LOG8470
Méthodes formelles en fiabilité et sécurité

Préliminaires mathématiques

John Mullins

Dép. de génie informatique et de génie logiciel
École Polytechnique de Montréal

John.Mullins@polymtl.ca

2018 - 2019

POLYTECHNIQUE MONTRÉAL

Contenu

Contenu

Propositions et prédicats

- Une *proposition* est un énoncé qui possède une valeur de vérité.
- Un *prédicats* est un énoncé qui contient des *variables*
- Un prédicat devient une proposition lorsque les variables sont affectées.
- $P(x)$, $P(x, y)$, $P(x_1, x_2, \cdots x_n)$ dénotent des prédicats à 1, 2, n variables

Example

- $P(x) \equiv x \leq 3$
- $P(x, y) \equiv (x \leq 3) \wedge (y \leq 7)$
- $P(x) \equiv \exists y, x = y^2$

Les ensembles

Ensemble

Un ensemble est une collection d'objets appelés *éléments*.

- On note $a \in X$ l'appartenance d'un élément a à l'ensemble X.
- Définition en *extension* ou par *énumération* : $X = \{a, b, c\}$
- Définition en *compréhension* : $X = \{x : P(x)\}$: l'ensemble X est formé des éléments x pour lesquels le prédicat $P(x)$ est vrai.
- X est un *sous-ensemble* de Y ($X \subseteq Y$) ssi

$$x \in X \rightarrow x \in Y$$

- $X = Y$ ssi $X \subseteq Y \wedge Y \subseteq X$

Les principaux constructeurs

1. **[Ensemble des parties]** Si X est un ensemble alors l'ensemble des sous-ensembles de X

$$\mathcal{P}(X) = \{Y : Y \subseteq X\}$$

2. **[Ensemble indexé]** Si I est un ensemble et si à tout $i \in I$ est associé un unique élément x_i (qui peut être lui-même un ensemble) alors l'ensemble

$$\{x_i : i \in I\}$$

3. **[Union]** Si X et Y sont des ensembles alors

$$X \cup Y = \{a : a \in X \text{ ou } a \in Y\}$$

Les principaux constructeurs (suite)

1. **[Union unaire]** Si X est un ensemble d'ensembles alors
$$\bigcup X = \{a : \text{Il existe } x \in X \text{ tel que } a \in x\}$$
est un ensemble. Si X est indexé par I, on notera $\bigcup X$ par $\bigcup_{i \in I} x_i$.
2. **[Intersection]** Si X et Y sont des ensembles alors
$$X \cap Y = \{a : a \in X \text{ et } a \in Y\}$$
3. **[Intersection unaire]** Si X est un ensemble d'ensembles alors
$$\bigcap X = \{a : \text{Pour tout } x \in X,\ a \in x\}$$
est un ensemble. Si X est indexé par I, on notera $\bigcap X$ par $\bigcap_{i \in I} x_i$.
4. **[Produit]** Si X et Y sont des ensembles alors
$$X \times Y = \{(a, b) : a \in X \text{ et } b \in Y\}$$
est un ensemble. (a, b) est appelé *paire ordonnée*.
5. **[Différence]** Si X et Y sont des ensembles alors

Relations et fonctions

Relation binaire R

$R \subseteq X \times Y$

Une *fonction partielle* de X dans Y

relation $f \subseteq X \times Y$ telle que pour tout $x \in X$ et $y, y' \in Y$, si $(x, y) \in f$ et $(x, y') \in f$ alors $y = y'$.

Une *fonction (totale) f* de X dans Y (notée $f : X \to Y$)

si pour tout $x \in X$, il existe $y \in Y$ tel que $(x, y) \in f$. On écrira $x \mapsto y$, $y = f(x)$ ou $y = fx$.

Composition de $R \subseteq X \times Y$ et $S \subseteq Y \times Z$

$S \circ R = \{(x, z) \in X \times Z : \text{Il existe } y \in Y \text{ tel que } (x, y) \in R \text{ et } (y, z) \in S\}$

Les relations d'équivalence

Relation d'équivalence $R \subseteq X \times X$

- **[réflexive]** xRx, pour tout $x \in X$
- **[symétrique]** Si xRy alors yRx, pour tout $x, y \in X$
- **[transitive]** Si xRy et yRz alors xRz, pour tout $x, y, z \in X$.

