CALCULS IMPOSSIBLES

& CALCULS DIFFICILES

aka. A Gentle introduction to Calculability and Complexity

Rémy C.

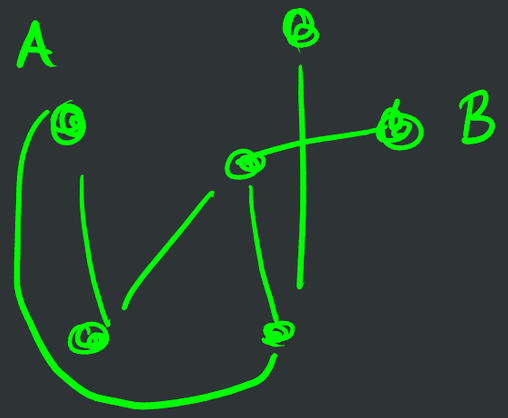
Séminaire des doctorant-es
I2M-CPT, Luminy, 12.05.22

Quels problèmes peuvent être résolus par un algorithme ?

Quels problèmes peuvent être résolus par un algorithme ?

Entrée → Sortie

Exemple : Entrée = Un graphe + deux sommets A et B

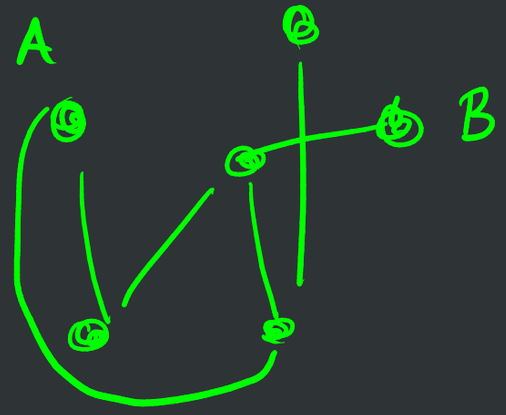


Sortie = Le plus court chemin A → B

Quels problèmes peuvent être résolus par un algorithme ?

Entrée $\longrightarrow$ {Vrai, Faux}

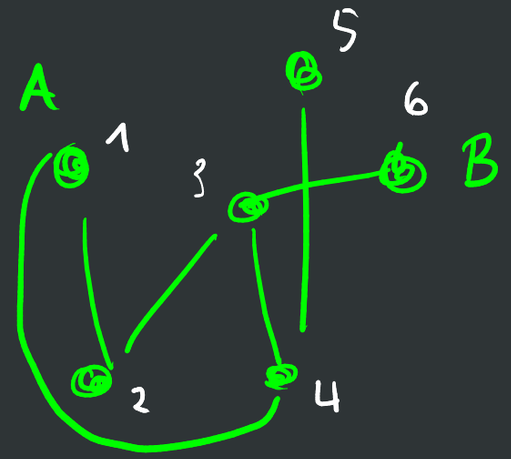
Exemple : Entrée = Un graphe
+
deux sommets
A et B
+
$k \in \mathbb{N}$



Sortie = Y a-t-il un chemin $A \to B$ de longueur $\leq k$?

Quels problèmes peuvent être résolus par un algorithme ?

Entrée $\longrightarrow$ {Vrai, Faux}



codage de l'entrée

$$\left(6,\begin{bmatrix}2,4\\1,3\\2,4,6\\1,3,5\\4\\3\end{bmatrix},1,6\right)$$

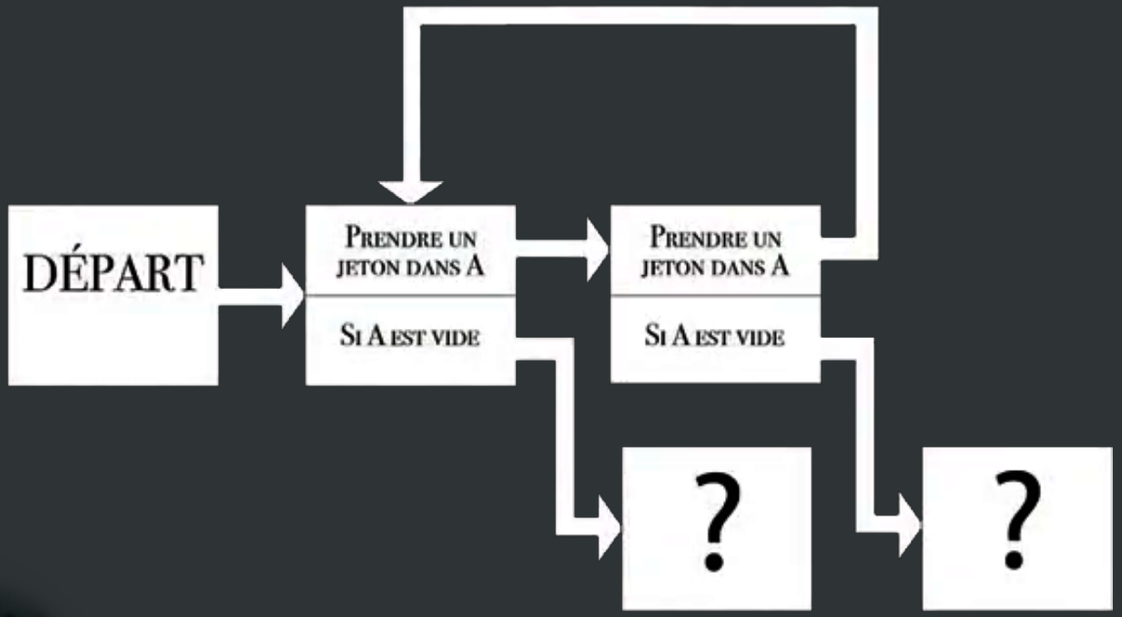
Quels problèmes peuvent être résolus par un algorithme ?

une procédure effective de calcul

un "modèle de calcul"

Exemple : les MACHINES À REGISTRES

Exemple : les MACHINES À REGISTRES



(Minsky 1961)

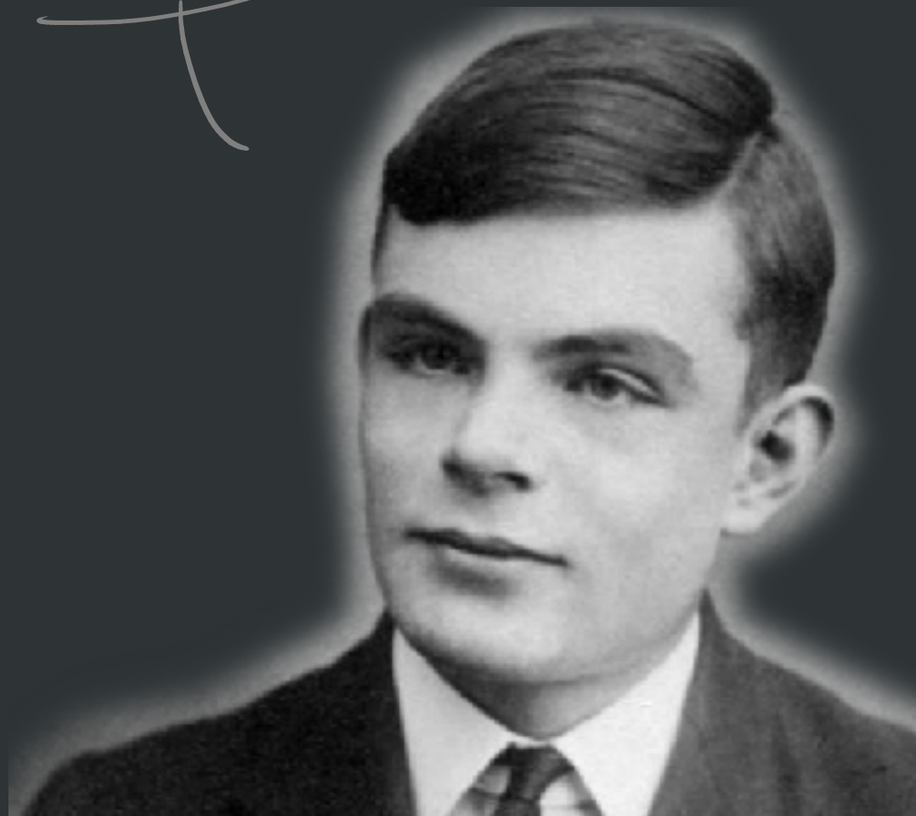
mais aussi (et surtout),

La MACHINE de TURING

(Turing 1936)

T avant les ordinateurs !

Yo

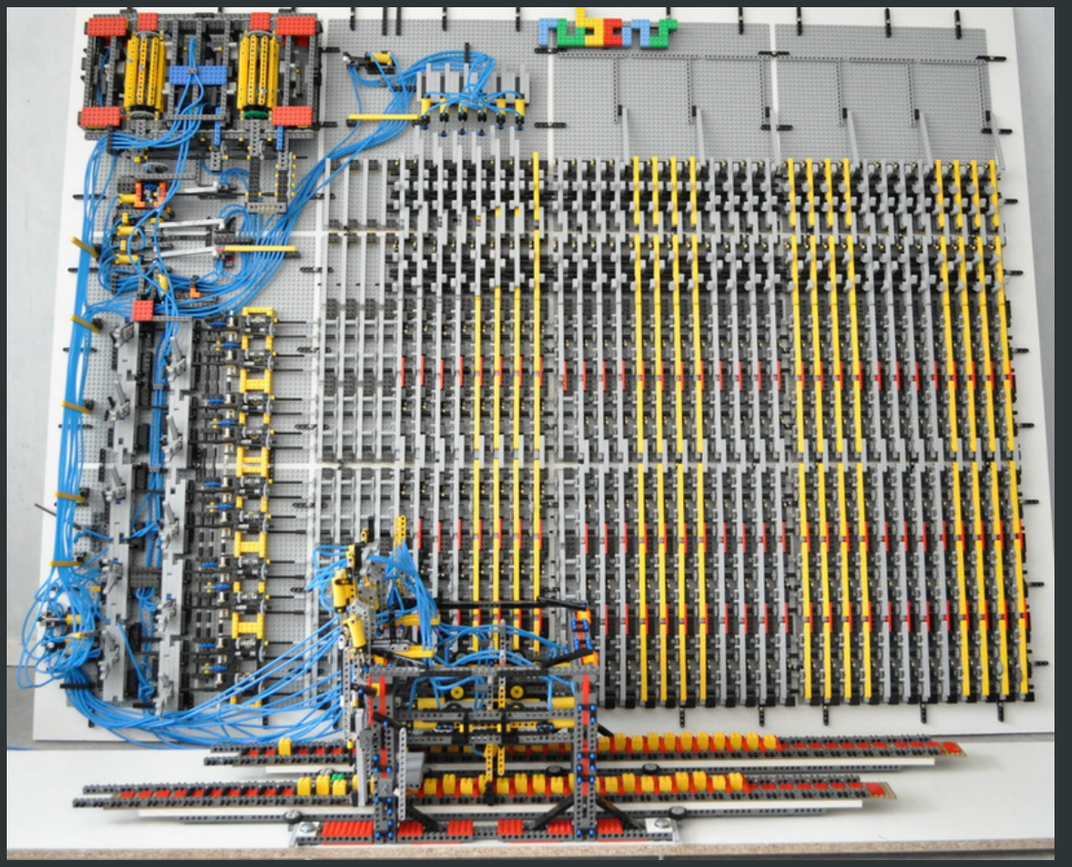


mais aussi (et surtout),

La MACHINE de TURING

Une vraie machine ...

en Lego

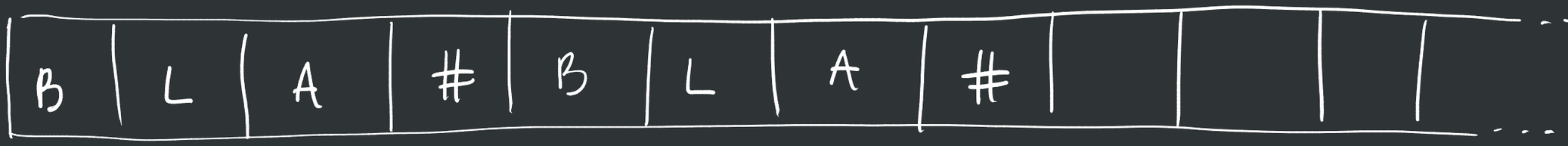


et un joli modèle théorique.

Une machine de Turing, c'est :

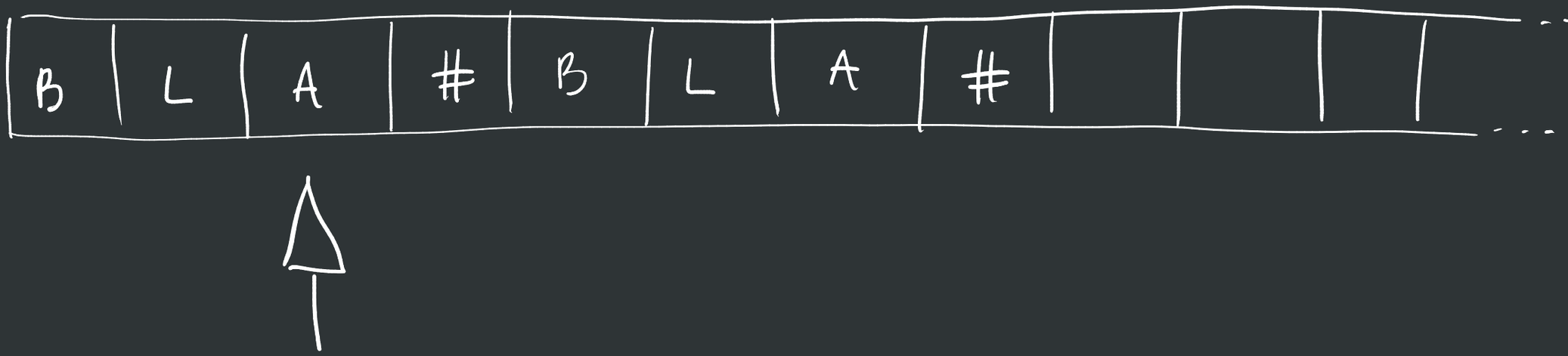
Une machine de Turing, c'est :

- un ruban infini avec des cases



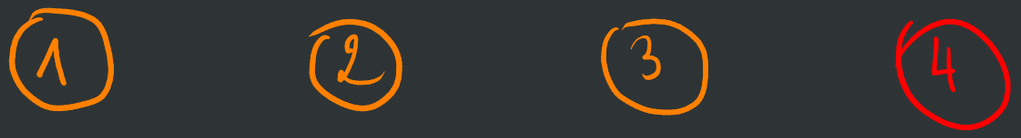
Une machine de Turing, c'est :

