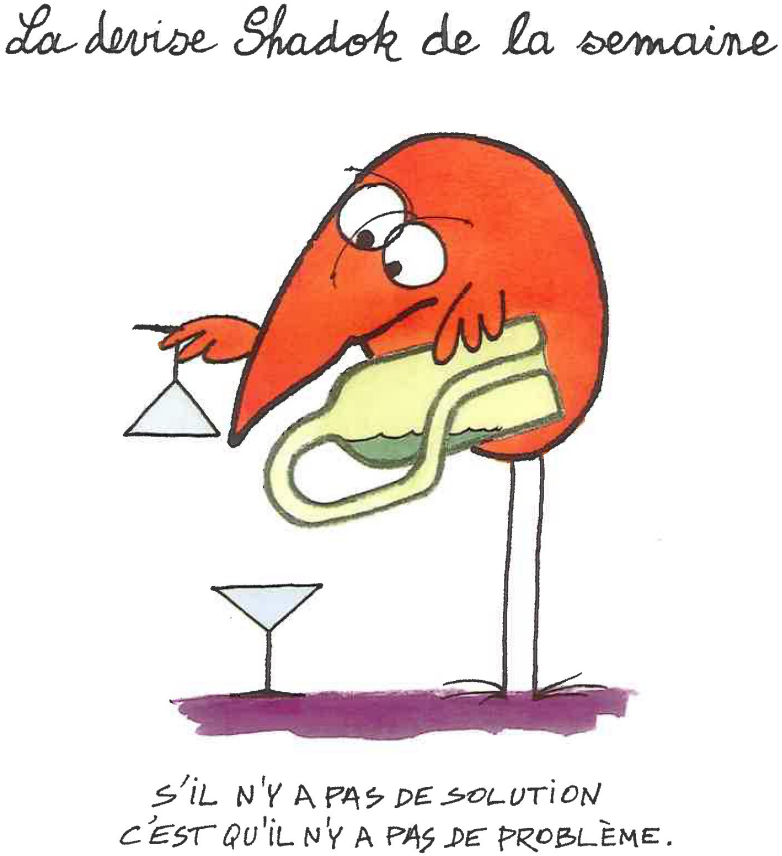
Décidabilité

Definition 5 (Décidabilité)

Un problème de décision est **décidable** s'il existe un algorithme qui répond à la question du problème en un nombre fini d'étapes sinon il est **indécidable**.

Décidabilité

Pendant très longtemps les mathématiciens ont considérés (intuitivement) que tout problème avait une solution, et qu'il suffisait de trouver la méthode qui permettrait de résoudre le problème.



Puis la question s'est naturellement posée notamment lors de la publication des 23 problèmes (1900) de David Hilbert.
Certains d'entre eux ne sont toujours pas résolus comme **la conjecture de Goldbach** qui affirme que

Conjecture de Goldbach

Tout nombre entier pair supérieur à 3 peut s'écrire comme la somme de deux nombres premiers ($12 = 5 + 7$).

Décidabilité

Depuis les travaux de Kurt Gödel (1931) et d'Alan Turing (1936), on sait maintenant qu'il existe une infinité de problèmes non décidables (de fonctions non calculables).
Ces travaux ont conduit à déterminer les contours de ce qui est faisable ce qui a permis à l'informatique de naître.

Décidabilité et Calculabilité

La notion de décidabilité et celle de calculabilité sont semblables.

- Si le problème P est décidable, alors il existe un algorithme qui le résout. La fonction P est donc calculable car il existe une machine de Turing déterministe qui le résout.
- Réciproquement, si la fonction P est calculable, il existe un algorithme qui en calcule chaque image. Donc pour chaque valeur a, un algorithme permet de savoir si P(a) est vraie ou fausse. Le problème P est décidable.

Problème de l'arrêt

Maintenant, une question reste : Toutes les fonctions sont-elles calculables ?
La réponse fut tranchée dans les années 1930. Toutes les fonctions ne sont pas calculables.
Le problème de l'arrêt en est un exemple.

Le problème de l'arrêt

Déterminer si un algorithme s'arrête.