- *classe d'équivalence*de x relativement à R :

$$[x]_R = \{y \in X : yRx\}$$

R induit sur X une partition

- Une partition d'un ensemble X est une famille $\{X_i\}$ de sous-ensembles de X disjoints entre eux qui recouvrent X.
- *quotient* de X par R :

$$X/R = \{[x]_R : x \in X\}$$

Contenu

Définitions inductives

Moyen élégant très utile en informatique

- Elle permet de construire effectivement des ensembles infinis d'objets
- Elle permet une technique de preuve, plus élégante que l'induction sur les entiers, pour prouver des propriétés requises à ces objets.
- Elle est utilisée pour définir les expressions arithmétiques, les expressions régulières, les piles, les files, les arbres, les programmes syntaxiquement valides, ... etc.

Definition

La définition inductive d'une partie X d'un ensemble consiste

- en la donnée explicite de certains éléments de X (bases)
- en la donnée d'une méthode de construction de nouveaux éléments de X à partir d'éléments déjà construits (étapes inductives)

Définitions inductives

Example (Arbres k-aires)

Base Un graphe formé d'un sommet (racine), est un arbre.

Induction Si $T_1, T_2, \ldots, T_k$ sont des arbres alors le graphe formé :

1. d'un nouveau sommet N,
2. de copies de $T_1, T_2, \ldots, T_k$,
3. de nouveaux arcs du sommet N à chacune des racines des arbres $T_1, T_2, \ldots, T_k$

est un arbre.

Example (Expressions arithmétiques)

Base : Un nombre ou une variable est une EA.

Induction : Si E et F sont des EA alors $E + F$, $E * F$ et (E) sont des EA

Définitions inductives

Definition

La définition inductive d'une partie X d'un ensemble U consiste

- en un sous-ensemble B de U (bases)
- en un ensemble K de fonctions partielles $f : U^{ar(f)} \rightarrow U$ où $ar(f)$ est l'arité de f (son nombre d'arguments) (étapes inductives)

X est alors est défini comme le plus petit ensemble vérifiant :

(B) $B \subseteq X$

(I) $\forall f \in K, \forall x_1, \ldots x_{ar(f)} \in X, f(x_1, \ldots, x_{ar(f)}) \in X$

Principe de preuve par induction

Théorème

Soit X un ensemble défini intuitivement par (B, K). Pour montrer

$$\forall x \in X, P(X)$$

où P est une propriété, il suffit de montrer :

Base pour tout $b \in B$, on a $P(b)$

Pas pour tout $f \in K$, pour tout $x_1, \ldots x_{ar(f)} \in U$, si $P(x_1), \ldots P(x_{ar(f)})$ sont vraies, alors $P(f(x_1, \ldots, x_{ar(f)}))$ est vraie

Principe de preuve par induction

Example (Tout arbre a exactement un sommet de plus que d'arcs)

Soit T un arbre à n sommets et e arcs et $P(T) \equiv n = e + 1$
Base : Si T est formé d'un seul sommet alors $n = 1$ et $e = 0$.
Pas d'induction : Soit T formé de la racine N et des sous-arbres $T_1, T_2, \ldots, T_k$. Supposons $P(T_i)$ pour chacun des sous-arbres T_i (pour $1 \leq i \leq k$) i.e. si T_i a n_i sommets et e_i arcs alors $n_i = e_i + 1$. On a :

$$\begin{aligned} n &= n_1 + n_2 + \cdots + n_k + 1 \\ &= (e_1 + 1) + (e_2 + 1) + \cdots (e_k + 1) + 1 \\ &= e_1 + e_2 + \cdots + e_k + k + 1 \\ &= e + 1 \end{aligned}$$

Principe de preuve par induction

Example (Toute EA est bien parenthésée)

