Master I d'informatique
INFO0705 : info. théorique
Pascal Mignot

Université de Reims Champagne-Ardenne

Aide-mémoire (Examen terminal)

Lorsque vous utilisez un résultat de cette feuille dans votre copie, il est conseillé de mettre la référence de ce résultat en utilisant le numéro de la section, et le numéro du résultat dans la section. Par exemple, le théorème de Rice a pour référence **D-18**.

A Langages réguliers

1. Un **alphabet** Σ est un ensemble **fini** de lettres. Σ^* est l'ensemble de toutes les chaines possibles pouvant être générées par l'alphabet Σ.
2. Un **langage** L sur l'alphabet Σ est un sous-ensemble de Σ^* (i.e. $L \subseteq \Sigma^*$).
3. Un automate déterministe fini (ADF) est un 5-uple $(Q, \Sigma, \delta, q_0, F)$, où Q est un ensemble fini d'états(ensemble des états), Σ est un ensemble fini de symboles (alphabet), $\delta : Q \times \Sigma \to Q$ est la fonction de transition, $q_0 \in Q$ est l'état de départ, $F \subset Q$ est l'ensemble des états acceptants.
4. **Langage accepté par un ADF** M est l'ensemble des chaines L acceptées par M.
5. **Langage régulier** est un langage accepté par un ADF M.
6. **Opérations régulières** : les opérations régulières sont l'**Union** $A \cup B = \{x \mid x \in A \text{ ou } x \in B\}$, la **Concaténation** $A \circ B = \{xy \mid x \in A \text{ et } y \in B\}$, et l'**Etoile** $A^* = \{x_1 x_2 ... x_k \mid k \geq 0 \text{ et } x_i \in A \text{ pour tout } i\}$ où A et B sont deux langages.
7. **Théorème** : L'ensemble des langages réguliers est fermé par toute opération régulière.
8. **L'union de deux ADFs** $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ et $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ donne un ADF $M = (Q, \Sigma, \delta, q_0, F)$ avec pour états $Q = Q_1 \times Q_2$, comme fonction de transition $\forall (r_1, r_2) \in Q$, $a \in \Sigma$, $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$, comme état initial $q_0 = (q_1, q_2)$ et comme états finaux $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$.
9. **Théorème** : l'ensemble des langages libres de contexte est fermé par les opérateurs d'intersection et de complémentarité (intersection : union avec $F = F_1 \times F_2$, complément : $\bar{F} = Q \backslash F$).
10. Un automate non-déterministe fini (ANF) est similaire à un ADF, excepté que sa fonction de transition est définie comme $\delta : Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$.
11. **Théorème** : un langage est régulier si et seulement si il est reconnu par un ANF.
12. Une **expression régulière** R (notée ER) sur un alphabet σ si R est un symbole $a \in \Sigma$, la chaîne vide ϵ, l'ensemble vide ϵ, une union d'expression régulière, une concaténation d'expression régulière, ou une opération étoile sur une expression régulières.
13. **Théorème** : un langage est régulier si et seulement si il est reconnu par un ER.
14. **Théorème** : si un langage est fini ($\#L$ est fini), alors il est régulier.
15. Deux mots u et v sont **L-équivalents** si $\forall z \in \Sigma^*, uz \in L \Leftrightarrow vz \in L$.
16. **Théorème** (Myhill-Nerode) : un langage L est régulier **si et seulement si** il existe un nombre fini de L-classes distinctes.
17. **Lemme de l'étoile** : soit un AFD $M = (Q, \Sigma, \delta, q_0, F)$ tel que $\mathcal{L}(M)$ soit infini. Pour tout mot w dans $\mathcal{L}(M)$ tel que $|w| \geq |Q|$, il existe une décomposition $w = xyz$ telle que $xy^i z \in \mathcal{L}(M)$ pour tout $i \geq 0$, $|y| > 0$ et $|xy| < |Q|$.

