Informatique Théorique 3
2004/05

Semaine 2

Répétition... Quiz...

http://fr.wikipedia.org/wiki/Algèbre_abstraite

http://fr.wikipedia.org/wiki/Relation_(mathématiques)
http://fr.wikipedia.org/wiki/Relation_binaire
http://fr.wikipedia.org/wiki/Relation_d'ordre
...

http://fr.wikipedia.org/wiki/Argument_de_la_diagonale_de_Cantor

Spécifier un langage

- par prédicat logique (plus ou moins formel)
  $\{ a \mid P(a) \}$
- composition ensembliste de langages existants
  $L_1 \cup L_2$ ...
- énumération des mots ...
  difficile pour des langages de taille infinie
- *utiliser un mécanisme de* ***production*** *des mots*

  Les Grammaires !

Opérations sur les langages

1.2.2 Définition (Opération sur les langages) Soit Σ un alphabet fini, L et L' deux langages sur Σ^*.

Concaténation. La concaténation LL' de L et L' est définie par $\boxed{LL' \triangleq \{ww' \mid w \in L \wedge w' \in L'\}}$.

Itération. La n^e itération $\boxed{L^n}$ d'un language n est définie de manière inductive par,

$$\boxed{L^0 \triangleq \{\epsilon\} \qquad L^{n+1} \triangleq LL^n}$$

Fermeture itérative. La fermeture itérative (ou de Kleene) L^* de L est définie par

$$\boxed{L^* \triangleq \bigcup_{n \in \mathbb{N}} L^n}$$

On utilise aussi la notation suivante, $\boxed{L^+ \triangleq LL^*}$.

1.2.3 Lemme

1. Propriétés de la fermeture de Kleene.

$$\boxed{L^*L^* = L^*} \quad \boxed{L^{**} = L^*} \quad \boxed{\emptyset^* = \{\epsilon\}}$$

Grammaires (I)

1.4.1 Définition (Grammaire générative)
Une grammaire est un quadruplet $\boxed{G \triangleq (V, \Sigma, P, S)}$.

- V est un alphabet non vide de symboles *non-terminaux*.
- Σ est un alphabet non vide de symboles *terminaux* disjoint de V ($V \cap \Sigma = \emptyset$).
- $\boxed{P \subseteq (\Gamma^+ \times \Gamma^*)}$ (avec $\Gamma \triangleq V \cup \Sigma$) est un ensemble fini de *productions.*
- $S \in V$ est le *symbole de départ*. $\square$

Nous utilisons des lettres grecques minuscules $\boxed{\alpha, \beta, \ldots}$ pour les éléments de $(V \cup \Sigma)^*$. Pour spécifier une production $(\alpha, \beta) \in P$, on note $\alpha \to \beta$ ou encore $\alpha \to_G \beta$.

Grammaires (II)

1.4.2 Définition (Dérivation directe) Soit $G \triangleq (V, \Sigma, P, S)$ une grammaire. La relation de *dérivation directe* dans G entre deux mots $\alpha, \beta \in (V \cup \Sigma)^*$ est définie par,

$$\boxed{(\alpha \Rightarrow_G \beta)} \quad \Leftrightarrow \quad \boxed{\exists \alpha', \beta', \gamma, \gamma' \in (V \cup \Sigma)^* : \begin{cases} \alpha' \rightarrow_G \beta' \\ \alpha = \gamma\alpha'\gamma' \\ \beta = \gamma\beta'\gamma' \end{cases}}$$

On dit alors que β est *directement dérivable* de α dans G. □

1.4.3 Définition (Dérivation) Soit $G \triangleq (V, \Sigma, P, S)$ une grammaire. On définit la relation de dérivation $\boxed{\Rightarrow_G^*}$ comme étant la fermeture réflexive et transitive de $\Rightarrow_G$.