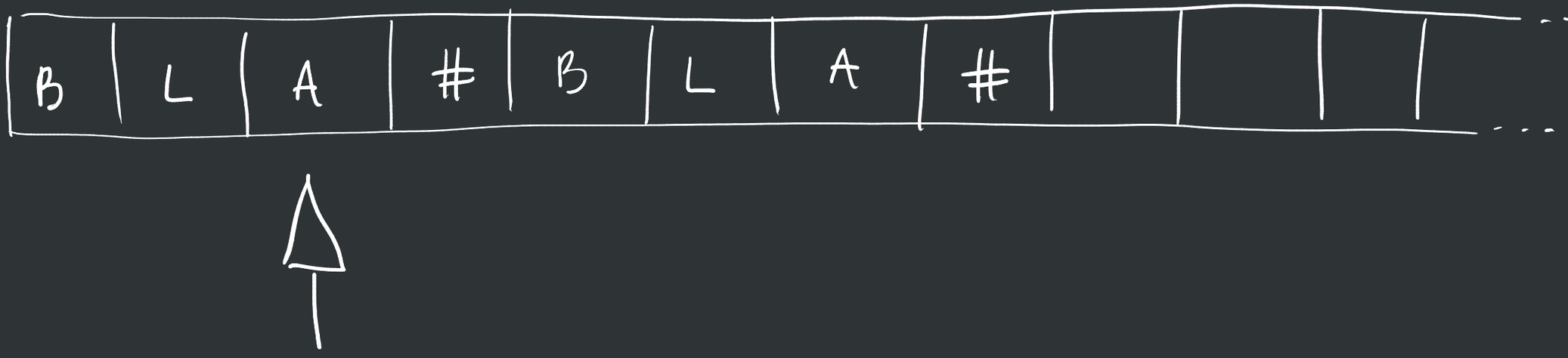
- un ruban infini avec des cases
- une tête de lecture du ruban



Une machine de Turing, c'est :

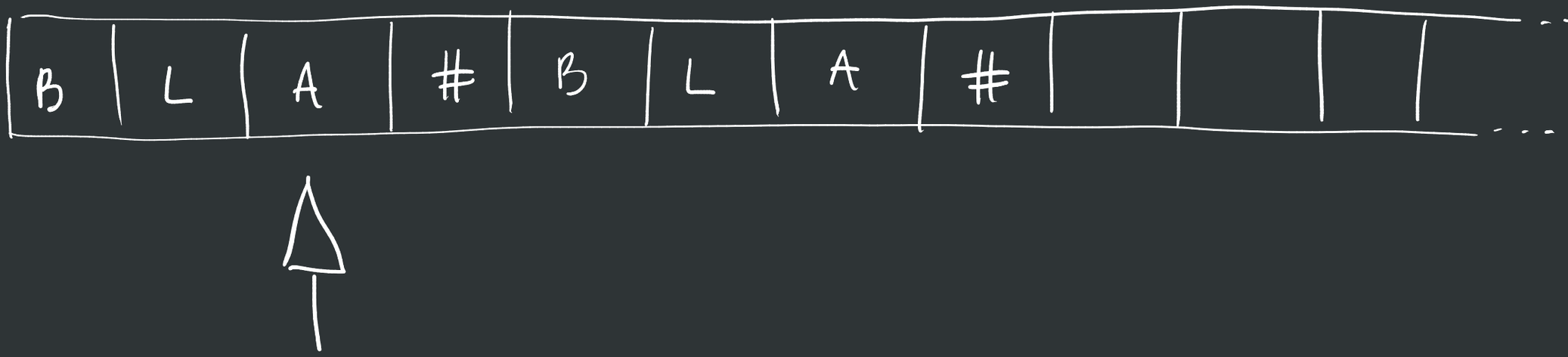
- un ruban infini avec des cases
- une tête de lecture du ruban
- des états, dont certains sont acceptants





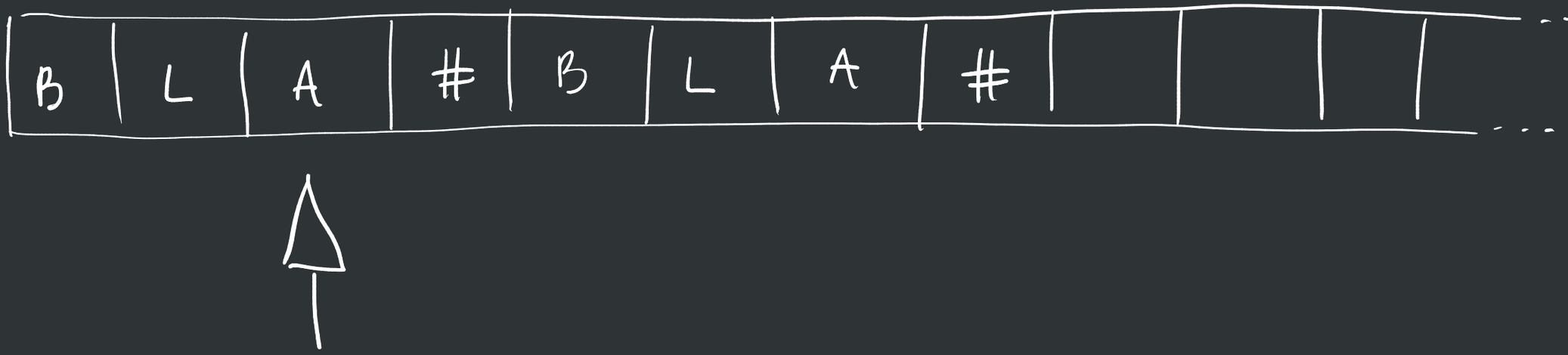
Une machine de Turing, c'est :

- un ruban infini avec des cases
- une tête de lecture du ruban
- des états, dont certains sont acceptants
- une fonction de transition $\text{état} \times \text{lettre} \to \text{état} \times \text{lettre} \times \{\leftarrow, \rightarrow\}$



Une machine de Turing, c'est :

- un ruban infini avec des cases
- une tête de lecture du ruban
- des états, dont certains sont acceptants
- une fonction de transition $\text{état} \times \text{lettre} \to \text{état} \times \text{lettre} \times \{\leftarrow, \rightarrow\}$
  - ↳ fonction, vraiment ?



Une machine de Turing, c'est :

- un ruban infini avec des cases
- une tête de lecture du ruban
- des états, dont certains sont acceptants
- une fonction de transition $\text{état} \times \text{lettre} \to \text{état} \times \text{lettre} \times \{\leftarrow, \rightarrow\}$

Formellement :

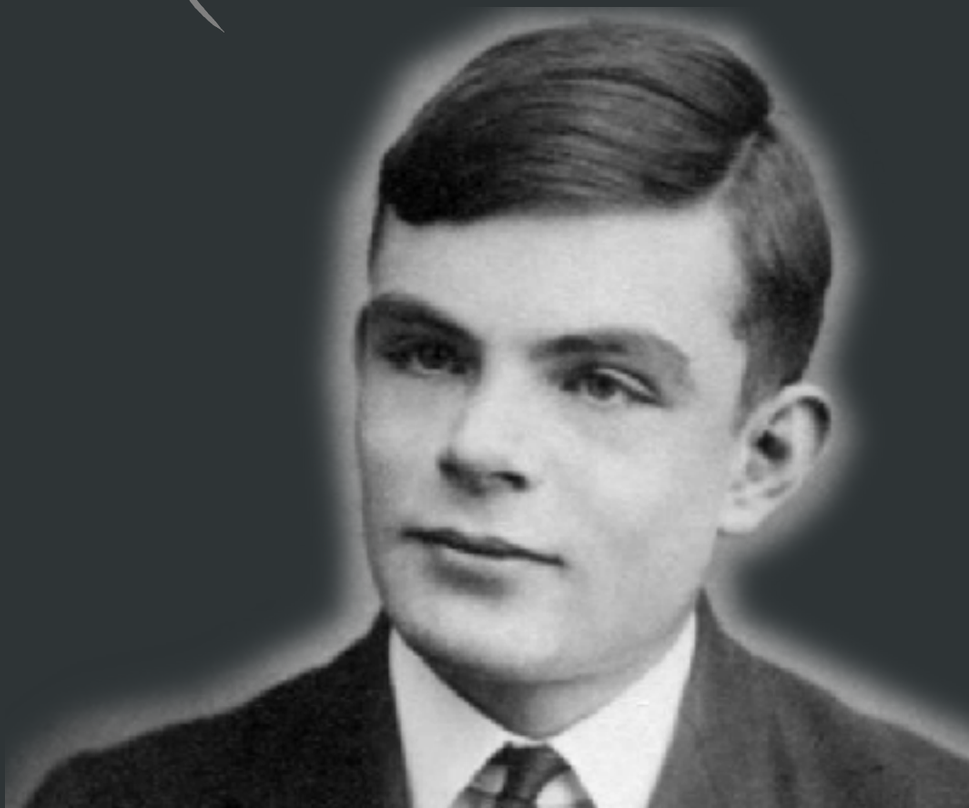
$$(\Gamma, \Sigma, Q, q_0, Q_A, \delta)$$

$\Sigma \subsetneq \Gamma$ $\quad q_0 \in Q$ $\quad Q_A \subseteq Q$ $\quad \delta \subseteq Q \times \Gamma \times Q \times \Gamma \times \{\leftarrow, \rightarrow\}$

ici dans le cas NON DÉTERMINISTE

Ouh la la
tout ça mérite bien
un petit exemple

Bon, d'accord.



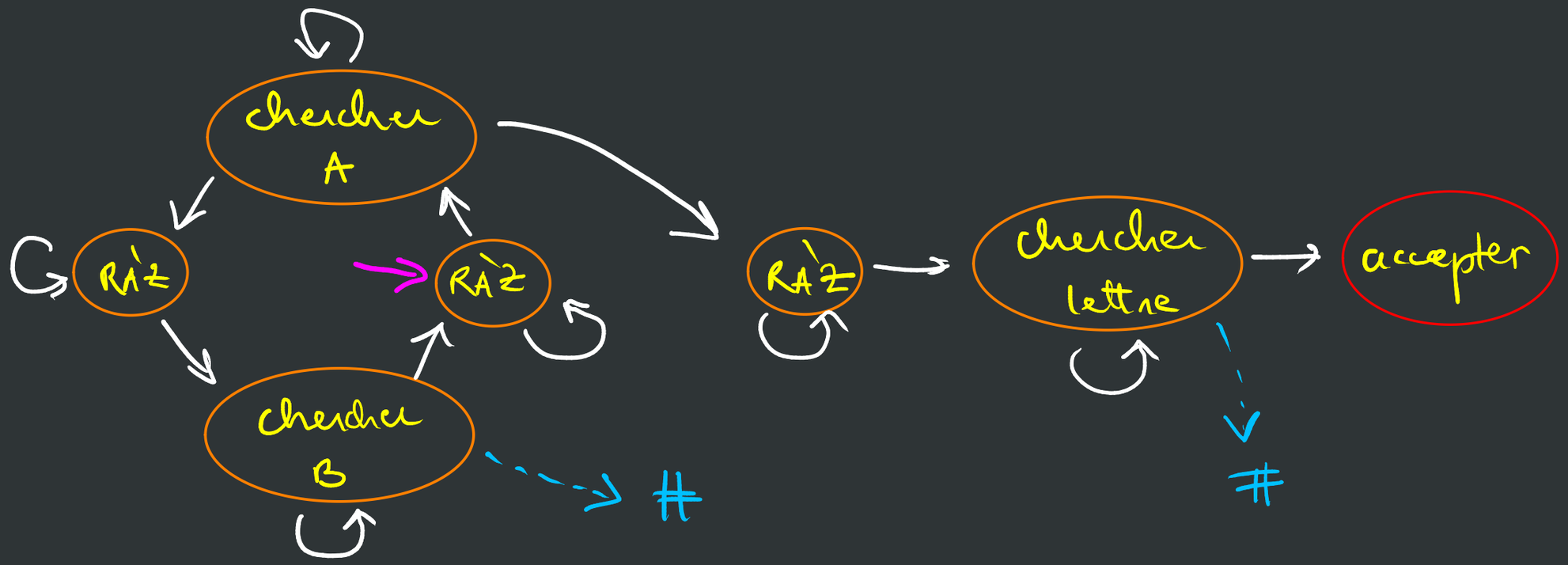
Une machine de Turing qui dit s'il y a le même nombre de A et de B dans un mot

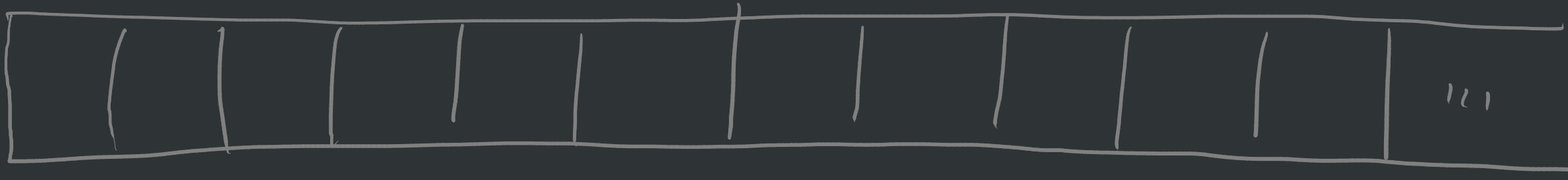
Une machine de Turing qui dit s'il y a le même nombre de A et de B dans un mot