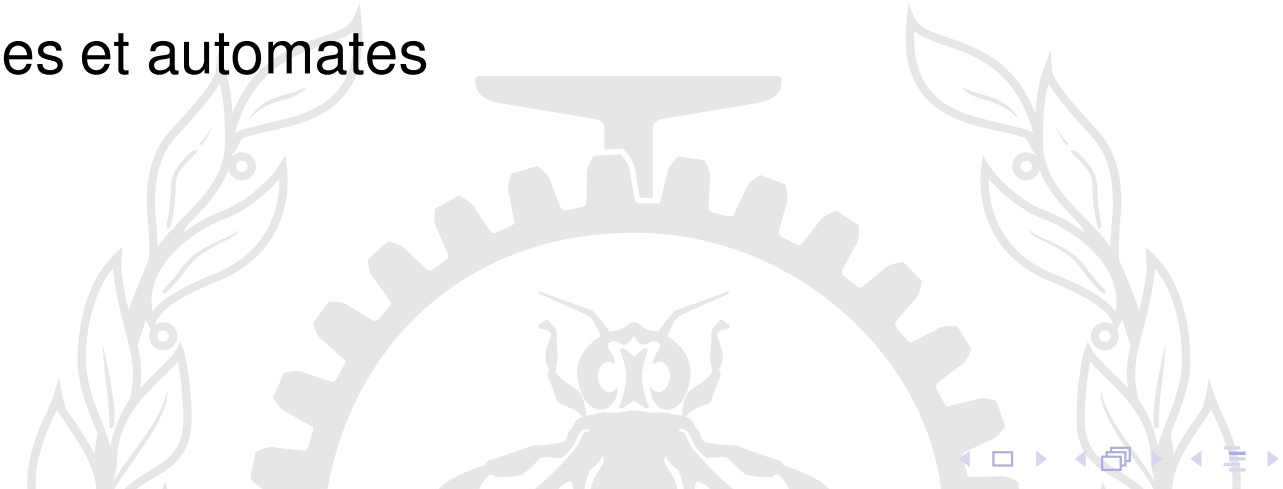
Soit $P(G) \equiv G$ est bien parenthèse
Base : Si G est une base, elle n'a pas de parenthèse
Induction : Il y a 3 constructeurs :

1. $G = E + F$.
2. $G = E * F$.
3. $G = (E)$.

On suppose (hypothèse d'induction) que $P(E)$ et $P(F)$ sont satisfaites.
Alors pour chacun des trois constructeurs de G :

1. Si $G = E + F$ alors G est bien parenthésée
2. Si $G = E * F$ alors G est bien parenthésée
3. Si $G = (E)$ alors G est bien parenthésée

Contenu



Mots et langages

Definition (Alphabet)

Un alphabet est un ensemble noté Σ dont les éléments sont appelés *lettres* ou *symboles*.

Definition (Mot)

Un *mot* sur Σ est une suite (ou chaîne) finie de lettres de Σ.

- Le mot *vide* est noté ϵ.
- La longueur d'un mot u est notée $|u|$. ϵ est le mot de longueur nulle.
- L'ensemble des mots finis sur Σ est noté Σ^*.

Mots et langages

Definition (Concaténation de mots)

Σ^* est muni d'une opération binaire, la *concaténation*. La concaténation du mot u avec le mot v et dénotée $u \cdot v$ ou simplement uv en omettant le $\cdot$, est le mot obtenu en ajoutant v à la suite de u. Cette opération est :

- *associative* et
- possède le mot vide comme *élément neutre*.

Mots et langages

Definition (Langage)

Une partie de Σ^* est appelée *langage* sur Σ.

Definition (Constructeurs de langages)

- La *concaténation* : $L \cdot L' = \{u \cdot u' : u \in L \wedge u' \in L'\}$

Remarque

- $L \cdot \emptyset = \emptyset \cdot L = \emptyset$
- $L \cdot \Sigma^* \neq \Sigma^* \neq \Sigma^* \cdot L$
- Si $\epsilon \in L$ alors $L \cdot \Sigma^* = \Sigma^* \cdot L = \Sigma^*$
- $L \cdot \{\epsilon\} = \{\epsilon\} \cdot L = L$

- La *fermeture de Kleene* : $L^* = \bigcup_{n \in \mathbb{N}} L^n$ où

$$L^n = \{u_1 u_2 \dots u_n : u_1, u_2, \dots u_n \in L\}$$

Mots et langages

Example

Soit les langages $L_1 = \{bb\}$ et $L_2 = \{\epsilon, bb, bbbb\}$. Les langages L_1^* et L_2^* sont formés, tous les deux, de tous les mots sur l'alphabet $\{b\}$ qui contiennent un nombre pairs de b.

Example

Le langage $L_1 = \{a, b\}^*\{bb\}\{a, b\}^*$ est formé de tous les mots sur l'alphabet $\{a, b\}$ qui contiennent le facteur bb.