Une *dérivation* de α_0 en α_n dans G est une séquence finie $\boxed{(\alpha_i)_{i \in [0,n]}}$ d'éléments de $(V \cup \Sigma)^*$ telle que $\forall i \in [0, n-1] : \alpha_i \Rightarrow_G \alpha_{i+1}$. On note généralement cette séquence $\boxed{\alpha_o \Rightarrow_G \alpha_1 \Rightarrow_G \ldots \Rightarrow_G \alpha_n}$. □

Grammaires (III)

1.4.4 Définition (Langage d'une grammaire) Soit $G = (V, \Sigma, P, S)$ une grammaire.

1. Un mot $\boxed{v \in \Sigma^*}$ est généré par G ssi $\boxed{S \Rightarrow_G^* v}$. Dans ce cas on dit que v est un mot *terminal* de la grammaire.
2. Le langage généré par G est défini par :

$$\boxed{\mathrm{L}(G) \triangleq \{\, v \in \Sigma^* \mid S \Rightarrow_G^* v \,\}}$$

Alors, on peut ***énumérer/produire des mots du langage d'une grammaire*** par application de toutes ses règles de manière systématique et exhaustive.

Types de grammaires (I)

1.4.5 Définition (Type d'une grammaire) Soit $G = (V, \Sigma, P, S)$ une grammaire. Un grammaire est dite de

Type 0 dans tous les cas.

Type 1 si pour toute production $\alpha \rightarrow_G \beta \in P$ on a,

1. $|\alpha| \leq |\beta|$
2. ou $\alpha = S \wedge \beta = \epsilon$.

La grammaire G est dite *contextuelle*.

Type 2 si pour toute production $\alpha \rightarrow_G \beta \in P$ on a $\alpha \in V$. La grammaire est dite *libre de contexte* ou *non-contextuelle*.

Type 3 si toutes les production $\alpha \rightarrow_G \beta \in P$ sont de la forme,

1. $A \rightarrow_G wB$
2. $A \rightarrow_G w$

avec $A, B \in V$ et $w \in \Sigma^*$. La grammaire est dite *régulière*.

Types de grammaires (II)

1.4.6 Définition L'ensemble des langages générés par les grammaires d'un type i est donné par :

$$\boxed{\mathcal{L}_i = \{\mathrm{L}(G) \mid G \text{ est de type } i\}}.$$

Un type de grammaire i est *inclus* dans un type j si $\mathcal{L}_i \subseteq \mathcal{L}_j$. □

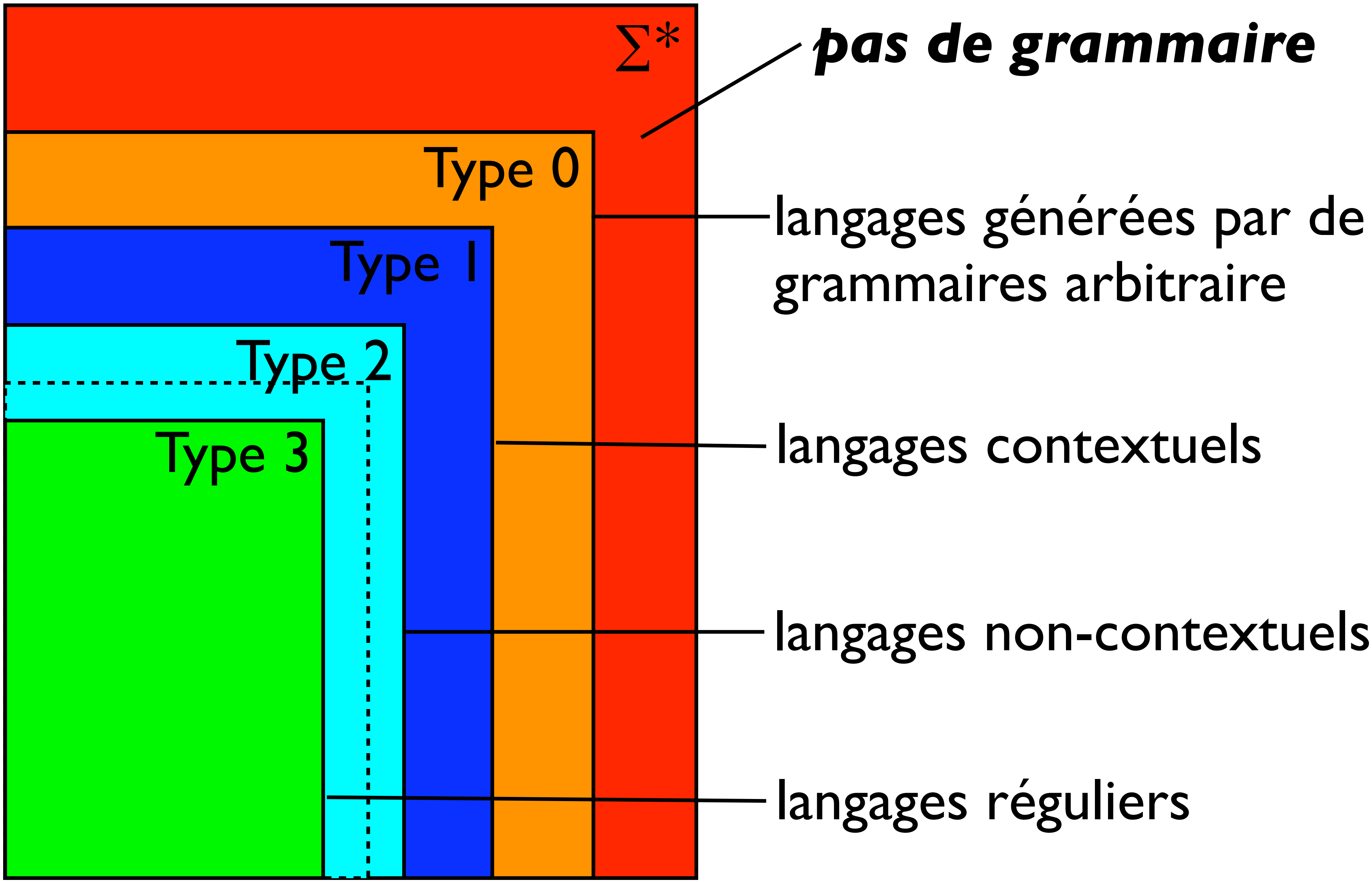
1.4.7 Définition (Type d'un langage) Un langage L est de type i si $L \in \mathcal{L}_i$ et $\boxed{L \text{ est } \textit{strictement} \text{ de type } i \text{ si } L \in \mathcal{L}_i \setminus \mathcal{L}_{i+1})}$, c'est à dire si L est généré par une grammaire de type i et par aucune grammaire de type $i+1$.