B Grammaire libre de contexte

1. Une **grammaire libre de contexte** (GLC) est un triplet (V, Σ, R, S) constitué d'un ensemble de variables V, d'un ensemble de symboles (terminaux) Σ, d'un ensemble de règle de réécriture sous

la forme $u \to U$ où U est une combinaison de variables de V ou de symboles de Σ, S est la variable de départ.

2. **Dérivations** : On écrit $u \Rightarrow v$ s'il existe une règle de réécriture $r \in R$ qui transforme u en v. On écrit $u \overset{*}{\Rightarrow} v$ s'il existe une suite de règles de réécriture $r_1 r_2 \dots r_k \in R^k$ qui transforme u en v.
3. Un mot w est généré par une GLC si $S \overset{*}{\Rightarrow} w$.
4. Un **langage libre de contexte** (LLC) est l'ensemble des mots de Σ^* générés par une GLC.
5. L'**union de 2 GLCs** $G_1 = (V_1, \Sigma, R_1, S_1)$ et $G_2 = (V_2, \Sigma, R_2, S_2)$ (où V_1 et V_2 sont disjoints) est une GLC $G = (V, \Sigma, R, S)$ telle que $V = V_1 \cup V_2$ et $R = R_1 \cup R_2 \cup S \to S_1 | S_2$.
6. Un grammaire linéaire (V, Σ, R, S) est une grammaire construite à partir d'un ADF $(Q, \Sigma, \delta, q_0, F)$, où l'on associe ① à chaque symbole $q_i \in Q$, une variable $v_i \in V$ ② à chaque transition $\delta(q_i, a) = q_j$, une règle $V_i \Rightarrow aV_j \in R$ ③ à chaque état final $q_i \in F$, une règle $V_i \to \epsilon$, ④ $S = V_0$ à l'état de départ q_0.
7. L'ensemble des langages libres de contexte est l'ensemble des langages acceptés par les GLCs.
8. **Théorème** : L'ensemble des langages libres de contexte est fermé par toute opération régulière.
9. **Théorème** : Si un langage est régulier, alors il est accepté par une GLC.
10. La **dérivation la plus à gauche** d'une chaîne consiste à chaque étape à remplacer la variable la plus à gauche.
11. Une chaîne générée par une GLC est **ambigüe** si cette GLC permet de dériver cette chaine de plus d'une façon en n'utilisant que des dérivations les plus à gauche.
12. Une grammaire est ambigüe si elle génère au moins une chaine ambigüe. Sinon, elle est non ambigüe.
13. Une GLC est sous **forme normale de Chomsky** si chaque règle est sous la forme $A \to BC$ ou $A \to a$ ou $S \to \epsilon$ où ① a est n'importe quelle terminal, ② A, B, C, S sont des variables telles que S est la variable de départ, A est n'importe quelle variable (y compris S) et B,C sont des variables différentes de S.
14. **Théorème** : tout LLC peut être généré par une GLC sous FNC en appliquant les étapes suivantes : ① ajouter une nouvelle règle $S_0 \to S$ ② supprimer les règles de la forme $A \to \epsilon$ (partout où A apparaît dans la RHS, ajouter une nouvelle règle sans A) ③ supprimer les règles unitaires $A \to B$ (pour toute règle $B \to U$, ajouter une règle $A \to U$) ④ supprimer les chaines $A \to u_1 u_2 \dots u_k$ (remplacer par les règles $A \to u_1 A_1$, $A_1 \to u_2 A_2$, $\dots A_{k-2} \to u_{k-1} u_k$ ⑤ supprimer les règles $A \to u_1 u_2$ où u_1 est un terminal (remplacer par $A \to U_1 u_2$ et $U_1 \to u_1$). Idem si u_2 est un terminal.
15. Un **automate à pile** (AP) est un 6-uple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ où Q un ensemble fini d'états, Σ un alphabet, Γ les symboles de la pile, δ une fonction de transition $Q \times \Sigma' \times \Gamma' \to \mathcal{P}(Q \times \Gamma')$ avec $\Sigma' = \Sigma \cup \{\epsilon\}$ et $\Gamma' = \Gamma \cup \{\epsilon\}$, q_0 est l'état de départ, F est l'ensemble des états acceptants.
16. **Théorème** : un langage est libre de contexte si et seulement si il est généré par un AP.
17. **Lemme de l'étoile** (pour une GLC) : soit un langage L généré par une GLC. Pour tous les mots w de L de longueur au moins p (dépendante du langage L), alors on peut trouver u, v, x, y, z tel que $w = uvxyz$, $|vy| > 0$, $|vxy| \leq p$, et tel que $\forall i \geq 0, uv^i xy^i z \in L$.
18. **Théorème** : l'ensemble des langages libres de contexte n'est pas fermé par les opérateurs d'intersection et de complémentarité.

