1. Langages rationnels

1.1. Premières définitions

Un **alphabet** est un ensemble fini souvent note A ou Σ dont les elements sont appeles **lettres** ou **symboles.**

Un **mot w sur un alphabet A de longueur $n \in N$** correspond à une suite finie $w_1, \dots, w_n$ de n lettres de l'alphabet A.

Le **mot vide** note ε ou parfois $\mathbf{1}$ correspond à l'unique mot de longueur 0.

Un **langage sur un alphabet** correspond à un ensemble de mots sur cet alphabet.

Le **monoïde libre A^* sur un alphabet A** correspond au langage constitue de tous les mots sur cet alphabet. Un langage sur A correspond donc à un sous-ensemble de A^*

Par extension, un **monoïde est dit libre** ssi il est juste isomorphe a un monoïde de cette forme A^*

La **concaténation d'un mot $u = u_1 \dots u_m$ avec un mot $v = v_1 \dots v_n$** est le mot $u \cdot v = uv = u_1 \dots u_m v_1 \dots v_n$.

La concaténation est associative, de neutre le mot vide ε, donc munit bien A^* d'une structure de monoide.

Ce monoïde est dit **libre** car les seules équations qu'il satisfait sont celles qui sont conséquences de la définition de monoïde. (Voir structure libre. algèbre universelle [Alm94])

Pour un alphabet de plus d'une lettre, la concaténation n'est pas commutative.

Un **monoïde** est un ensemble muni d'une l.c.i. associative et qui admet un neutre.

Exemples de monoïdes : A^* muni de la concaténation, $P(A^*)$ muni du produit de langages, tout groupe, $M = \{1, \alpha, \beta\}$ de neutre 1 de loi ($xy = x$) définie par $\alpha\beta = \alpha^2 = \alpha$ et $\beta\alpha = \beta^2 = \beta$, $M = \{1, \alpha, \beta\}$ de neutre 1 de loi $xy = y$, $M = \{1\} \cup I \times J$ de loi $(i,j)(i',j') = (i,j')$, $(N, +)$ est un monoïde.

Un **sous-monoïde** d'un monoïde X est une partie de X qui contient le neutre de X, et qui est stable par la l.c.i., càd $M \subseteq X,\ \varepsilon \in M,\ MM \subseteq M$

Une partie $M \subseteq X$ est un sous-monoïde de X ssi l'identité $M \to^{id} X$ est un morphisme de monoïdes.

Exemple : Les sous-monoïdes de A^* sont les L^* avec $L \subseteq A^*$.

Exemple : L'ensemble des parties rationnelles de A^* est un sous-monoïde de $P(A^*)$

Un **morphisme de monoïdes** est une application f entre deux monoïdes compatible avec la structure de monoïde, c'est-à-dire telle que $f(\varepsilon) = \varepsilon'$ et telle que $f(uv) = f(u)f(v)$.

La longueur d'un mot est un morphisme de monoïdes de $(A^*, \cdot) \to (N, +)$

Une application $\mu: A \to M$ correspond à un morphisme de monoïdes $A^* \to M$ en posant $\hat{\mu}(a_1 \dots a_n) = \mu(a_1) \dots \mu(a_n)$

Un **préfixe** d'un mot w est un mot u tel qu'il existe un mot v tel que $w = uv$

Un **suffixe** d'un mot w est un mot u tel qu'il existe un mot v tel que $w = vu$

Une **occurrence d'un mot u** de longueur p **dans un mot w** de longueur n est un indice $k \in \{1, \dots, n\}$ à partir duquel on peut lire u dans w, autrement dit tel que $\forall i \in \{1, \dots, p\}\ w_{k+i-1} = u_i$.

Un **facteur** d'un mot w est un mot u tel qu'il existe deux mots v, v' tel que $w = vuv'$, autrement dit ssi le mot u admet une occurrence dans le mot w.

Un **sous-mot** d'un mot $w = w_1 \dots w_n$ est un mot u obtenu en supprimant certaines lettres de w, cad tel qu'il existe une sous-suite croissante $1 \le i_1 < \dots < i_k \le n$ d'indices tels que $u = w_{i_1} \dots w_{i_k}$

Un facteur est un sous-mot constitue de lettres consécutives.