Example

Le langage $\{aa, ab, ba, bb\}^*$ est formé de tous les mots sur l'alphabet $\{a, b\}$ de longueur paire : $\{a, b\}^* \setminus \{aa, ab, ba, bb\}^*$ est formé de tous les mots sur l'alphabet $\{a, b\}$ de longueur impaire. De façon alternative : $\{a, b\}\{aa, ab, ba, bb\}^*$

Mots et langages

Definition (Expressions régulières)

- ϵ, $\emptyset$ et a (pour tout $a \in \Sigma$) sont des ER
- Si u, v sont des ER alors $u \cdot v$, $u + v$ et u^* sont des ER

Definition (Sémantique des expressions régulières)

La sémantique des expressions régulières est donnée par l'application $L : ER \to \mathcal{P}(\Sigma^*)$ définie par

- $L(\epsilon) = \{\epsilon\}$ et $L(\emptyset) = \emptyset$
- Pour tout $a \in \Sigma$, $L(a) = \{a\}$
- Si u, v sont des ER alors $L(u \cdot v) = L(u) \cdot L(v)$, $L(u + v) = L(u) \cup L(v)$ et $L(u^*) = L(u)^*$

Definition (Langages réguliers)

L est régulier ssi il existe une ER u telle que $L(u) = L$

Automates d'états finis

Definition (Automates à états finis déterministe)

Un *automate à états finis déterministe* (AFD) est un quintuplet $(S, \Sigma, \delta, s_0, F)$ où

- S est un ensemble fini d'*états*
- Σ, un alphabet
- $\delta : S \times \Sigma \to S$ est appelée *fonction de transition* .
- s_0, un *état initial*
- Un ensemble F d'*états finaux ou acceptants* avec $F \subseteq S$

Automates d'états finis

Definition (Exécution d'un AFD)

On étend δ à une fonction $\Delta : S \times \Sigma^* \to S$:

- $\Delta(q, \epsilon) = q$
- $\Delta(q, xa) = \delta(\Delta(q, x), a)$.

Soit un mot $w = x_0 x_1 \ldots x_n$. La suite $s_0 q_1 \ldots q_{n+1}$ telle que

$$q_{i+1} = \Delta(s_0, x_0 x_1 \ldots x_i), \text{ pour tout } 0 \leq i \leq n$$

est appelé exécution de w.

Example (Exécution de 110101 d'un AFD qui reconnait $\Sigma^* 01 \Sigma^*$)

Automates d'états finis

Definition (Langage reconnu par un AFD)

Le langage $L(\mathcal{A})$ défini par :

$$L(\mathcal{A}) = \{w \in \Sigma^* : \Delta(s_0, w) \in F\}$$

est appelé le langage reconnu par $\mathcal{A}$.

Example (AFD qui reconnait $\Sigma^*01\Sigma^*$)

Automates d'états finis

Definition (Automates à états finis non-déterministe)

Un *automate à états finis non-déterministe* (AFN) est un quintuplet $(S, \Sigma, \delta, s_0, F)$ où

- S est un ensemble fini d'*états*
- Σ, un alphabet
- $\delta : S \times \Sigma \to \mathcal{P}(S)$ est appelée *fonction de transition* .
- s_0, un *état initial*
- Un ensemble F d'*états finaux ou acceptants* avec $F \subseteq S$

Automates d'états finis

Definition (Exécution d'un AFN)

On étend δ à une fonction $\Delta : S \times \Sigma^* \to \mathcal{P}(S)$:

- $\Delta(q, \epsilon) = \{q\}$
- Soit $w = xa$ où $x \in \Sigma^*$ et $a \in \Sigma$ alors

$$\Delta(q, w) = \{r_1, r_2, \ldots, r_m\}$$

où $\Delta(q, x) = \{p_1, p_2, \ldots, p_k\}$ et $\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, r_2, \ldots, r_m\}$.

Soit un mot $w = x_0 x_1 \ldots x_n$. Une suite $s_0 q_1 \ldots q_{n+1}$ telle que

$$q_{i+1} \in \Delta(s_0, x_0 x_1 \ldots x_i), \text{ pour tout } 0 \leq i \leq n$$

est appelé exécution de w.