C Machine de Turing

1. Une **machine de Turing** (MT) est un 7-uple $(Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ où Q est l'ensemble fini des états, Σ est l'alphabet d'entrée, Γ est l'alphabet de la bande, $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ est la fonction de transition, $q_0 \in Q$ est l'état de départ, $q_a \in Q$ est l'état acceptant, $q_r \in Q$ est l'état rejetant.
2. L'**exécution d'une MT** commence avec un mot w écrit sur sa bande et dans l'état q_0. La MT accepte (resp. rejette) w si elle atteint l'état q_a (resp. q_s), sinon elle ne s'arrête jamais.
3. Une MT est dans l'état $uq_i v$ si la bande contient la chaîne uv et le pointeur de lecture placé sur le premier caractères de la chaîne u.
4. Un langage L est accepté par une MT M si pour tout $w \in L$, M s'arrête sur un état acceptant.

5. Un langage est (récursivement) **énumérable** s'il existe une MT qui l'accepte.
6. Une MT décide un langage L si MT accepte w pour tout $w \in L$ et rejette w pour tout $w \notin L$.
7. **Théorème** : tous les modèles de MT (transition S, multibandes, ...) sont équivalents.
8. **Complétude de Turing** : un formalisme de machine est Turing-complet s'il permet de simuler une MT.
9. une MT est dite **non déterministe** (MTND) s'il existe plus d'une transition possible à partir d'un même état et symbole d'entrée.
10. L'**exécution d'une MTND** conduit alors à l'évaluation de toutes les transitions possibles en même temps. La MNTD accepte l'entrée si au moins une branche l'accepte, rejette l'entrée si toutes les branches la rejettent ou boucle à l'infini.
11. **Théorème** : toute MTND a un MT équivalente.
12. un énumérateur est une MT avec une bande de travail qui envoie sur sa sortie (éventuellement avec répétition) l'ensemble des mots reconnus par un langage.
13. **Thèse de Church-Turing :** tout calcul informatique est équivalent à un algorithme s'exécutant sur une MT.
14. **Encodage :** on note $< O >$ l'encodage de l'objet O sur la bande d'entrée d'une MT.
15. une **MT universelle** est une MT qui peut simuler une MT M arbitraire sur une entrée arbitraire w.