$\Sigma = \{A, B\}$ $\Gamma = \{A, B, C, \#\}$

Une machine de Turing qui dit s'il y a le même nombre de A et de B dans un mot

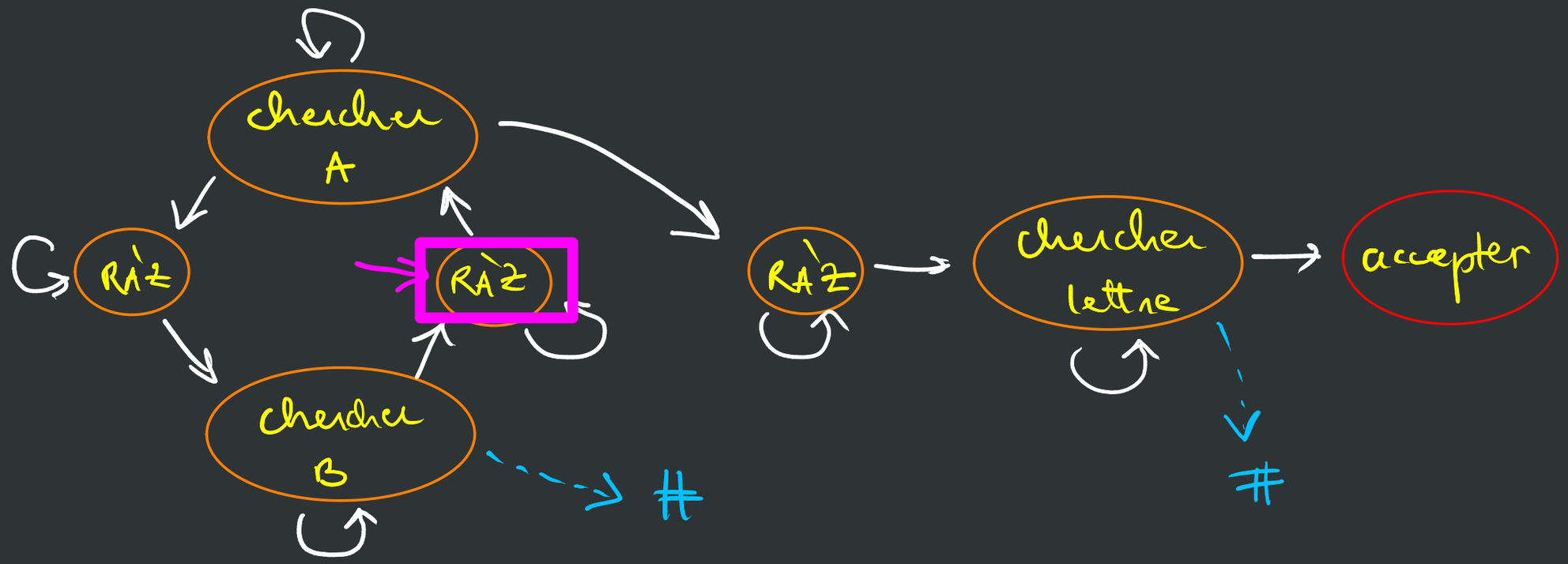
$\Sigma = \{A, B\}$ $\Gamma = \{A, B, C, \#\}$

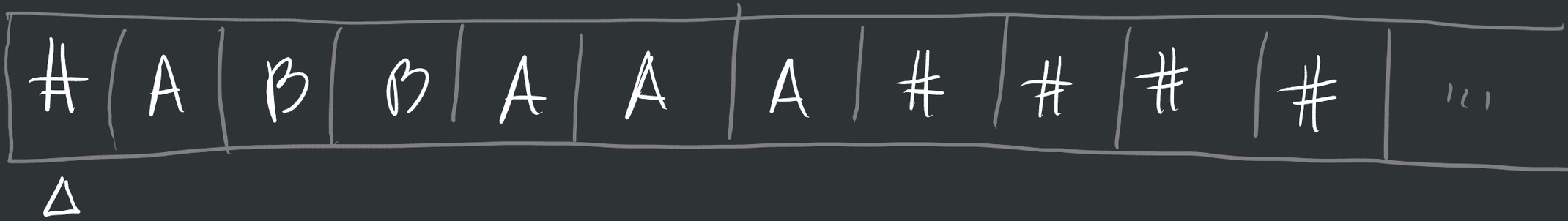




Une machine de Turing qui dit s'il y a le même nombre de A et de B dans un mot

$\Sigma = \{A, B\}$ $\quad\quad$ $\Gamma = \{A, B, C, \#\}$

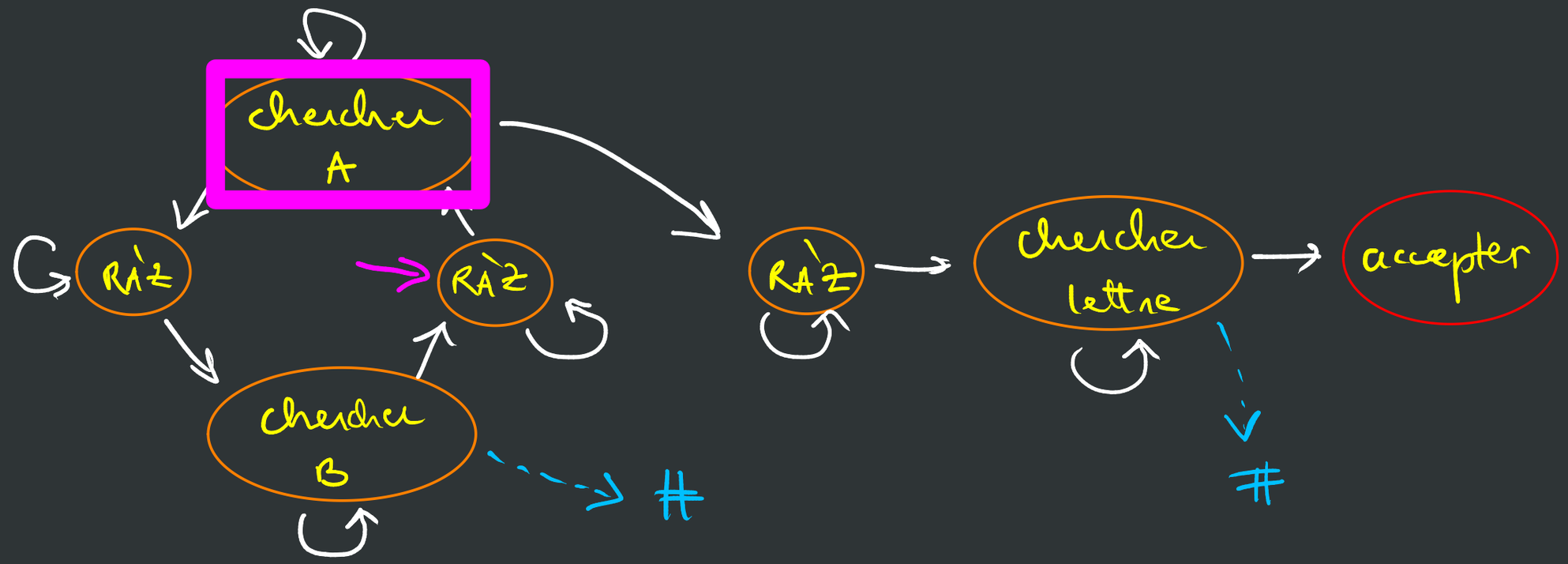




△

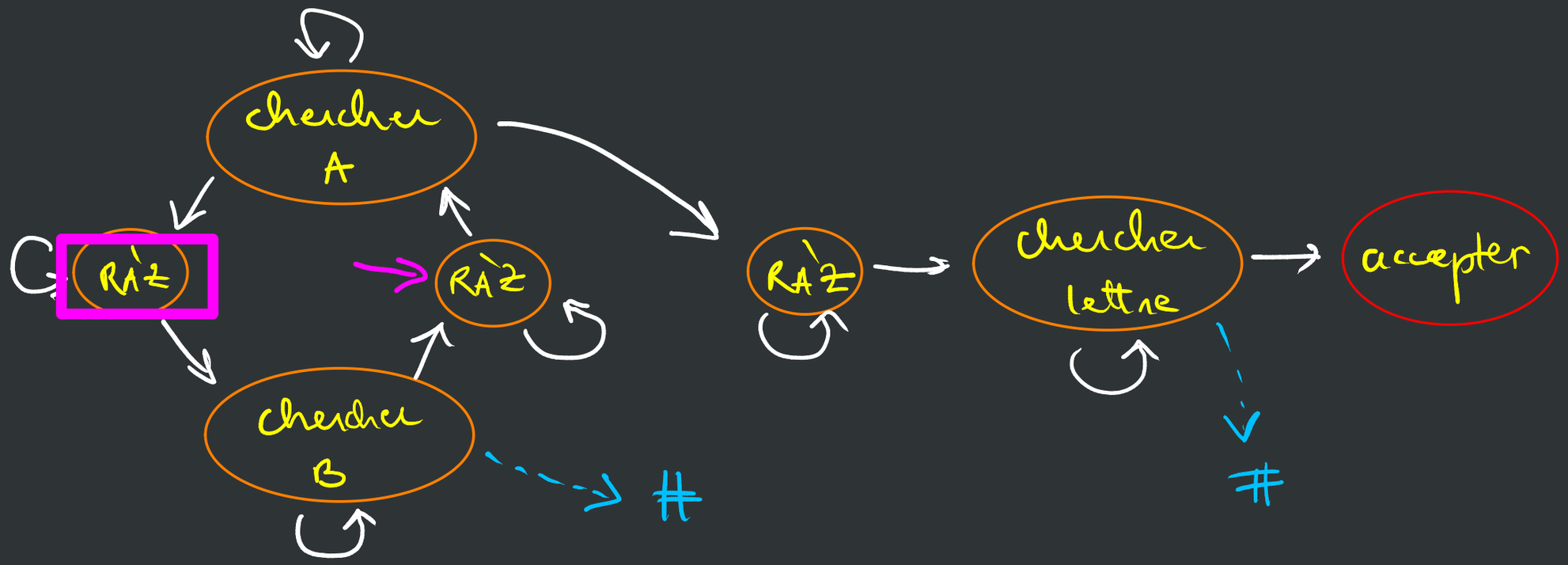
Une machine de Turing qui dit s'il y a le même nombre de A et de B dans un mot

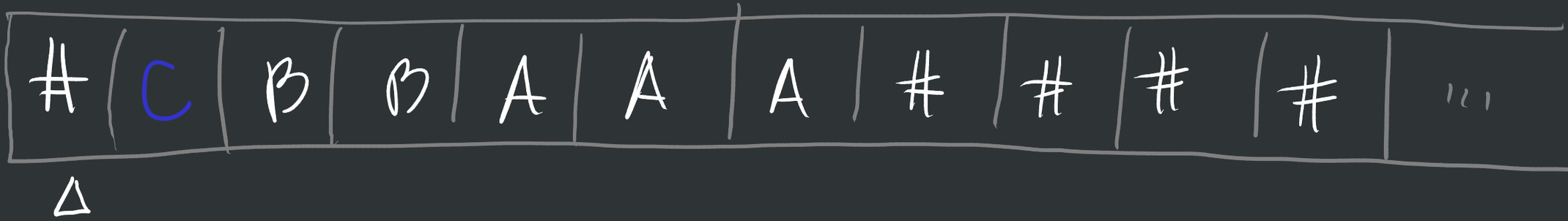
$\Sigma = \{A, B\}$ $\quad\quad$ $\Gamma = \{A, B, C, \#\}$



Une machine de Turing qui dit s'il y a le même nombre de A et de B dans un mot

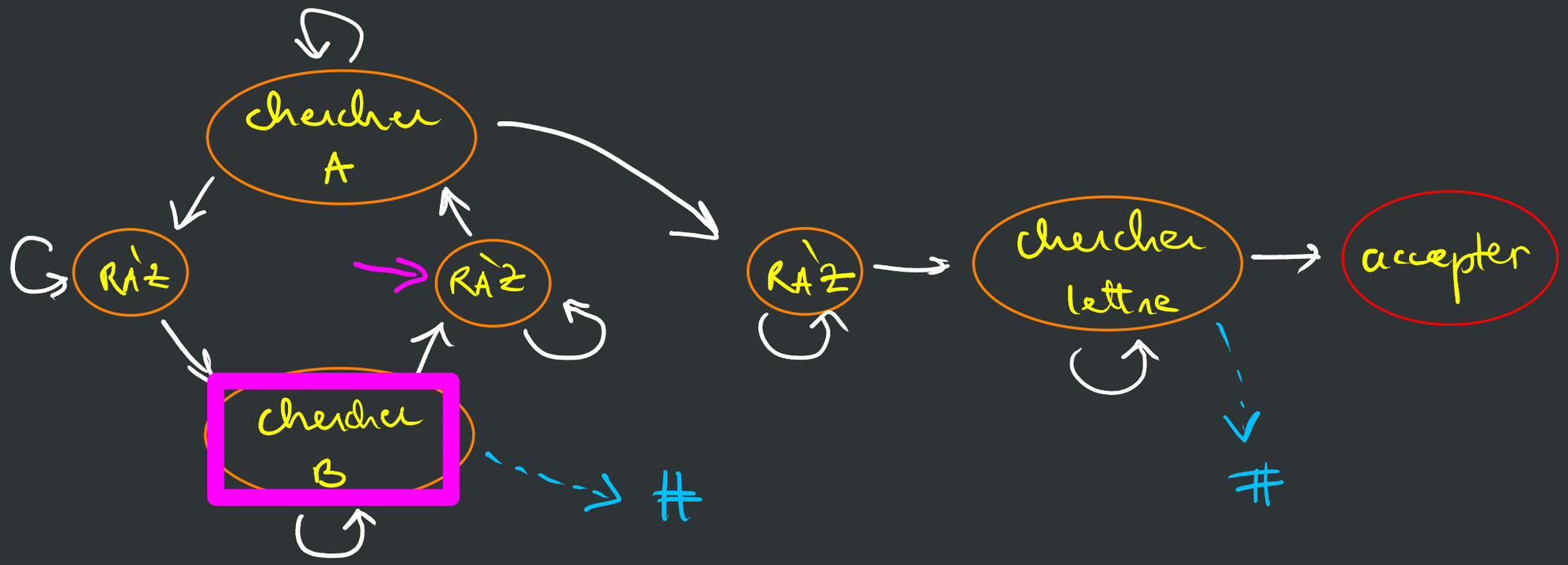
$\Sigma = \{A, B\}$ $\quad\quad$ $\Gamma = \{A, B, C, \#\}$