Problème de l'arrêt

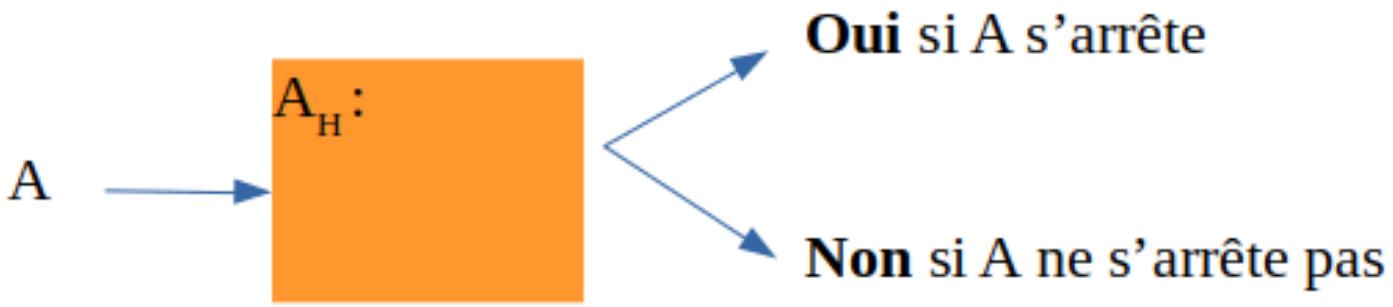
Le problème de l'arrêt

Le problème de l'arrêt est indécidable. Cela se démontre par l'absurde.

On suppose que ce problème est décidable. On montre ensuite qu'il y a une incohérence.

Preuve d'Alan Turing avec le problème dit de l'arrêt.

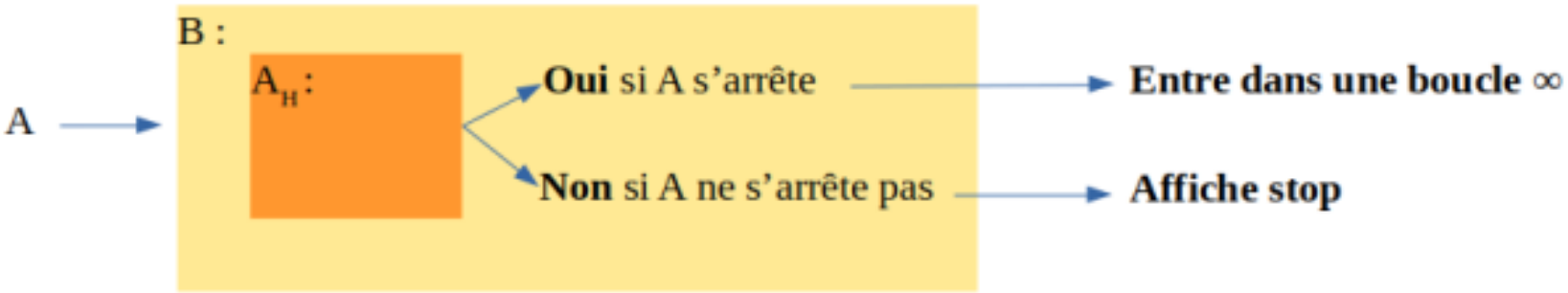
Supposons l'existence d'un algorithme A_H qui permet de savoir si un algorithme A s'arrête ou pas



Preuve du problème de l'arrêt

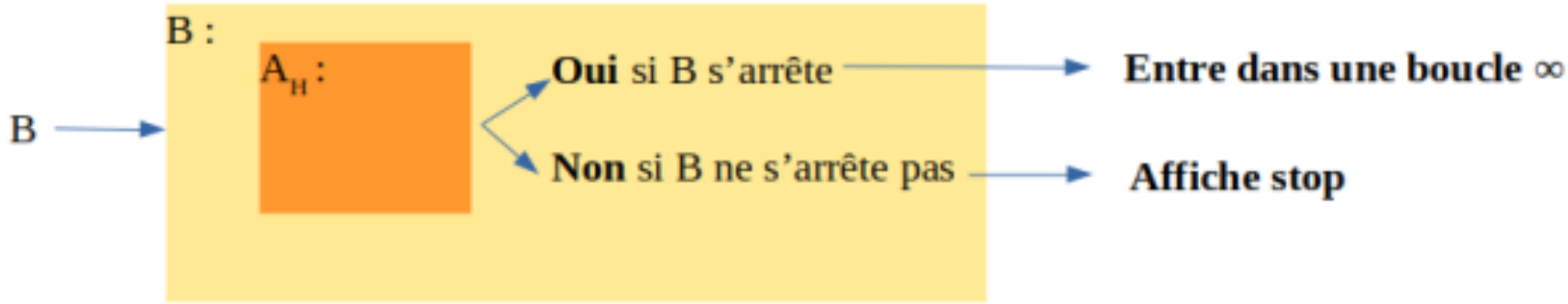
On considère alors l'algorithme B, qui n'existe que si A_H existe. B prend en entrée un algorithme A, puis exécute A_H sur A.