D Décidabilité

1. **classes de langages** : $\mathcal{RE}$ = classe des langages énumérables. $co\mathcal{RE}$ = classe des langages dont le complément est énumérable (=co-énumérable). $\mathcal{R}$ = classe des langages décidables.
2. **Théorème** : si un langage L est décidable, alors $\bar{L}$ est énumérable.
3. **Théorème** : $\mathcal{R} = \mathcal{RE} \cap co\mathcal{RE}$ (un langage est décidable si et seulement si il est énumérable et co-énumérable).
4. **Problèmes d'acceptation** : on note $A_{\texttt{xxx}} = \{ \langle M, w \rangle \mid$ M est un `xxx` qui accepte w $\}$. $A_{\texttt{ADF}}$, $A_{\texttt{ANF}}$, $A_{\texttt{GLC}}$ sont décidables. $A_{\texttt{MT}}$ est énumérable mais pas décidable.
5. Deux ensembles A et B ont la même taille s'il existe une bijection entre A et B.
6. Un ensemble A est dénombrable s'il a la même taille que $\mathbb{N}$ (donc $\mathbb{N}$ est dénombrable).
7. L'ensemble $\mathbb{R}$ n'est pas dénombrable.
8. si Σ un ensemble fini, alors Σ^* est dénombrable.
9. L'ensemble des MTs est dénombrable.
10. Soit $\mathbb{B}$ l'ensemble des suites binaires de longueur infinie. $\mathbb{B}$ n'est pas dénombrable.
11. L'ensemble des langages $\mathcal{L}$ sur un alphabet Σ fini n'est pas dénombrable.
12. $H_{\text{MT}} = \{< M, w > \mid M$ est une MT qui s'arrête sur $w\}$ est indécidable.
13. $E_{\text{MT}} = \{< M > \mid M$ est une MT et $L(M) = \emptyset\}$ est indécidable.
14. $EQ_{\text{MT}} = \{< M_1, M_2 > \mid M_1$ et M_2 sont des MTs et $L(M_1) = L(M_2)\}$ est indécidable.
15. Une propriété P est un sous-ensemble de $\mathcal{RE}$.
16. Une propriété P d'un langage est dite triviale si $P = P_\emptyset$ (où $P\emptyset$ est la propriété qui ne reconnait aucun langage) ou $P = P_{\text{ALL}}$ (où P_{ALL} est la propriété qui reconnait tous les langages).
17. Une propriété P d'un langage est dite non triviale si elle n'est pas triviale, et si il existe une MT M qui reconnait P, et une MT $\overline{M}$ qui reconnait $\overline{P}$.
18. **Théorème de Rice** : si P est une propriété non-triviale des langages $\mathcal{RE}$, alors P est indécidable.
19. Une fonction $f : \Sigma^* \to \Sigma^*$ est (totalement) calculable s'il existe une MT M qui commence avec l'entrée w sur sa bande et s'arrête pour tout w et avec seulement $f(w)$ écrit sur sa bande.
20. Une fonction $f : \Sigma^* \to \Sigma^*$ est partiellement calculable s'il existe une MT M qui commence avec l'entrée w sur sa bande ; s'arrête pour tout w et avec seulement $f(w)$ écrit sur sa bande si $f(w)$ est défini ; ne s'arrête pas si $f(w)$ est indéfini.
21. Toutes les fonctions arithmétiques classiques sont calculables.
22. Soit l'ensemble de toutes les MTs à une bande avec $\Sigma = \{\sqcup, 1\}$, la bande infinie des deux côtées, initialisée avec $\sqcup$. Soit le sous-ensemble S_n des MTs M à n états tel que $M(\epsilon)$ s'arrête. Soit $\mathcal{S}(n)$ le nombre maximum de transitions exécutées par une machine de S_n sur l'entrée ϵ. La fonction $\mathcal{S}(n)$

(dite du castor affairé) n'est pas calculable.

23. Une réduction d'un langage A vers un langage B s'il existe une fonction calculable $f : \Sigma^* \to \Sigma^*$ telle que pour tout w, $w \in A \Leftrightarrow f(w) \in B$. Notation : $A \leq_m B$.
24. Si $A \leq_m B$ et B est décidable, alors A est décidable.
25. Si $A \leq_m B$ et A est indécidable alors B est indécidable.
26. $A \leq_m B$ implique $\bar{A} \leq_m \bar{B}$
27. Si $A \leq_m B$ et A n'est pas $\mathcal{RE}$ alors B n'est pas $\mathcal{RE}$.
28. Si $A \leq_m B$ et A n'est pas $co\mathcal{RE}$ alors B n'est pas $co\mathcal{RE}$.