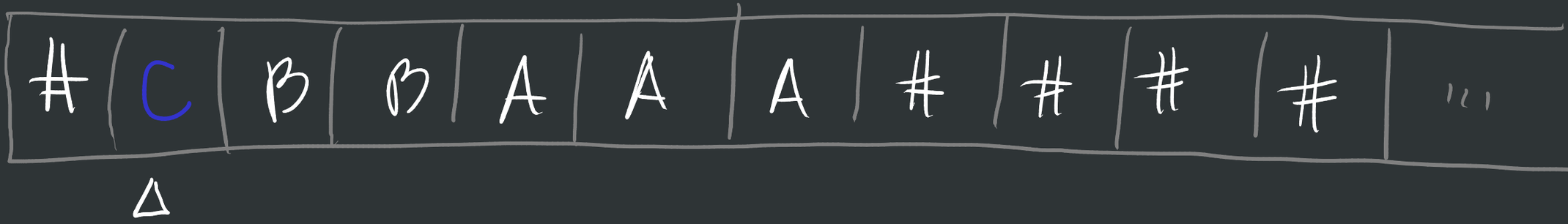




Une machine de Turing qui dit s'il y a le même nombre de A et de B dans un mot

$\Sigma = \{A, B\}$ $\quad\quad$ $\Gamma = \{A, B, C, \#\}$

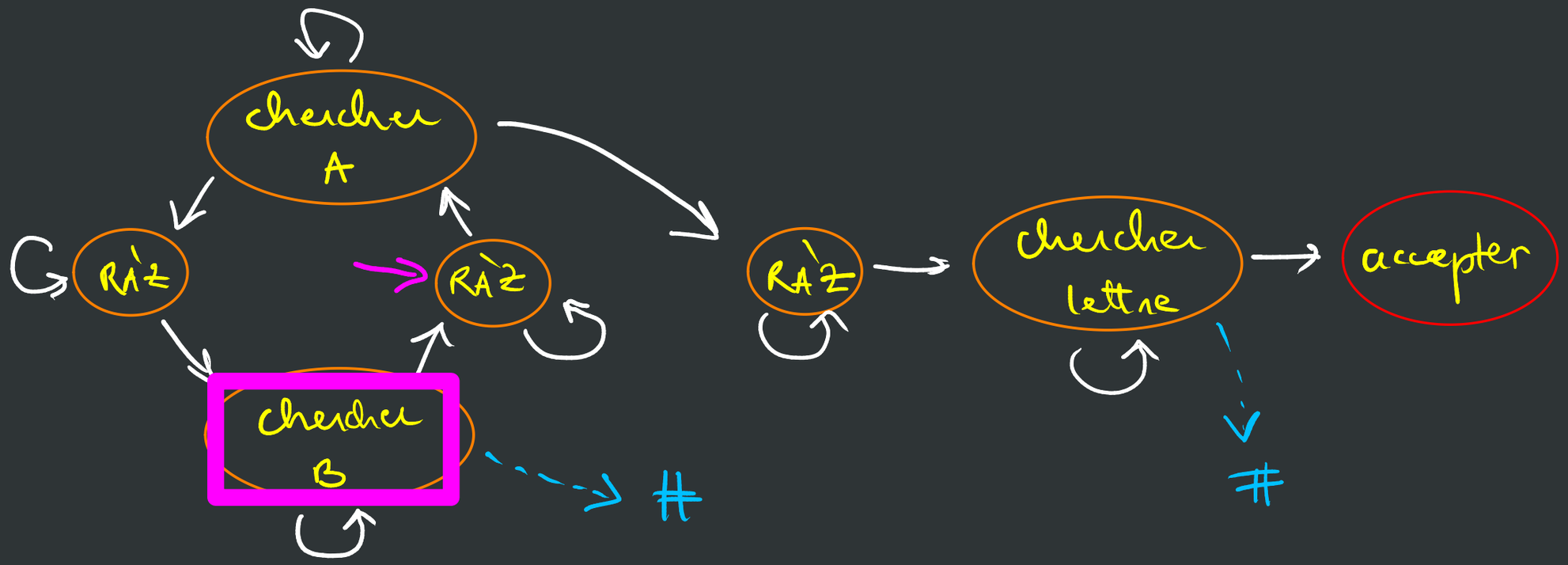




△

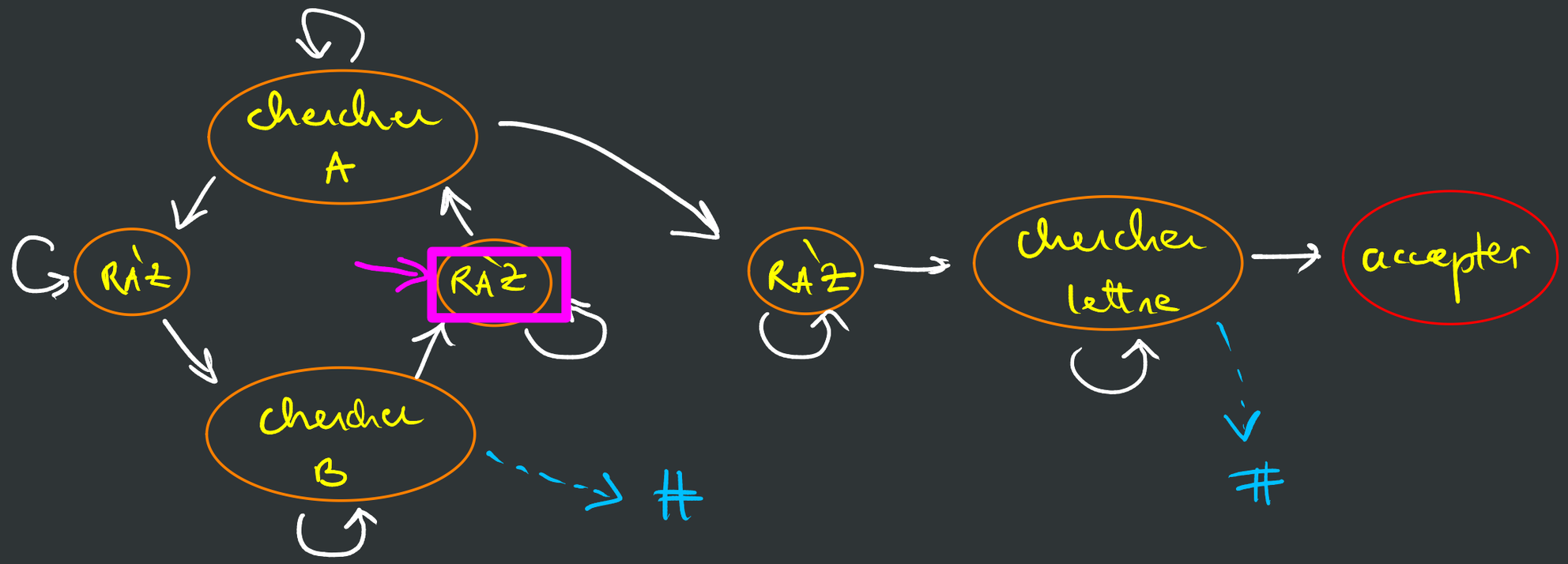
Une machine de Turing qui dit s'il y a le même nombre de A et de B dans un mot

$\Sigma = \{A, B\}$ $\quad\quad$ $\Gamma = \{A, B, C, \#\}$



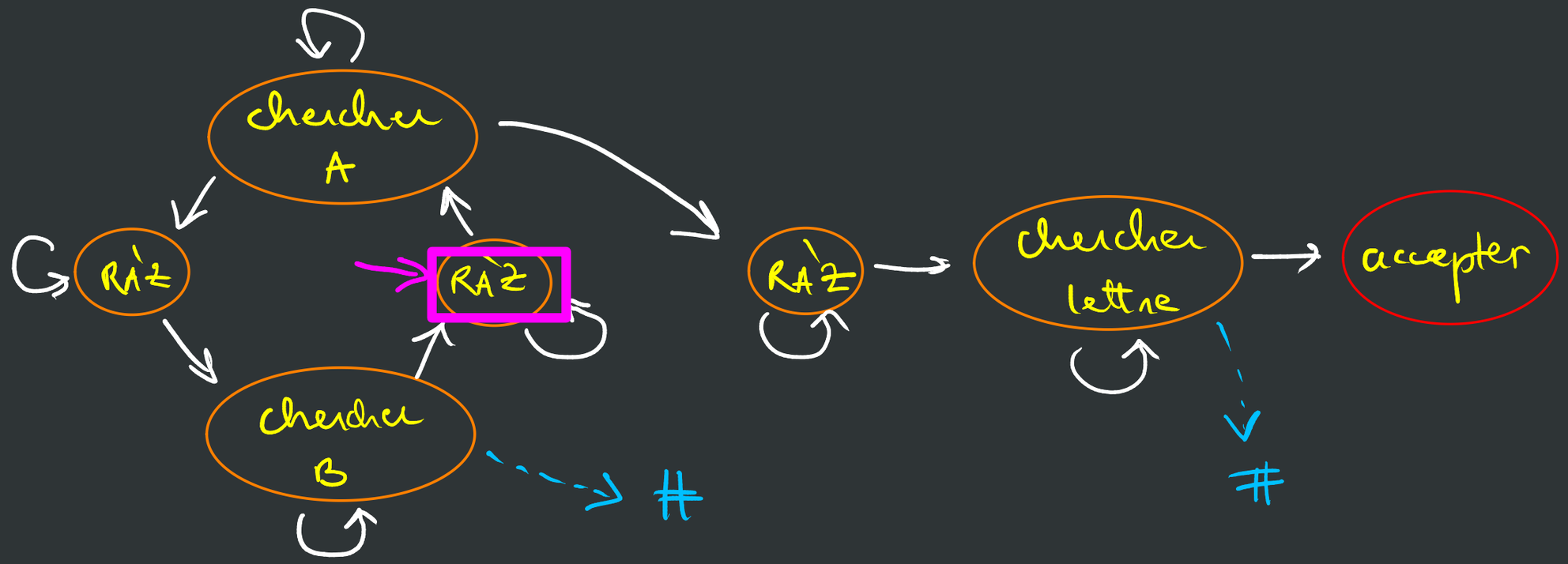
Une machine de Turing qui dit s'il y a le même nombre de A et de B dans un mot

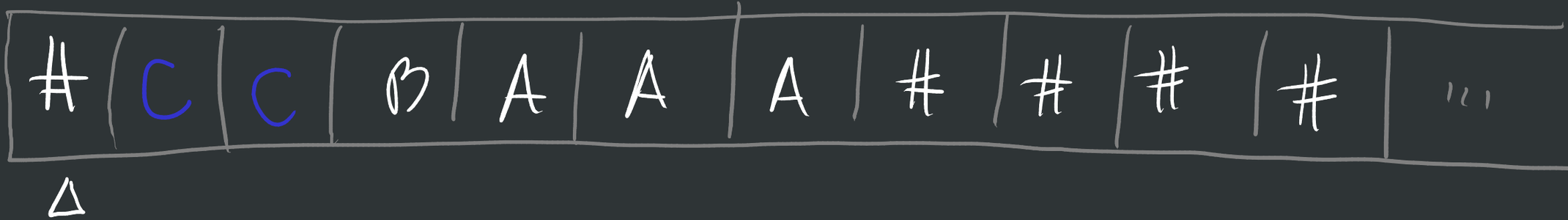
$\Sigma = \{A, B\}$ $\quad\quad$ $\Gamma = \{A, B, C, \#\}$



Une machine de Turing qui dit s'il y a le même nombre de A et de B dans un mot

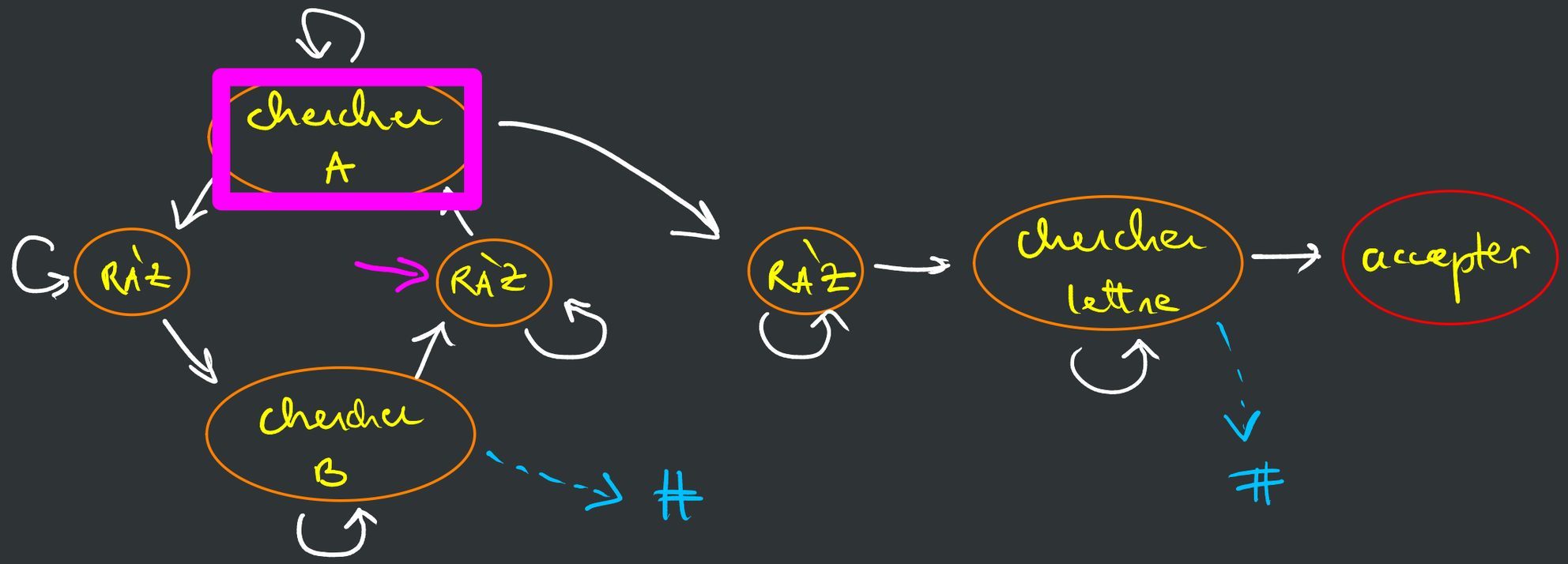
$\Sigma = \{A, B\}$ $\quad\quad$ $\Gamma = \{A, B, C, \#\}$





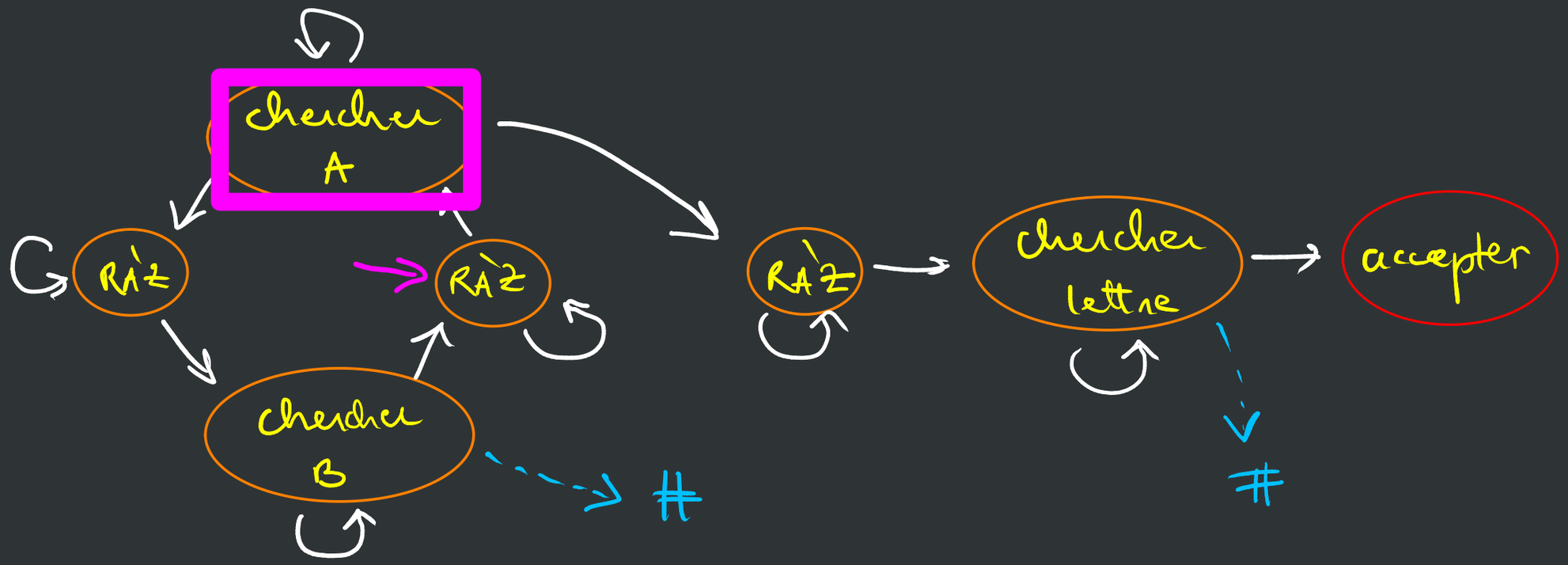
Une machine de Turing qui dit s'il y a le même nombre de A et de B dans un mot