- Si A s'arrête alors B entre dans une boucle infinie
- Sinon B s'arrête en affichant par exemple "stop"



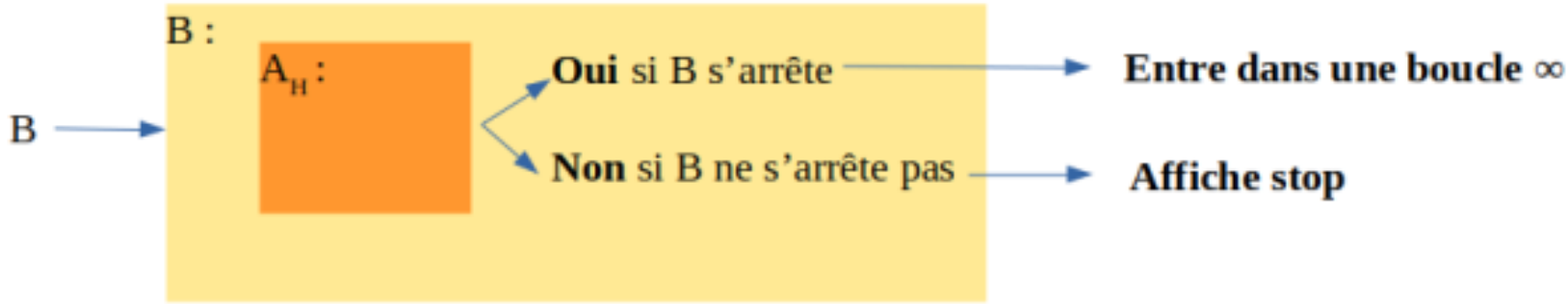
Preuve du problème de l'arrêt

B étant un algorithme on peut donc effectuer B(B)



Preuve du problème de l'arrêt

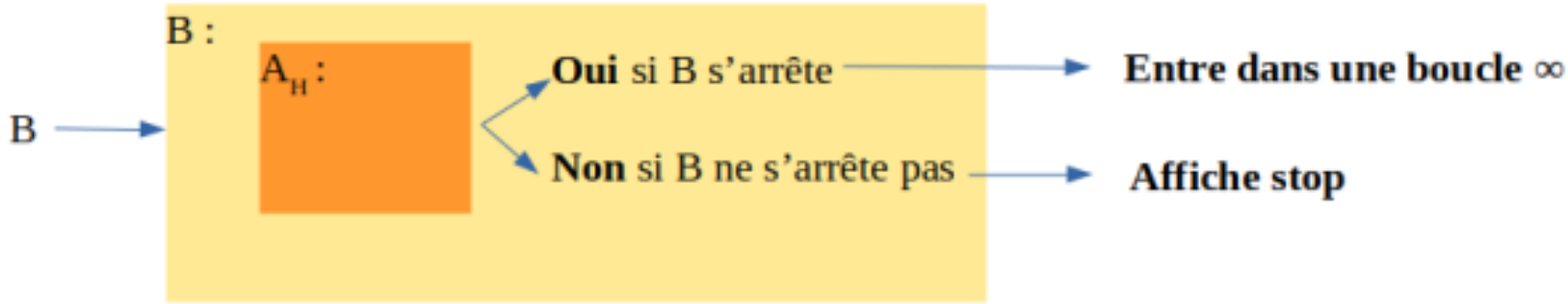
B étant un algorithme on peut donc effectuer B(B)



- Si B s'arrête alors B ne s'arrête pas;
- Si B ne s'arrête pas alors B s'arrête .

Preuve du problème de l'arrêt

B étant un algorithme on peut donc effectuer B(B)



- Si B s'arrête alors B ne s'arrête pas;
- Si B ne s'arrête pas alors B s'arrête .

Ces contradictions prouve que B n'existe pas et donc que A_H n'existe pas.