E Complexité temporelle

1. On dit que g est une borne asymptotique supérieure pour f (noté $f(n) = O(g(n))$) si $\exists c > 0, n_0 \in \mathbb{N}^* \mid \forall n \geq n_0, f(n) \leq c.g(n)$. Se comprend comme f ne grandit pas plus vite que g.
2. $f(n) = O(g(n)) = O(h(n))$ signifie $f(n) = O(g(n))$ et $g(n) = O(h(n))$.
3. **Ordres de grandeur** : 1 (constante), $\log n$ (logarithmique), n (linéaire), $n \log n$ (quasi-linéaire), n^k ($k > 1$, polynomial), k^n ($k > 1$, exponentiel), n^n ou $n!$ (poly-exponentiel).
4. Le **temps d'exécution** d'une MT M est une fonction $f : \mathbb{N} \to \mathbb{N}$ où $f(n)$ est le nombre maximum de pas nécessaires à M pour décider une chaîne d'entrée de longueur n.
5. La classe de complexité temporelle $\texttt{TIME}(t(n))$ est l'ensemble de tous les langages décidables par une MT à temps $O(t(n))$.
6. Pour toute MT à k-bandes qui s'exécute en temps $t(n)$, il existe une MT à une bande qui s'exécute en temps $O(k^2 t(n)^2)$.
7. Si un MT M s'exécute en temps $t(n) = O(n^c)$ avec $c > 1$, alors on dit que M s'exécute en temps polynomial.
8. Toute MT M_k à k-bandes à temps polynomial possède une MT M à une bande équivalente à temps polynomial.
9. Une MTND M est un décideur pour le langage $\mathcal{L}(M)$ si l'évaluation de toutes ses branches s'arrête pour toutes les entrées.
10. le **temps d'exécution** d'une MTND M est la fonction $f : \mathbb{N} \to \mathbb{N}$ où $f(n)$ retourne le nombre maximum de transitions que M traverse dans n'importe laquelle de ses branches de calcul sur une entrée de longueur n (*i.e.* $f(|w|)$ est le temps d'exécution de la branche la plus longue de $M(\langle w \rangle)$).
11. Pour toute MT non déterministe M' qui s'exécute en temps $t(n)$, il existe une MT à déterministe M à une bande qui s'exécute en temps $2^{O(t(n))}$.
12. **P** est la classe des langages qui sont décidables en temps polynomial par une MT à simple bande. Autrement dit : $\mathbf{P} = \bigcup_k \texttt{TIME}(n^k)$.
13. Exemples de langage de **P** :
    - $\texttt{PATH} = \{\langle G, s, t \rangle \mid G$ est un graphe avec un chemin de s vers $t\}$
    - $\texttt{RELPRIME} = \{\langle x, y \rangle \mid x$ et y sont entiers et $\texttt{PGCD}(x, y) = 1\}$
14. Un **vérificateur** V pour un langage A est un algorithme tel que $\forall w \in A$, $\exists c$ tel que $V(\langle w, c \rangle)$ accepte, et $\forall w \notin A$, $\forall c$, $V(\langle w, c \rangle)$ rejette.
15. La classe **NP** est la classe des langages qui sont polynomialement vérifiables.
16. **Théorème** : Un langage est dans **NP** si et seulement si il est décidé par une MT non-déterministe en temps polynomial.
    Autrement dit, si on note $\texttt{NTIME}(t(n))$ = ensemble des langages qui peuvent être décidés par une MT non-déterministe à temps $O(t(n))$ alors, $\mathbf{NP} = \bigcup_k \texttt{NTIME}(n^k)$.
17. Exemples de langages de **NP** :
    - $\texttt{HAMPATH} = \{ \langle G, s, t \rangle \mid$ G graphe orienté avec un chemin Hamiltonien de s à $t\}$ où un chemin Hamiltonien dans un graphe orienté est un chemin orienté qui passe exactement une seule fois par tous les sommets.
    - $\texttt{COMPOSITES} = \{x \mid x = pq,$ pour deux entiers $p, q > 1\}$.
    - $\texttt{CLIQUE} = \{\langle G, k \rangle \mid G$ est un graphe avec une k-clique$\}$
      où un graphe complet K_p est un graphe qui contient p noœuds, et tel que chaque noœud soit

connecté au $p-1$ autres noœuds par une arête, et une p-clique est un sous-graphe complet K_p d'un graphe G non orienté.
- $\texttt{SUBSET-SUM} = \left\{\langle S, t\rangle \mid \exists S' \subseteq S, \sum_{x_i \in S'} x_i = t\right\}$.
- $\texttt{SAT} = \{\langle F\rangle \mid F$ est une expression logique satisfiable$\}$.
- $\texttt{FNC-SAT} = \{\langle F\rangle \mid F$ est une FNC satisfiable$\}$.
- $\texttt{SAT}_3 = \{\langle F\rangle \mid F$ est une $\texttt{FNC}_3$ satisfiable$\}$