$\Sigma = \{A, B\}$ $\quad\quad$ $\Gamma = \{A, B, C, \#\}$



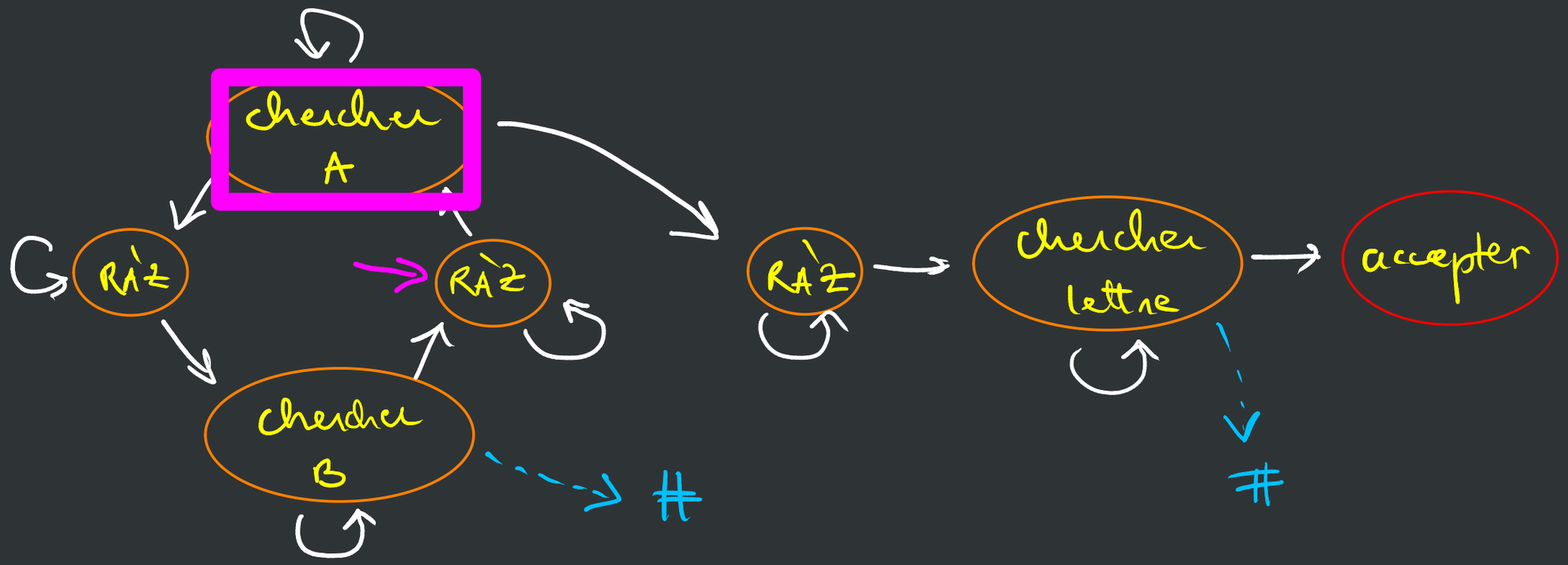
Une machine de Turing qui dit s'il y a le même nombre de A et de B dans un mot

$\Sigma = \{A, B\}$ $\quad\quad$ $\Gamma = \{A, B, C, \#\}$



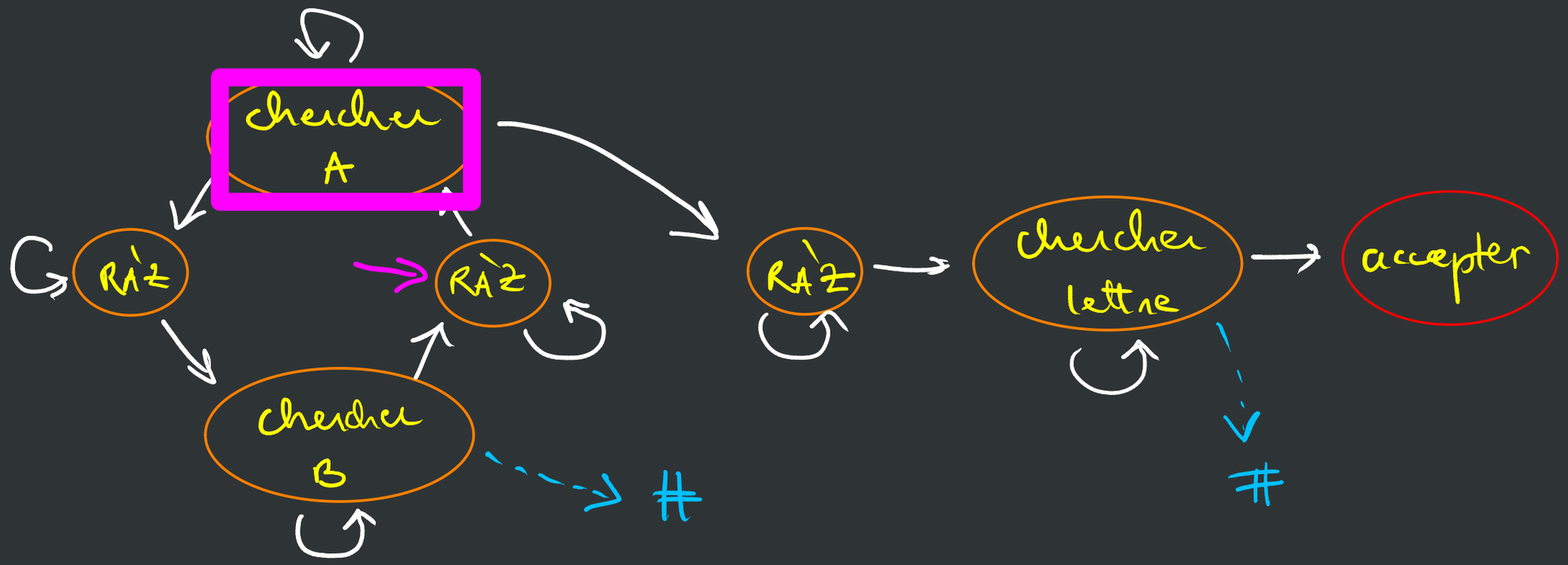
Une machine de Turing qui dit s'il y a le même nombre de A et de B dans un mot

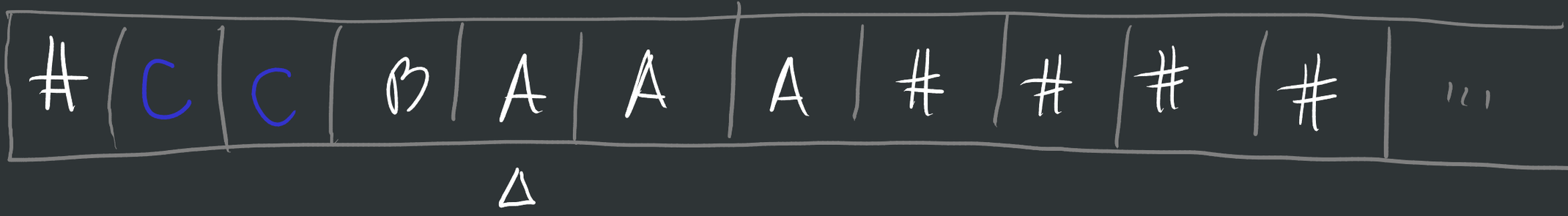
$\Sigma = \{A, B\}$ $\quad\quad$ $\Gamma = \{A, B, C, \#\}$



Une machine de Turing qui dit s'il y a le même nombre de A et de B dans un mot

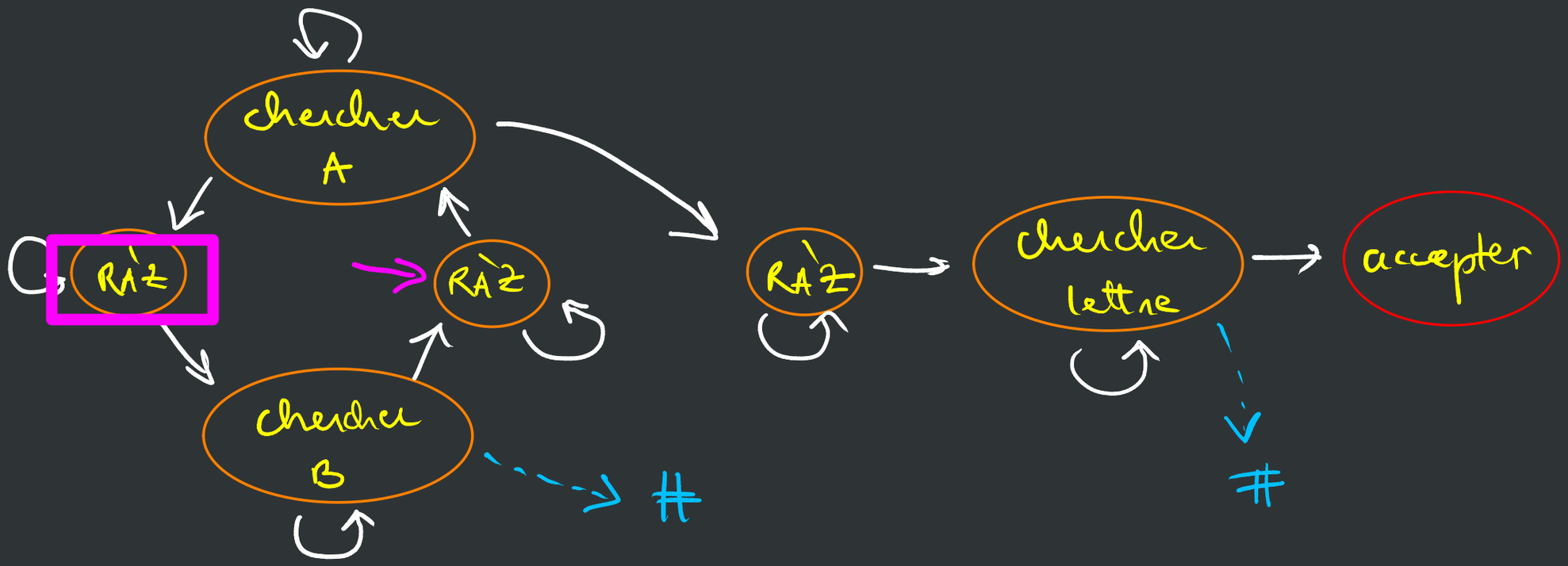
$\Sigma = \{A, B\}$ $\quad\quad$ $\Gamma = \{A, B, C, \#\}$





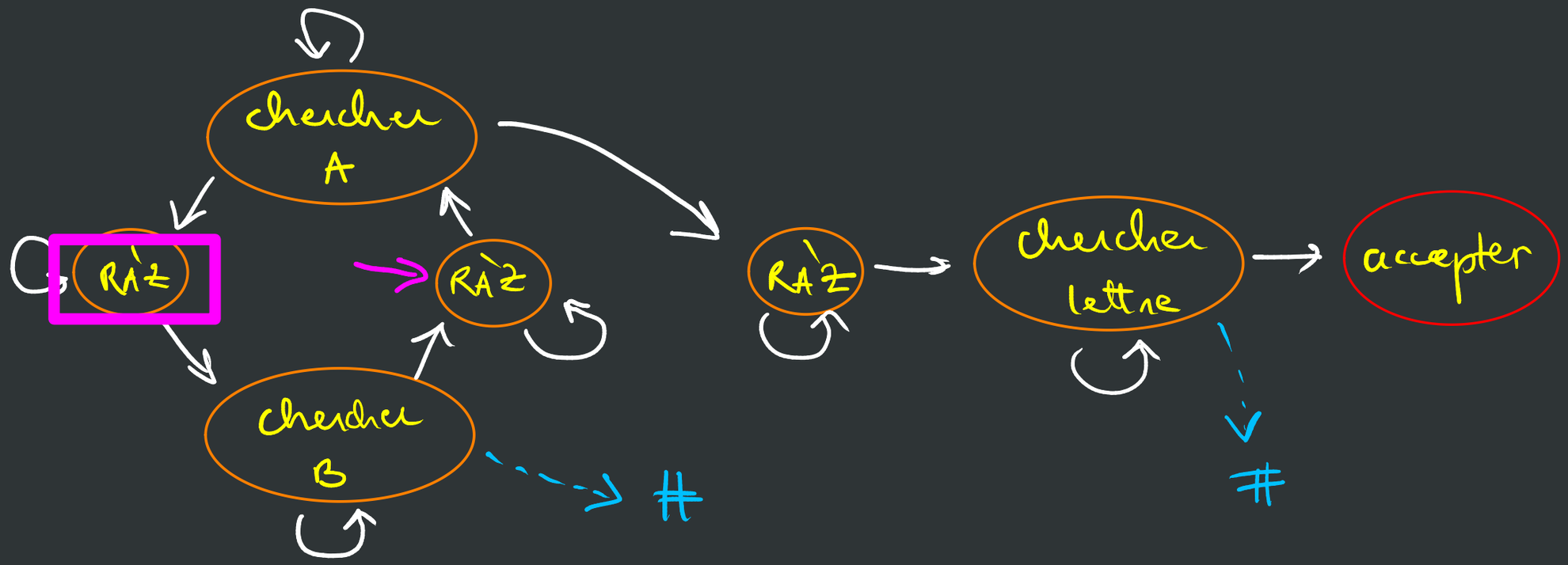
Une machine de Turing qui dit s'il y a le même nombre de A et de B dans un mot