1.4.8 Théorème (Hiérarchie de Chomsky)
La relation entre les types de langages est :

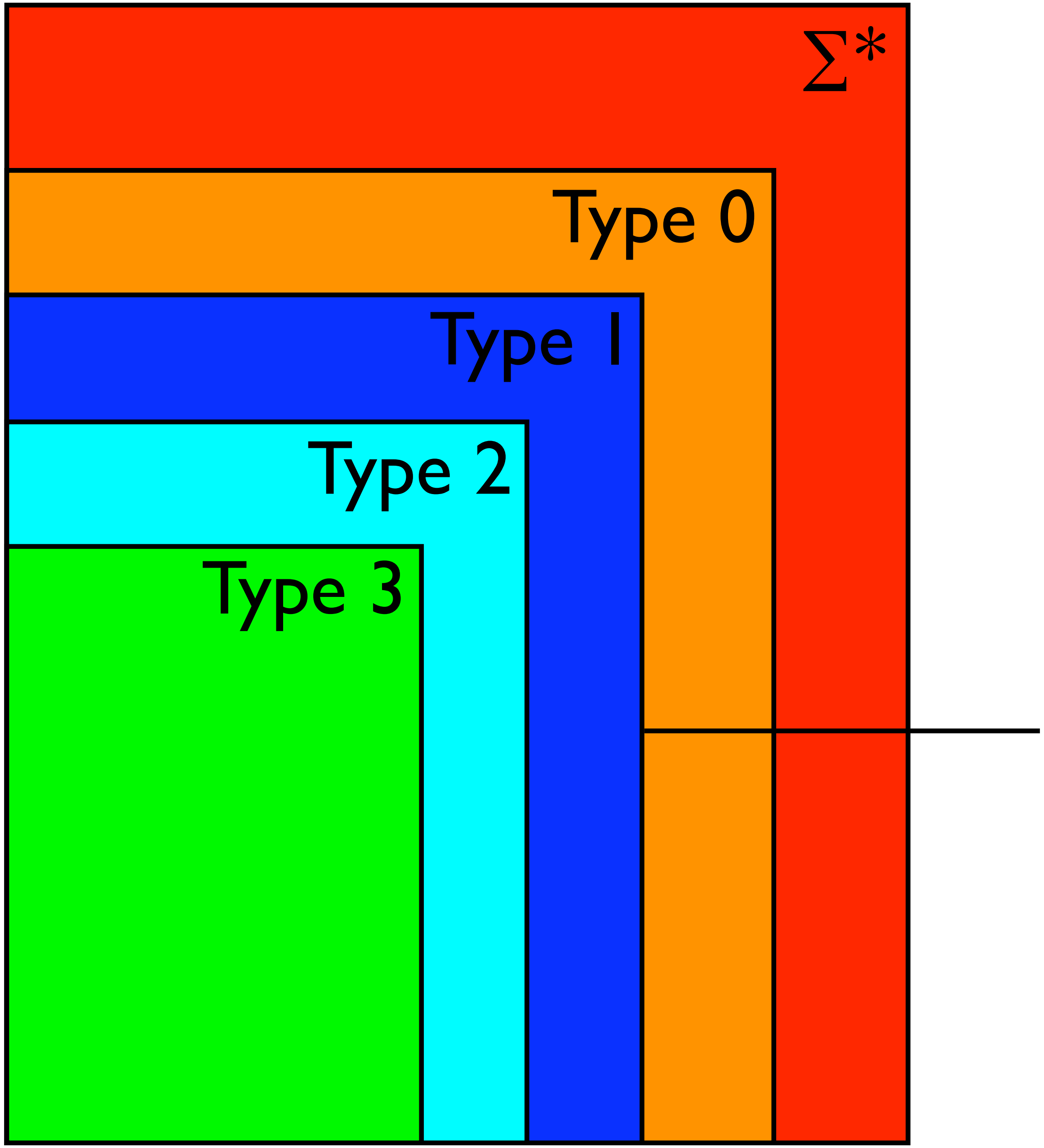
$$\mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1 \subset \mathcal{L}_0$$