18. $\mathbf{P} \subseteq \mathbf{NP} \subseteq \texttt{EXPTIME}$ où $\texttt{EXPTIME} = \bigcup_k \texttt{TIME}(2^{n^k})$.
19. Une fonction $f : \Sigma^* \to \Sigma^*$ est une **fonction calculable en temps polynomial** s'il existe une MT M à temps polynomial qui s'arrête avec $f(w)$ sur sa bande lorsque son entrée est w.
20. Un langage A est **réductible en temps polynomial** en un langage B, s'il existe une fonction f calculable en temps polynomial telle que : $w \in A \Leftrightarrow f(w) \in B$. On note $A \leq_p B$.
21. Soit $A \leq_P B$ et $B \in \mathbf{P}$. Alors $A \in \mathbf{P}$.
22. Un langage est B est **NP**-complet si $B \in \mathbf{NP}$ et $\forall A \in \mathbf{NP}$, $A \leq_P B$.
23. Si B est **NP**-complet et $B \in \mathbf{P}$ alors $\mathbf{P} = \mathbf{NP}$.
24. Si B est **NP**-complet et $B \leq_P C$ et $C \in \mathbf{NP}$ alors $\mathbf{C}$ est **NP**-complet.
25. **Théorème (Cook-Levin) :** SAT et FNC-SAT sont **NP**-complets.
26. Exemples de langages **NP**-complets :
    - $\texttt{SAT}_3$ (par réduction de FNC-SAT à $\texttt{SAT}_3$).
    - CLIQUE (par réduction de $\texttt{SAT}_3$ à CLIQUE).
    - SUBSET-SUM (par réduction de $\texttt{SAT}_3$ à SUBSET-SUM).
27. Un langage C appartient à *co***NP** si $\overline{C} \in \mathbf{NP}$.
28. Un langage C est *co***NP**-complet si $\overline{C} \in \mathbf{NP}$ et $\forall D \in$ *co***NP**, $D \leq_p C$.

F Complexité spatiale

1. L'**espace d'exécution** d'une MT M est une fonction $f : \mathbb{N} \to \mathbb{N}$ où $f(n)$ est le nombre maximum de cases de la bande nécessaires à M pour décider une chaîne d'entrée de longueur n.
2. L'**espace d'exécution** d'une MTND M est la fonction $f : \mathbb{N} \to \mathbb{N}$ où $f(n)$ est le nombre maximum de cases de la bande que n'importe quelle branche de calcul de M traverse pour une entrée de longueur n.
3. $\texttt{SPACE}(f(n))$ est l'ensemble des langages décidés par une MTD M qui s'exécute en espace $f(n)$.
4. $\texttt{NSPACE}(f(n))$ est l'ensemble des langages décidés par une MTND M qui s'exécute en espace $f(n)$.
5. $\texttt{SAT} \in \texttt{SPACE}(n)$.
6. Soit $\texttt{ALL}_{\texttt{ANF}} = \{\langle M\rangle \mid M$ est un ANF et $L(M) = \Sigma^*\}$, $\overline{\texttt{ALL}_{\texttt{ANF}}} \in \texttt{NSPACE}(n)$.
7. **Théorème (Savitch) :** soit une fonction $f : \mathbb{N} \to \mathbb{R}$ telle que $f(n) \geq n$, alors $\texttt{NSPACE}(f(n)) \subseteq \texttt{SPACE}(f(n)^2)$.
8. PSPACE est la classe des langages qui sont décidables en temps polynomial par une MTD, à savoir $\texttt{PSPACE} = \cup_k \texttt{SPACE}(n^k)$.
9. NPSPACE est la classe des langages qui sont décidables en temps polynomial par une MTND, à savoir $\texttt{NPSPACE} = \cup_k \texttt{NSPACE}(n^k)$.
10. **Proposition :** $\forall f(n), \texttt{SPACE}(f(n)) = \text{co}\texttt{SPACE}(f(n))$.
11. **Théorème :** $\texttt{PSPACE} = \texttt{NPSPACE}$.
    conséquence : $\texttt{PSPACE} = \text{co}\texttt{PSPACE} = \texttt{NSPACE} = \text{co}\texttt{NSPACE}$
12. **Proposition :** une MT M en espace $f(n)$ a au plus $f(n).2^{O(f(n))}$ configurations différentes.
13. **Théorème :** $\mathbf{P} \subseteq \mathbf{NP} \subseteq \texttt{PSPACE} \subseteq \texttt{EXPTIME}$
14. Un langage B est PSPACE-complet si $B \in \texttt{PSPACE}$ et $\forall A \in \texttt{PSPACE}$, $A \leq_{\mathbf{P}} B$.
15. **Théorème :** TQBF est **PSPACE**-complets.
16. FORMULA-GAME (jeu à formule) = paire (X, Ψ) où X est une liste de littéraux (= variables booléennes, *i.e.* $X = [x_1, x_2, \ldots, x_m]$) et Ψ est une formule booléenne sans quantificateurs.
17. **Proposition :** FORMULA-GAME est **PSPACE**-complets.
18. Modèle de MT pour un décideur sub-linéaire : MT à deux bandes, pour laquelle la première bande contient l'entrée (en lecture seule) et la seconde bande est la bande de travail (en lecture/écriture).