Example (Exécution de 110101 d'un AFN qui reconnait $\Sigma^* 01 \Sigma^*$)

Automates d'états finis

Definition (Langage reconnu par un AFN)

Le langage $L(\mathcal{A})$ défini par :

$$L(\mathcal{A}) = \{w \in \Sigma^* : \Delta(s_0, w) \cap F \neq \emptyset\}$$

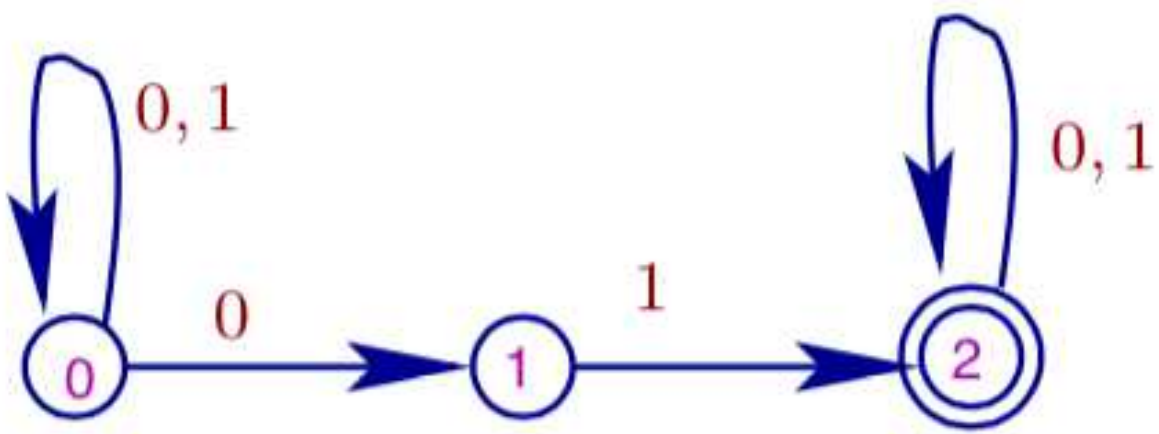
est appelé le langage reconnu par $\mathcal{A}$.

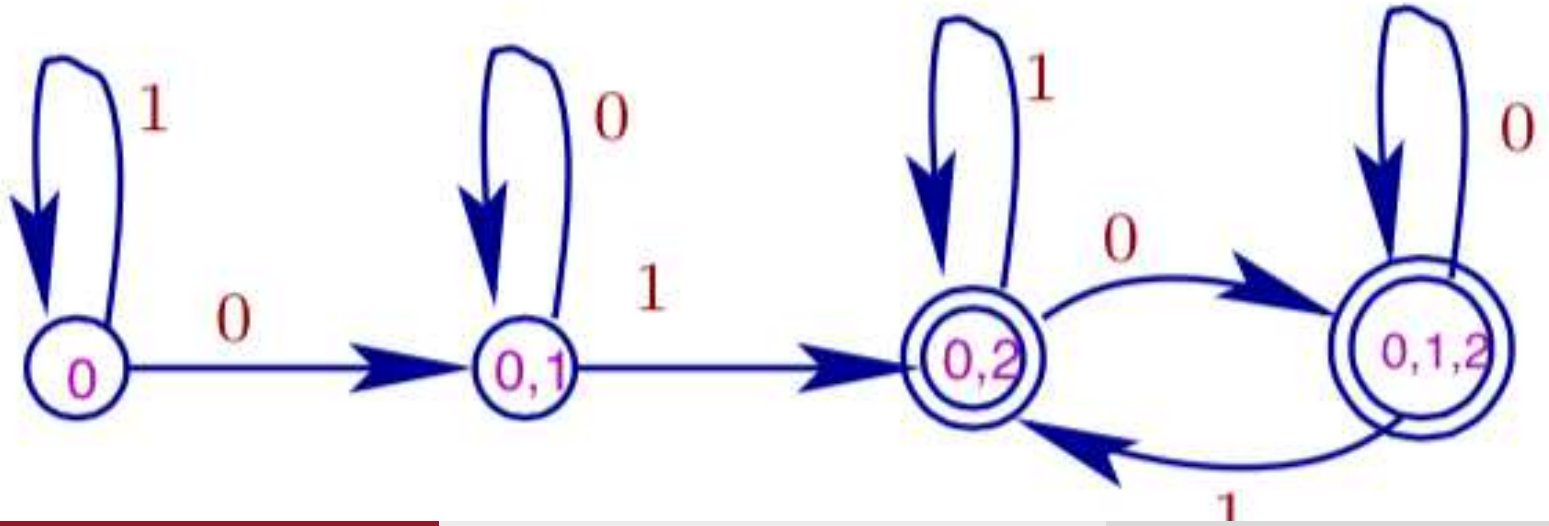
Example (AFN qui reconnait $\Sigma^*01\Sigma^*$)

Theorem (Kleene)