Les inclusions sont strictes.

Hiérarchie de Chomsky (I)



Hiérarchie de Chomsky (III)



Décider les « problèmes de type 1 »

T_n^m « mots de long. ≤n dérivable en moins de m étapes »

$$\begin{array}{lcl} T_n^0 & = & \{ S \} \\ T_n^{m+1} & = & \mathsf{Drv}_n(T_n^m) \\ \mathsf{Drv}_n(X) & = & X \cup \{ \alpha' \in \Gamma^* \mid \boxed{\exists \alpha \in X \,.\, \alpha \Rightarrow_G \alpha'} \wedge \boxed{|\alpha'| \leq n} \} \end{array}$$

Type 1 si pour toute production $\alpha \rightarrow_G \beta \in P$ on a,

1. $|\alpha| \leq |\beta|$
2. ou $\alpha = S \wedge \beta = \epsilon$.

```
INPUT(G, x);
n := |x|;
T := S;
REPEAT
      T' := T;
      T := Drv_n(T');
UNTIL (x ∈ T)  OR  (T = T');
IF   x ∈ T
     THEN WriteString (" x element of L(G) ")
     ELSE WriteString (" x not element of L(G) ")
END
```

Automates finis déterministes (AFD)

2.1.1 Définition Un *automate fini déterministe (AFD)* est un quintuplet

$$M = (Q, \Sigma, \delta, s, F)$$

où

- Q est un ensemble fini d'*états*,
- Σ est un ensemble fini de symboles, *l'alphabet (d'entrée)*,
- $\boxed{\delta : Q \times \Sigma \to Q}$ est la *fonction (totale) de transition*,
- $s \in Q$ est l'*état initial* (ou de *départ*),
- $F \subseteq Q$ est un sous-ensemble de Q, les *états accepteurs* (ou *finaux*).