19. $\mathtt{L} = \mathtt{SPACE}(\log n)$.
20. $\mathtt{NL} = \mathtt{NSPACE}(\log n)$. **Exemple :** $\mathtt{PATH} \in \mathtt{NL}$
21. Un **transducteur** est une MT avec une bande d'entrée en lecture seule, une bande de travail et une bande de sortie en écriture seule.
22. Une fonction calculable en espace logarithmique $f(w)$ est un transducteur qui prend en entrée $\langle w \rangle$, dont la bande de travail contient au plus $O(\log n)$ symboles, et qui renvoie la valeur stockée sur sa bande de sortie au moment où le transducteur s'arrête.
23. Un langage A est dit réductible en espace logarithmique à un langage B si il existe une réduction de A à B par une fonction calculable en espace logarithmique. On la note : $A \leq_{\mathtt{L}} B$.
24. Un langage est dit $\mathtt{NL}$-complet si $B \in \mathtt{NL}$ et $\forall A \in \mathtt{NL}$, $A \leq_{\mathtt{L}} B$.
25. **Théorème :** si $A \leq_L B$ et $B \in \mathtt{L}$ alors $A \in \mathtt{L}$.
26. **Proposition :** si n'importe quel langage $\mathtt{NL}$-complet est dans $\mathtt{L}$, alors $\mathtt{L} = \mathtt{NL}$.
27. **Proposition :** $\mathtt{PATH}$ et $\overline{\mathtt{PATH}}$ sont $\mathtt{NL}$-complets.
28. **Théorème :** $\mathtt{NL} = \mathtt{coNL}$
29. **Théorème :** $\mathtt{NL} \subseteq \mathbf{P}$

G Rappel de logique :

1. **littéral** = variable booléenne (1/vrai ou 0/faux).
2. **clause** = plusieurs littéraux connectés par $\vee$.
3. **forme normale conjonctive** = plusieurs clauses connectées par $\wedge$.
4. Une $\mathtt{FNC}$ F est satisfiable s'il existe une combinaison de littéraux telle que F soit logiquement vrai.
5. Une $\mathtt{FNC}_k$ est une $\mathtt{FNC}$ constituée de k-clauses.
6. **quantificateur** = $\forall$ (pour tout) et $\exists$ (il existe)
7. **formule booléenne quantifiée** (ou $\mathtt{QBF}$) = expression logique dont **tous** les littéraux sont quantifiés.