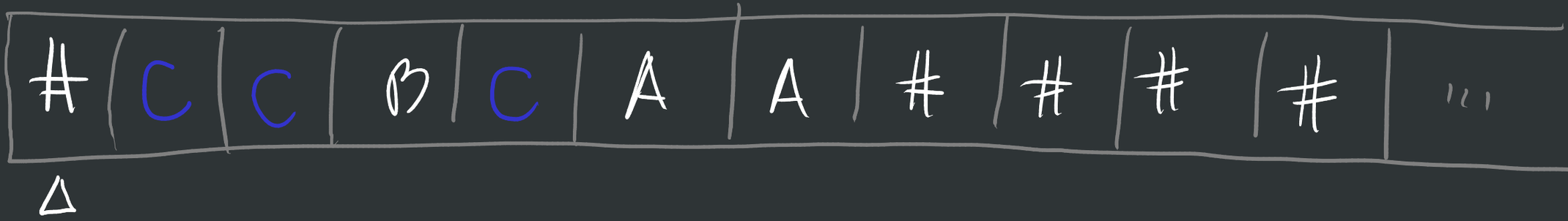
$\Sigma = \{A, B\}$ $\quad\quad$ $\Gamma = \{A, B, C, \#\}$



Une machine de Turing qui dit s'il y a le même nombre de A et de B dans un mot

$\Sigma = \{A, B\}$ $\quad\quad$ $\Gamma = \{A, B, C, \#\}$

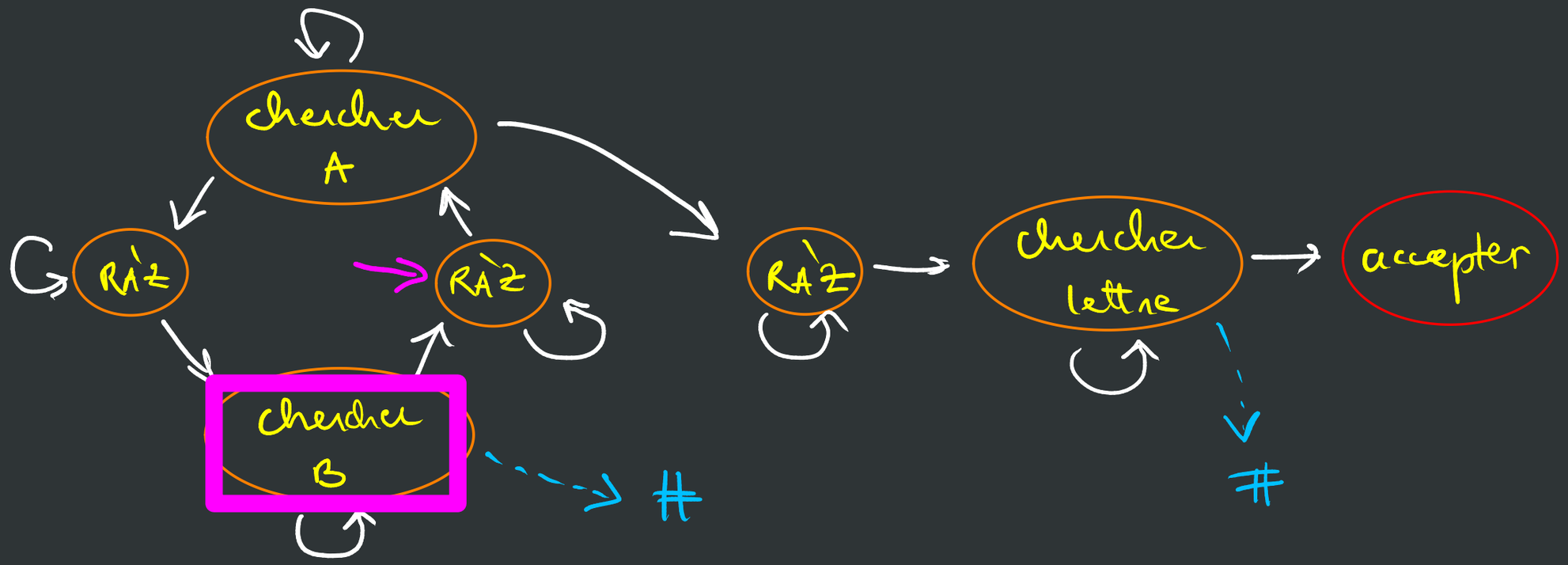


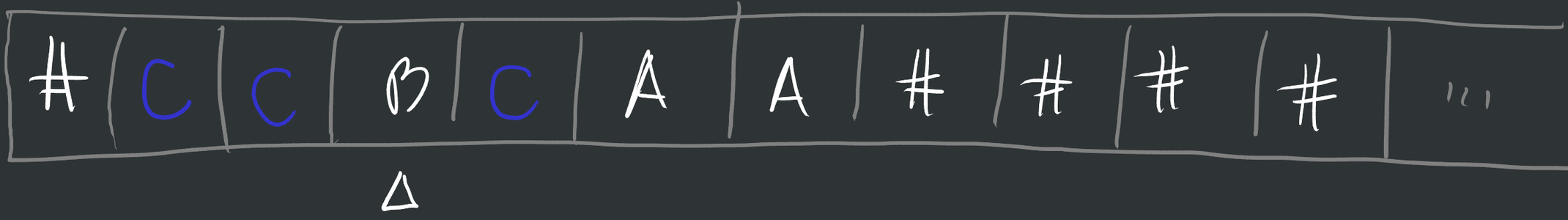


Δ

Une machine de Turing qui dit s'il y a le même nombre de A et de B dans un mot

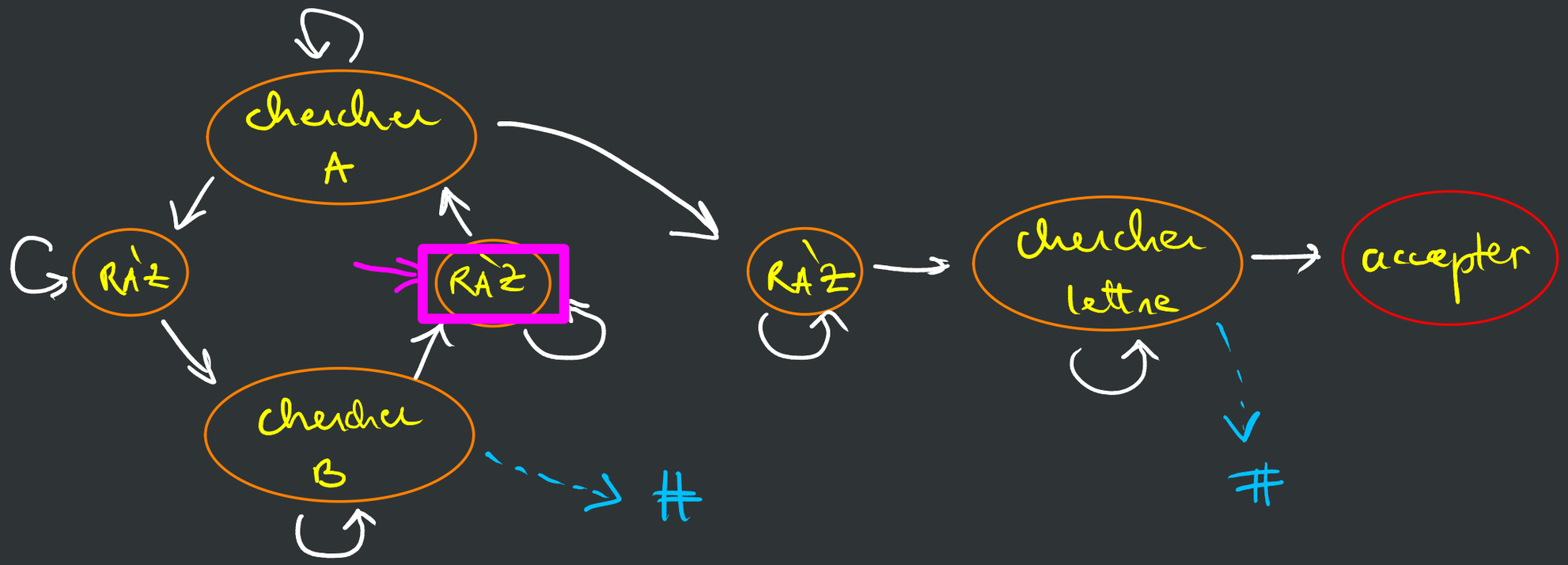
$\Sigma = \{A, B\}$ $\quad\quad$ $\Gamma = \{A, B, C, \#\}$





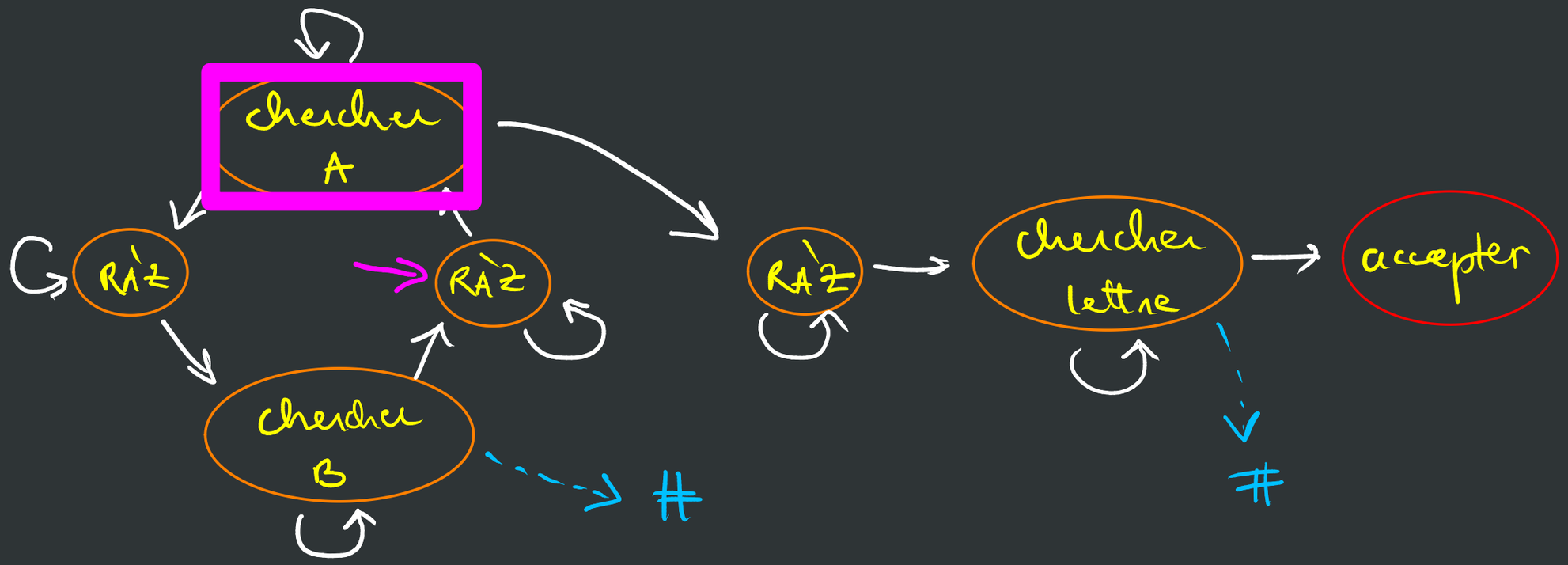
Une machine de Turing qui dit s'il y a le même nombre de A et de B dans un mot

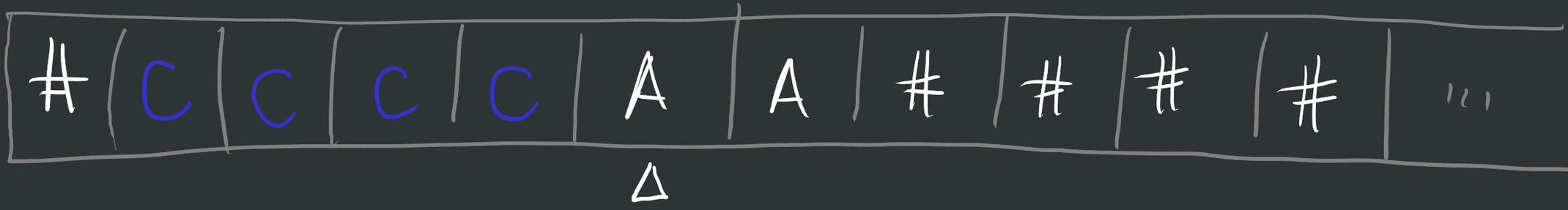
$\Sigma = \{A, B\}$ $\quad\quad$ $\Gamma = \{A, B, C, \#\}$



Une machine de Turing qui dit s'il y a le même nombre de A et de B dans un mot

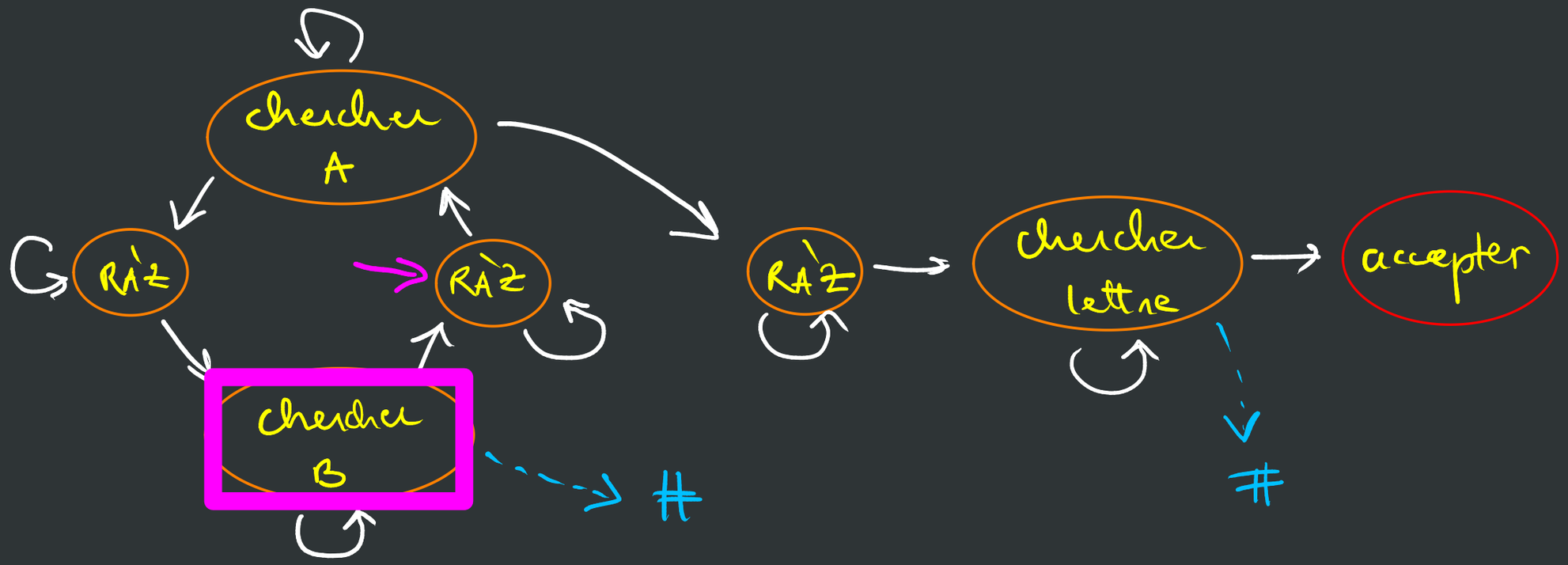
$\Sigma = \{A, B\}$ $\quad\quad$ $\Gamma = \{A, B, C, \#\}$





Une machine de Turing qui dit s'il y a le même nombre de A et de B dans un mot

$\Sigma = \{A, B\}$ $\quad\quad$ $\Gamma = \{A, B, C, \#\}$



Δ

Déf Un langage $L \subseteq \Sigma^*$ est accepté par une machine M si M accepte exactement les mots de L.