Fonctionnement des AFDs

2.1.3 Définition (Fonction de transition sur les mots) Étant donné un AFD $(Q, \Sigma, \delta, s, F)$, on étend la fonction de transition δ en une fonction de transition $\boxed{\widehat{\delta} : Q \times \Sigma^* \to Q}$ sur les mots définie par :

$$\boxed{\begin{array}{rcl} \widehat{\delta}(q, \epsilon) & \triangleq & q \\ \boxed{\widehat{\delta}(q, wa)} & \triangleq & \delta(\boxed{\widehat{\delta}(q, w)}, a) \end{array}}$$

2.1.4 Définition (Langage accepté par un automate) Soit $M = (Q, \Sigma, \delta, s, F)$ un AFD.

1. M *accepte* le mot $w \in \Sigma^*$ ssi $\boxed{\widehat{\delta}(s, w) \in F}$. Dans le cas contraire on dit que M *rejette* w.
2. Le *langage accepté par* M est défini par :

$$\boxed{\mathrm{L}(M) \triangleq \{\, w \in \Sigma^* \mid \widehat{\delta}(s, w) \in F \,\}}$$

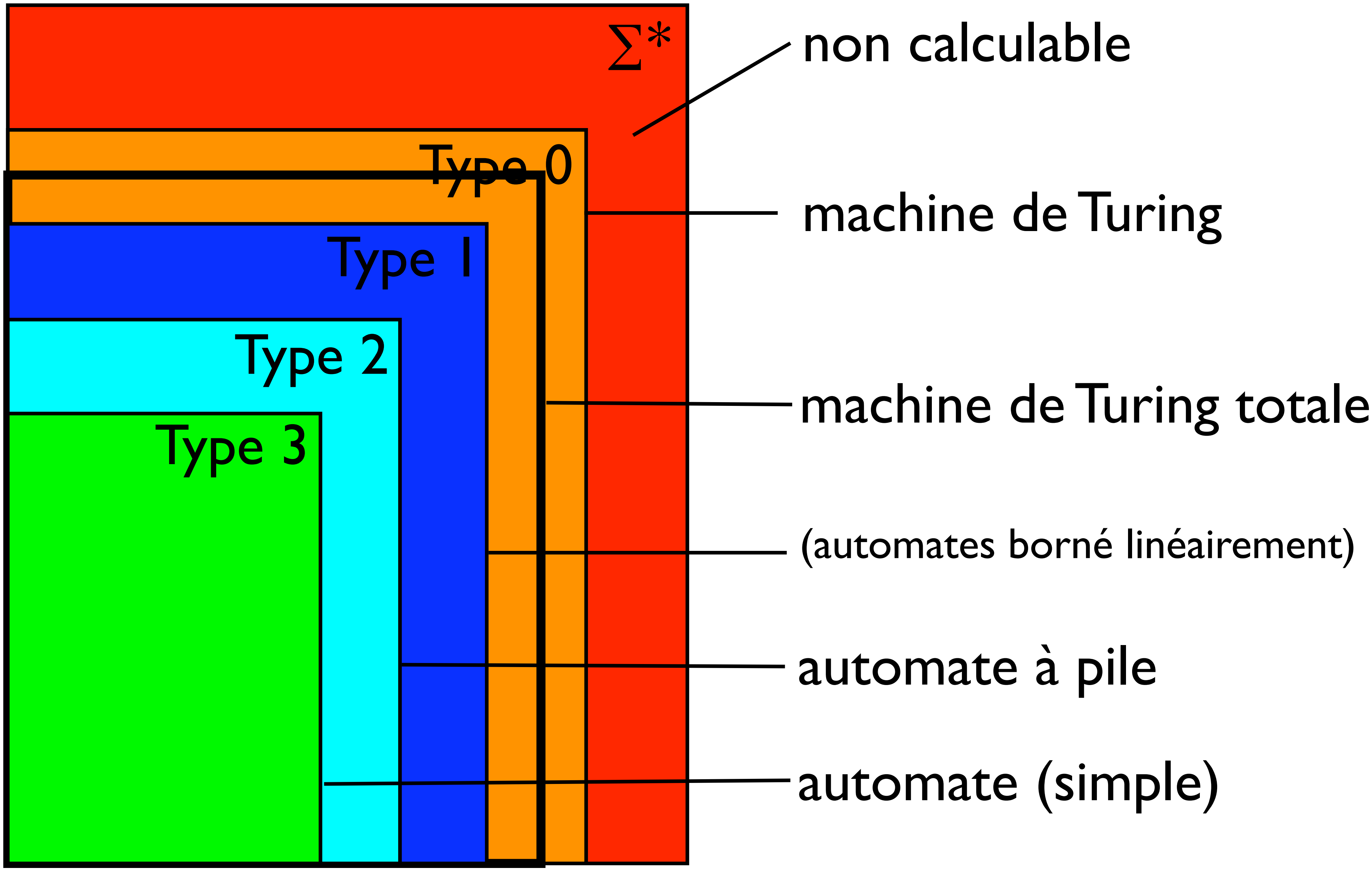
Langages réguliers

2.1.5 Définition (Langage régulier) Un langage $L \subseteq \Sigma^*$ est dit *régulier* s'il existe un automate fini M tel que $\mathrm{L}(M) = L$.