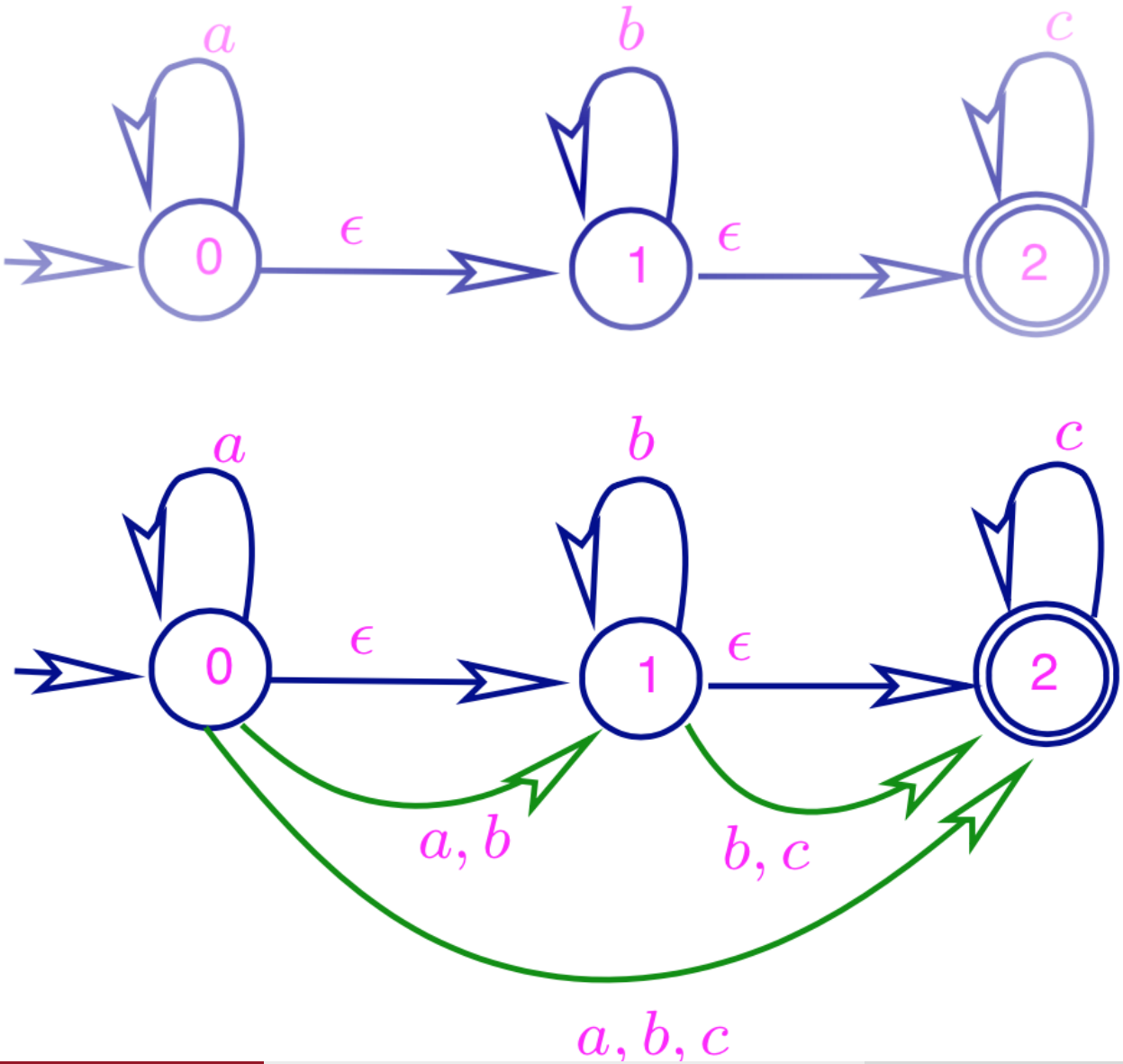
Un langage est reconnaissable par un AFD si et seulement s'il est régulier.

Déterminisation (Rabin-Scott, 1959)





Élimination des ϵ-transitions



Problèmes de décisions

D'autres propriétés de fermeture

1. Intersection
2. Complémentation

Problème du vide

Étant donné un automate fini $\mathcal{A}$ on on peut décider si $L(\mathcal{A}) = \emptyset$

Problème d'inclusion

Étant donné des automate fini $\mathcal{A}$ et $\mathcal{B}$ on on peut décider si $L(\mathcal{A}) \subseteq L(\mathcal{B})$