Déf Un langage $L \subseteq \Sigma^*$ est décidé par une machine M si M

- $\rightarrow$ accepte les mots de L
- $\rightarrow$ rejette les mots de $\Sigma^* - L$.

Déf Les langages acceptés (resp. décidés) par au moins une machine de Turing sont dits récursivement énumérables (resp. récursifs).

question : est-ce que tout est décidable ?

Question : est-ce que tout est décidable ?

Non.

Théorème : Le problème de l'arrêt, défini par :

→ Entrée = une machine M et une entrée w de M

→ Sortie = VRAi si M s'arrête sur l'entrée w

n'est décidé par aucune machine de Turing.

(Turing 1936)

Soit A une machine décidant le problème de l'arrêt : A accepte $\langle M, w \rangle$ si M s'arrête sur w
A rejette $\langle M, w \rangle$ si M boucle sur w.

Construisons la machine P définie sur l'entrée $\langle M \rangle$ par :

si A accepte $\langle M, \langle M \rangle \rangle$
alors boucler
sinon accepter

Soit A une machine décidant le problème de l'arrêt : A accepte $\langle M, w \rangle$ si M s'arrête sur w
A rejette $\langle M, w \rangle$ si M boucle sur w.

Construisons la machine P définie sur l'entrée $\langle M \rangle$ par :

si M s'arrête sur $\langle M \rangle$
alors boucler
sinon accepter

Soit A une machine décidant le problème de l'arrêt : A accepte $\langle M, w \rangle$ si M s'arrête sur w
A rejette $\langle M, w \rangle$ si M boucle sur w.

Construisons la machine P définie sur l'entrée $\langle M \rangle$ par :

si M s'arrête sur $\langle M \rangle$
alors boucler
sinon accepter

alors P s'arrête sur $\langle P \rangle$

$\Updownarrow$

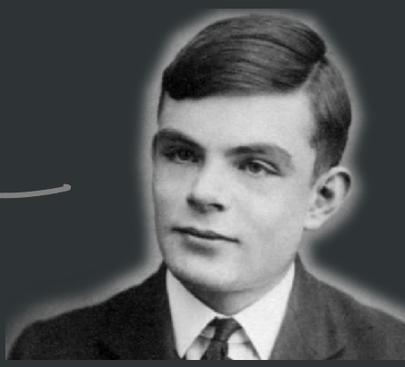
P boucle sur $\langle P \rangle$,

donc A n'existe pas. □

Exercice 1 : Le problème de l'arrêt est indécidable mais il est récursivement énumérable.

Exercice 2 : il existe des langages qui ne sont même pas récursivement énumérables.

indice : Combien y a-t-il de machines de Turing ?



Exercice 1 : Le problème de l'arrêt est indécidable mais il est récursivement énumérable.

Exercice 2 : il existe des langages qui ne sont même pas récursivement énumérables.

Un autre exemple connu (mais compliqué) :

Th. de Matiyasevitch (10^e problème de Hilbert) :

La résolution des équations diophantiennes est indécidable.

Thèse de Church-Turing : Tout modèle de calcul raisonnable définit la même notion de décidabilité que les machines de Turing.

C'est le cas notamment :

→ des fonctions récursives

→ du λ-calcul ♡

Thèse de Church-Turing : Tout modèle de calcul raisonnable définit la même notion de décidabilité que les machines de Turing.

C'est le cas notamment :

→ des fonctions récursives

→ du λ-calcul ♡

et des machines à registres !

On sait désormais ce qu'est un calcul impossible. Qu'en est des calculs difficiles ?

Déf : Le temps d'exécution de M sur w est :

$T_M(w)$ = le nombre d'étapes de calcul de M sur l'entrée w

Déf : Le temps d'exécution de M sur w est :

$$T_M(w) = \min_{\text{calcul}} \{ \text{le nombre d'étapes de calcul de M sur l'entrée } w \}$$

Déf : Le **temps d'exécution** de M sur w est :

$$T_M(w) = \min_{\text{calcul}} \{ \text{le nombre d'étapes de calcul de M sur l'entrée w} \}$$

Déf : La classe **P** des problèmes résolubles en temps déterministe polynomial est celle de tous les problèmes tels que :

$\exists$ M déterministe qui résout le problème tq

$\exists$ P un polynôme,

$\forall$ w entrée du problème, $T_M(w) \leq P(|w|)$.

Déf : Le **temps d'exécution** de M sur w est :

$$T_M(w) = \min_{\text{calcul}} \{ \text{le nombre d'étapes de calcul de M sur l'entrée w} \}$$

Déf : La classe **P** des problèmes résolubles en temps déterministe polynomial est celle de tous les problèmes tels que :

$\exists$ M déterministe qui résout le problème tq

$\exists$ P un polynôme,

$\forall$ w entrée du problème, $T_M(w) \leq P(|w|)$.

↓

attention au codage !

Déf : Le temps d'exécution de M sur w est :

$$T_M(w) = \min_{\text{calcul}} \{ \text{le nombre d'étapes de calcul de M sur l'entrée w} \}$$

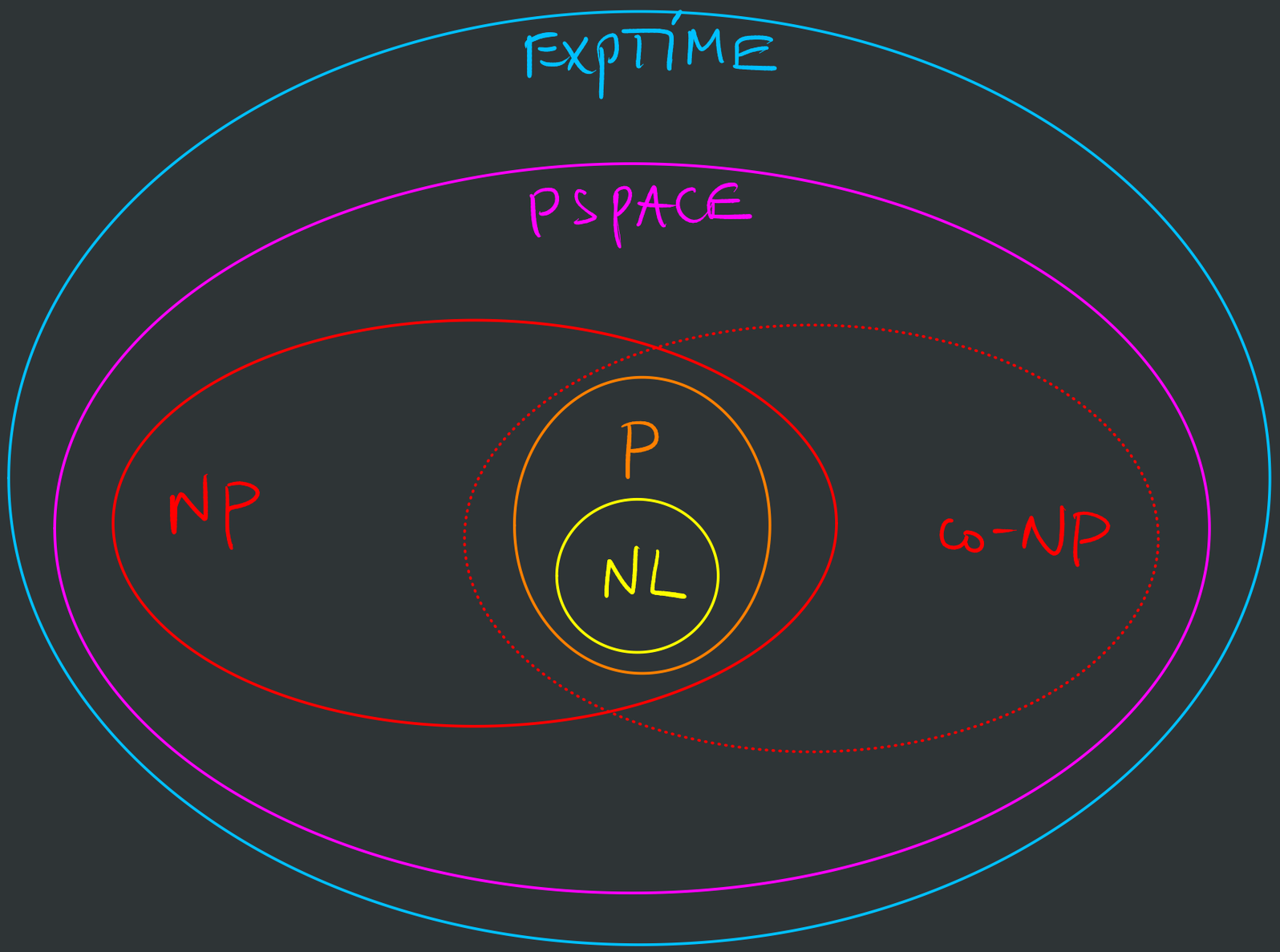
Déf : La classe NP des problèmes résolubles en temps non-déterministe polynomial est celle de tous les problèmes tels que :

$\exists$ M non-déterministe qui résout le problème tq

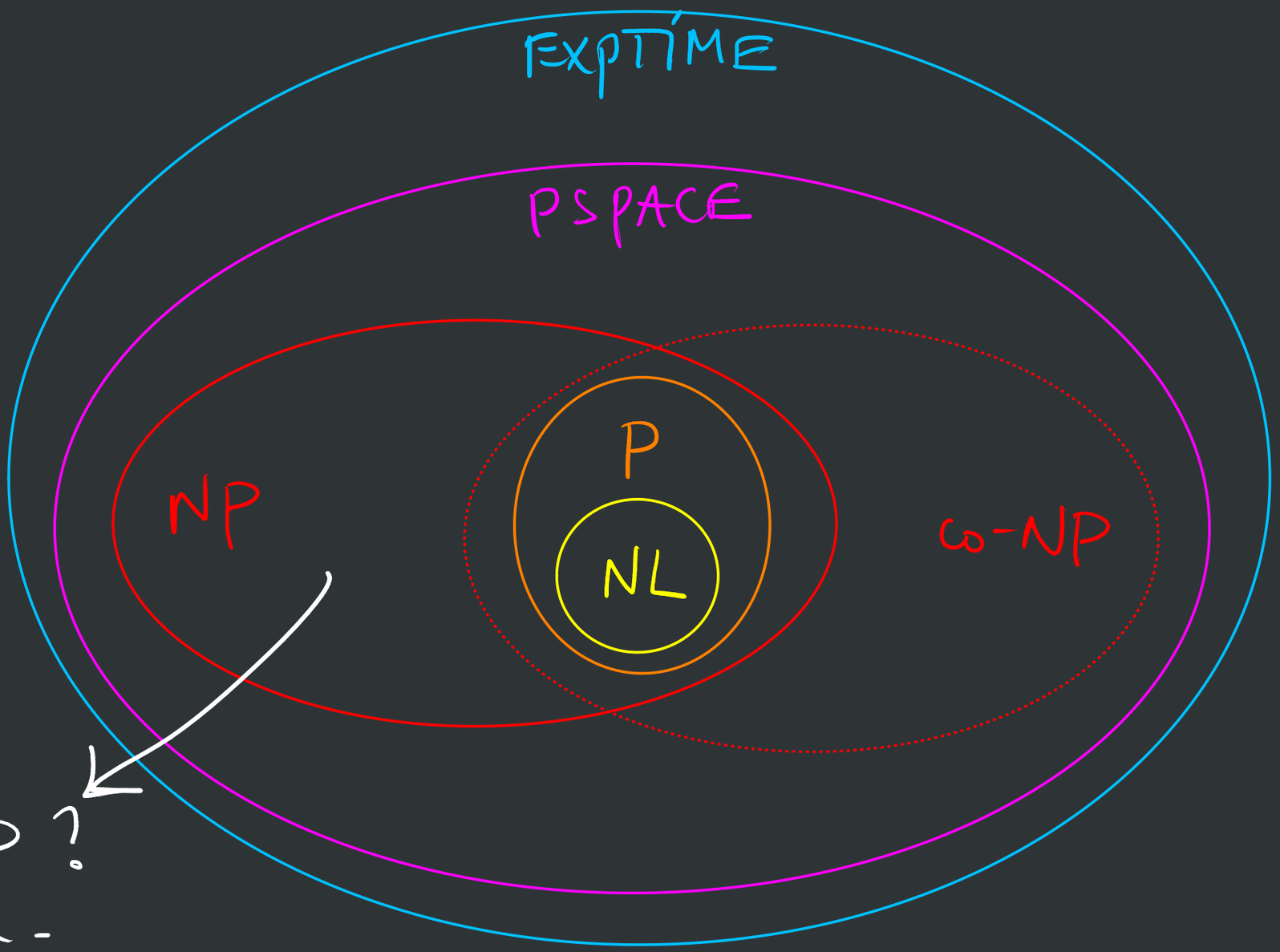
$\exists$ P un polynôme,

$\forall$ w entrée du problème, $T_M(w) \leq P(|w|)$.

D'autres classes de complexité :



D'autres classes de complexité :



P=NP ?
Mystère -

Déf Un problème est NP-complet si

(1) il est dans NP

(2) il est plus dur que tous les problèmes de NP.

Déf Un problème est NP-complet si

(1) il est dans NP

(2) il est plus dur que tous les problèmes de NP.

↓

A est plus dur que B si

si je sais résoudre A en temps $O(f)$ → alors je sais résoudre B en temps $O(f)$

Déf Un problème est NP-complet si

(1) il est dans NP

(2) il est plus dur que tous les problèmes de NP.

↓

A est plus dur que B si

$\varphi(b)$ instance de A ⟵ (temps $O(f)$) b instance de B

tq $\varphi(b)$ solution de A $\Leftrightarrow$ b solution de B

Résolution en temps $O(f)$

Thm (Cook-Levin, 1971) :

il existe un problème NP-complet.

Problème SAT :

→ Entrée = $\bigwedge_{i=1}^{n} \bigvee_{j=1}^{p_n} x_{ij}^{*}$, $x_{ij}^{*} = x_{ij}$ ou $\neg x_{ij}$

→ Sortie = Existe-t-il une instanciation des x_{ij} à $\{\top, \bot\}$ tq la formule soit vraie ?