2.1.6 Théorème Soit L un langage sur Σ. Les deux propositions suivantes sont équivalentes :

1. Il existe un AFD $M = (Q, \Sigma, \delta, s, F)$ avec $\mathrm{L}(M) = L$.
2. Il existe une grammaire régulière $G = (V, \Sigma, P, S)$ avec $\mathrm{L}(G) = L$.

Hiérarchie de Chomsky (III)



Problèmes de décision (I)

1.5.2 Définition (Problèmes de décision) Un *problème de décision (binaire)* est un ensemble $\boxed{P = P^{\oplus} \cup P^{\ominus}}$ constitué de deux parties disjointes ($P^{\oplus} \cap P^{\ominus} = \emptyset$), les *instances positives* $P^{\oplus}$ et *négatives* $P^{\ominus}$ du problème.

En informatique théorique, on conjecture que tout problème peut être représenté par (encodé dans) un langage sur un alphabet donné.

1.5.3 Thèse Pour tout problème P, il existe un *encodage* de P.

1.5.4 Définition (Problème de décision encodés) Un problème binaire P est dit *encodé sur un alphabet* Σ si $P \subseteq \Sigma^*$ (souvent $P = \Sigma^*$).

Noter que dans ce cas, par $P = P^{\oplus} \cup P^{\ominus}$, les ensembles $P^{\oplus}$ et $P^{\ominus}$ des instances positives et négatives sont représentés par des *langages* sur Σ.

Fonctions (semi-) caractéristiques

1.5.1 Définition (Fonctions caractéristiques) Soit A et B deux ensembles tels que $\boxed{A \subseteq B}$.

1. La *fonction caractéristique* χ_A^B de A par rapport à B est définie par :

$$\boxed{\begin{array}{rccl} \chi_A^B : & B & \to & \{0,1\} \\ & b & \mapsto & \begin{cases} 1 & \text{si } b \in A \\ 0 & \text{si } b \notin A \end{cases} \end{array}}$$

2. La *fonction semi-caractéristique* ${\chi'}_A^B$ de A par rapport à B est définie par :

$$\boxed{\begin{array}{rccl} {\chi'}_A^B : & B & \rightharpoonup & \{0,1\} \\ & b & \mapsto & 1 \quad \text{si } b \in A \end{array}}$$

Lorsque l'ensemble B dont on parle est évident (dans le contexte), on ne le mentionne pas explicitement et on écrit simplement $\boxed{\chi_A}$. Noter que, par définition, une fonction caractéristique est totale.

Problèmes de décision (II)

1.5.5 Définition (Décidabilité) Soit $P = P^{\oplus} \cup P^{\ominus}$ un problème encodé sur Σ. Soit $L \triangleq P^{\oplus}$. P est *décidable* si la fonction caractéristique χ_L^P de L par rapport à P est calculable. P est *semi-décidable* si la fonction semi-caractéristique ${\chi'}_L^P$ de L par rapport à P est calculable.

$$\begin{array}{rccl} \chi_L : & P & \longrightarrow & \{0,1\} \\ & w & \longmapsto & \begin{cases} 1 & \text{si } w \in L \\ 0 & \text{si } w \notin L \end{cases} \end{array}$$

$$\begin{array}{rccl} \chi'_L : & P & \rightharpoonup & \{0,1\} \\ & w & \longmapsto & 1 \quad \text{si } w \in L \end{array}$$

□

Calculabilité

A la place de regarder toutes les fonctions de l'univers,

nous limitons notre recherche à des fonctions
caractéristiques pour des problèmes binaire

bref,

à la question de calculabilité
de la reconnaissance des langages sur des alphabets finis.

Déjà là, les limites fondamentales apparaissent.

Hiérarchie de Chomsky (II)