Thm (Cook-Levin, 1971) :

il existe un problème NP-complet.

Problème 3SAT :

→ Entrée = $\bigwedge_{i=1}^{n} \bigvee_{j=1}^{3} x_{ij}^{*}$, $x_{ij}^{*} = x_{ij}$ ou $\neg x_{ij}$

→ Sortie = Existe-t-il une instanciation des x_{ij} à $\{\top, \bot\}$ tq la formule soit vraie ?

$$\left[x_0 \vee x_1 \vee \bar{x_3}\right] \wedge \left[x_2 \vee \bar{x_2} \vee x_1\right] \wedge \left[x_0 \vee x_1 \vee x_3\right] \wedge \left[x_0 \vee x_2 \vee x_1\right]$$

Thm (Cook-Levin, 1971) :

il existe un problème NP-complet.

Problème 3SAT :

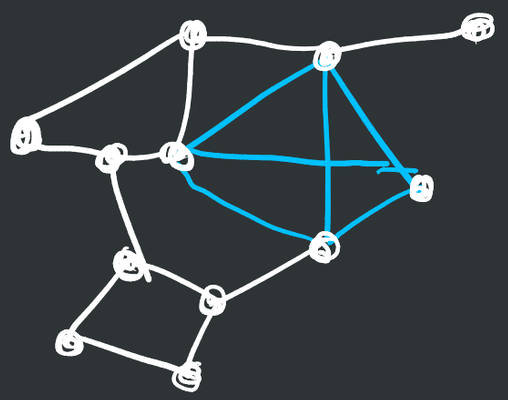
→ Entrée = $\bigwedge_{i=1}^{n} \bigvee_{j=1}^{3} x_{ij}^{*}$, $x_{ij}^{*} = x_{ij}$ ou $\neg x_{ij}$

→ Sortie = Existe-t-il une instanciation des x_{ij} à $\{\top, \bot\}$ tq la formule soit vraie ?

$[(\overset{\top}{x_0} \vee x_1 \vee \bar{x}_3] \wedge [(\overset{\top}{x_2} \vee \bar{x}_2 \vee x_1] \wedge [(\overset{\top}{x_0} \vee x_1 \vee x_3] \wedge [(\overset{\top}{x_0} \vee x_2 \vee x_1]$ ☺

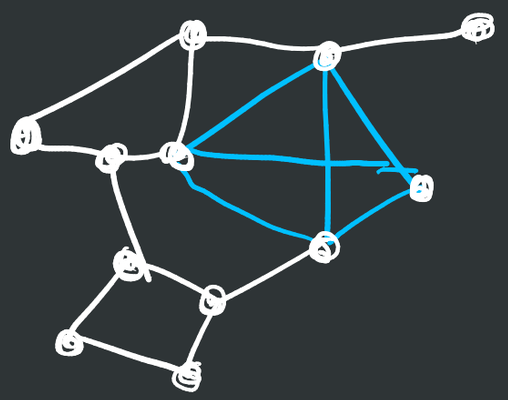
Problème CLIQUE : étant donnés | G un graphe
k ∈ ℕ

Y a-t-il une clique de taille k dans G ?



Problème CLIQUE : étant donnés | G un graphe
k ∈ ℕ

Y a-t-il une clique de taille k dans G ?

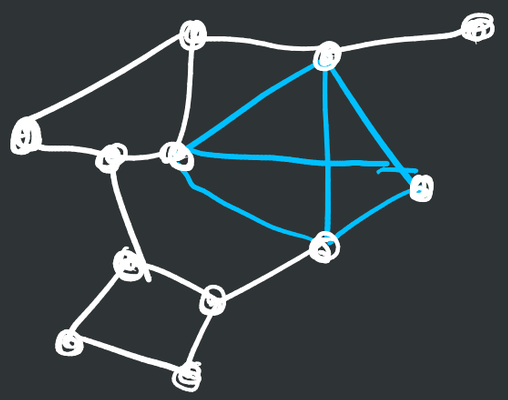


---

(1) CLIQUE est dans NP.

Problème CLIQUE : étant donnés | G un graphe
k ∈ ℕ

Y a-t-il une clique de taille k dans G ?



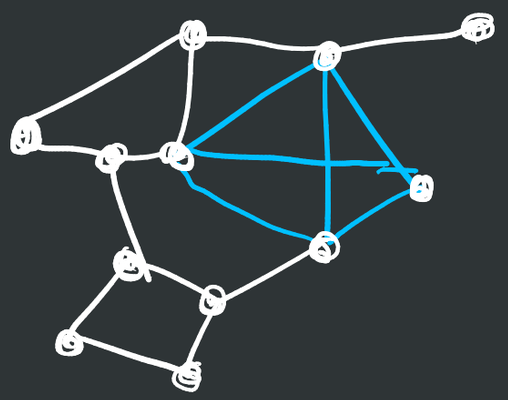
---

(1) CLIQUE est dans NP.

(2) CLIQUE est
plus dur que tout problème de NP.

Problème CLIQUE : étant donnés | G un graphe
k ∈ ℕ

Y a-t-il une clique de taille k dans G ?



---

(1) CLIQUE est dans NP.

(2) CLIQUE est plus dur que 3SAT qui est plus dur que tout problème de NP.

(2) CLIQUE est plus dur que 3SAT qui est plus dur que tout problème de NP.

Pour toute instance F de 3SAT $\longrightarrow$ Une instance (G, k) de CLIQUE

telle que F solution de 3SAT $\Leftrightarrow$ (G, k) solution de CLIQUE.

(2) CLIQUE est plus dur que 3SAT qui est plus dur que tout problème de NP.

Pour toute instance F de 3SAT $\longrightarrow$ Une instance (G, k) de CLIQUE

$$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

telle que F solution de 3SAT $\Leftrightarrow$ (G, k) solution de CLIQUE.

(2) CLIQUE est plus dur que 3SAT qui est plus dur que tout problème de NP.

Pour toute instance F de 3SAT $\longrightarrow$ Une instance (G, k) de CLIQUE

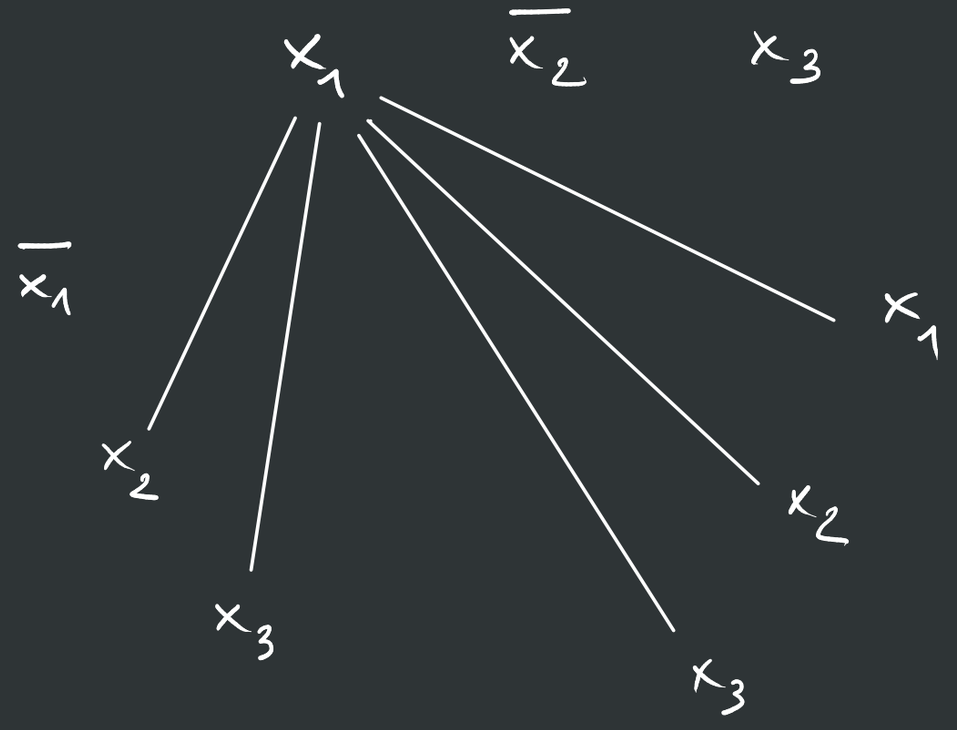
$$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

telle que F solution de 3SAT $\Leftrightarrow$ (G, k) solution de CLIQUE.

(2) CLIQUE est plus dur que 3SAT qui est plus dur que tout problème de NP.

Pour toute instance F de 3SAT $\longrightarrow$ Une instance (G, k) de CLIQUE

$$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$



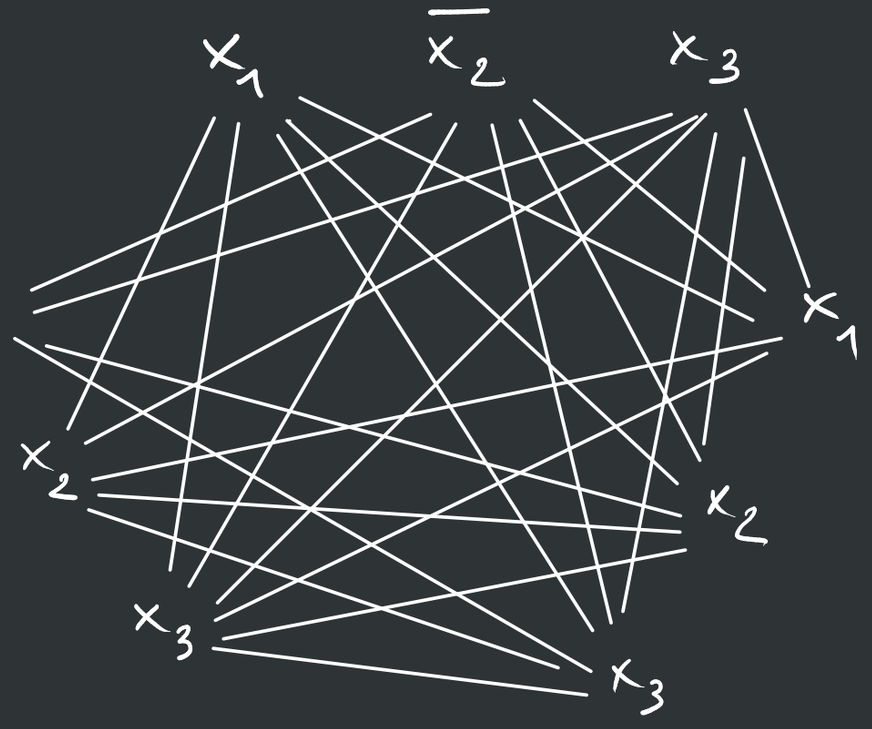
telle que F solution de 3SAT $\Leftrightarrow$ (G, k) solution de CLIQUE.

(2) CLIQUE est plus dur que 3SAT qui est plus dur que tout problème de NP.

Pour toute instance F de 3SAT $\longrightarrow$ Une instance (G, k) de CLIQUE

$$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

$G =$



$k = 3$

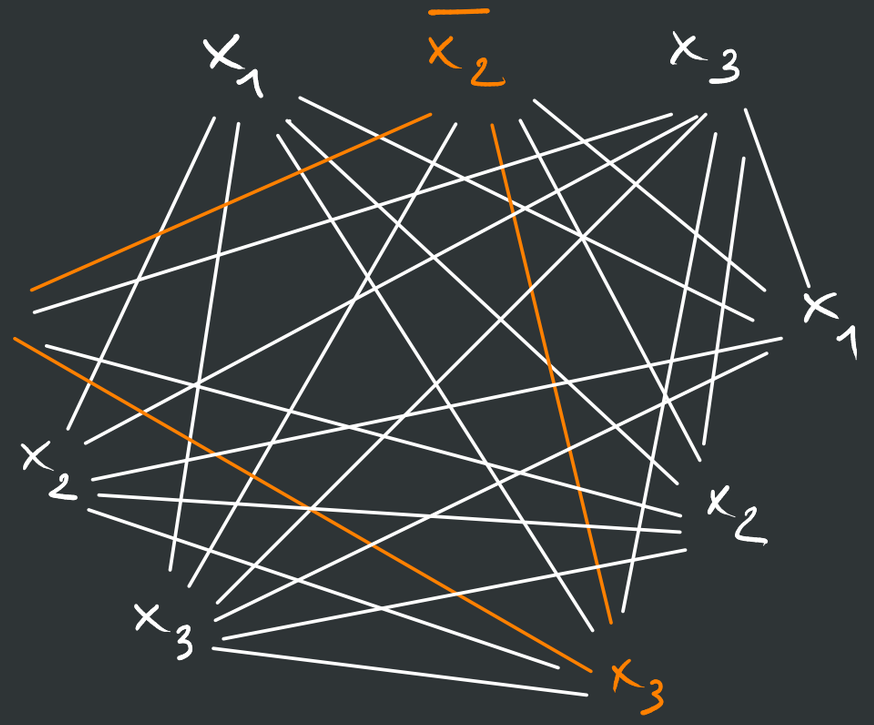
telle que F solution de 3SAT $\Leftrightarrow$ (G, k) solution de CLIQUE.

(2) CLIQUE est plus dur que 3SAT qui est plus dur que tout problème de NP.

Pour toute instance F de 3SAT $\longrightarrow$ Une instance (G, k) de CLIQUE

$$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

$G =$



$k = 3$

telle que F solution de 3SAT $\Leftrightarrow$ (G, k) solution de CLIQUE.

PLEIIIN de problèmes NP-complets

Hamiltonian cycle

Vertex cover

k-color

Max-cut

Traveling salesman problem

Knapsack problem

PLEIIIN de problèmes NP-complets

- Hamiltonian cycle
- Vertex cover
- k-color
- Max-cut
- Traveling salesman problem
- Knapsack problem

Existence d'une solution pour une équation diophantienne quadratique

$$\int_0^{2\pi} \prod_{i=1}^{n} \cos(a_i x)\, dx \neq 0$$

PLEIIIN de problèmes NP-complets

Hamiltonian cycle

Vertex cover

k-color

Max-cut

Traveling salesman problem

Knapsack problem

Optimal solution to Rubik's cube

Sudoku Tetris

Existence d'une solution pour une équation diophantienne quadratique

$$\int_0^{2\pi} \prod_{i=1}^{n} \cos(a_i x)\,dx \neq 0$$

NP-Completeness of Pandemic

KENICHIRO NAKAI[1] YASUHIKO TAKENAGA[1,a)]

Received: August 31, 2011, Accepted: December 16, 2011

Abstract: Pandemic is a multi-player board game which simulates the outbreak of epidemics and the human effort to prevent them. It is a characteristic of this game that all the players cooperate for a goal and they are not competitive. We show that the problem to decide if the player can win the generalized Pandemic from the given situation of the game is NP-complete.

Des questions ?