Une **occurrence circulaire d'un mot u** de longueur p **dans un mot w** de longueur $n \ge p$ est un indice $k \in \{1, \dots, n\}$ à partir duquel on peut lire u dans w en revenant au début si on atteint la fin, autrement dit tel que $\forall i \in \{1, \dots, p\}\ w_{(k+i-1)'} = u_i$ avec $k' = \big((k-1)\%n\big) + 1$

Un **facteur circulaire** d'un mot w est un mot u qui admet une occurrence circulaire dans le mot w.

Pour un alphabet et une longueur n fixee, on peut construire un mot dans lequel tous les mots de longueur n de l'alphabet ont une occurrence circulaire. Un tel mot est un **mot de Brujin d'ordre n**.

1100 est un mot de Brujin d'ordre 2 sur $\{0,1\}$, 11101000 est un mot de Brujin d'ordre 3 sur $\{0,1\}$
Il y a bijection entre les mots de Brujin d'ordre k et les cycles eulériens de l'automate d'états A^{k-1} de transitions $\{au \to^a ub : a, b \in A \text{ et } u \in A^{k-2}\}$.
Un **morphisme de mots finis** est un morphisme de monoïdes entre deux monoïdes libres $A^* \to B^*$
Un morphisme de mots finis $A^* \to B^*$ est completement determine par les images des lettres de A.
Un morphisme de mots finis $A^* \to B^*$ peut donc être vu comme une application $\mu: A \to B^*$
Un morphisme de mots finis est **effaçant** ssi il envoie une certaine lettre de A sur le mot vide de B
Un mot w sur un alphabet est un **carré** ssi $w = uu = u^2$ avec u un mot.
Un mot sur un alphabet est dit **double** ssi toute lettre qui apparait au moins une fois dans le mot, y apparait au moins 2 fois. Càd toute lettre de l'alphabet a 0 ou au moins 2 occurrences dans le mot.
Tout mot de longueur $\geq 2^n$ sur un alphabet de n lettres, contient au moins un facteur double (par récurrence sur n). Cette borne de 2^n est optimale (mot de Zimin).

1.2. Operations rationnelles

L'union d'un langage L avec un langage L' est le langage $\boldsymbol{L + L' = L \cup L'}$
Le produit d'un langage L par un langage L' est le langage $\boldsymbol{LL' = \{uv : u \in L, v \in L'\}}$
Exemple : Sur un même alphabet A, si L (resp L') est l'ensemble de tous les mots de longueur paire (resp. impaire), alors $L + L' = A^*, LL' = L'L = L',\ L'L' = L \setminus \{\varepsilon\},\ LL = L$ (L sous-monoide de A^*)
La puissance k d'un langage L est $\boldsymbol{L^k = LL^{k-1}}$ si $k \geq 1$, $\boldsymbol{L^0 = \{\varepsilon\}}$ si $k = 0$.
En général $L^k \neq \{u^k : u \in L,\ k \in N\}$. $L^k = \{u_1 \dots u_k : u_1 \in L_1, \dots, u_k \in L_k\}$
Un sous-monoïde de A^* correspond à un langage sur A qui contient le mot vide, et stable par concaténation.
L'étoile d'un langage L est $\boldsymbol{L^* = \bigcup_{k \in N} L^k}$, c'est-à-dire que l'étoile d'un langage est le plus petit sous-monoïde de A^* qui contient le langage L.
Inversement tout sous-monoïde de A^* est étoile d'un langage, car étoile de lui-même.
Exemples : L'étoile du langage $\{a, ba\}$ est l'ensemble de tous les mots dans lesquels chaque b est suivi d'un a.
L'étoile du langage $\{a^n b : n \geq 0\}$ est constitué du mot vide et de tous les mots qui se terminent par un b.
L'étoile du langage vide est constitué du mot vide. $\emptyset^* = \{\varepsilon\}$
Pour un langage L on note $\boldsymbol{L^+ = LL^* = L^*L = \bigcup_{k \geq 1} L^k}$
Sur un alphabet A, A^+ est l'ensemble des mots non vides.

1.3. Combinatoire des mots

1.3.1. Périodicités

Une **période d'un mot** de longueur n, est un entier $p \in \{1, \dots, n\}$ tel que $\forall i \in \{1, \dots, |w| - p\}\ w_i = w_{i+p}$, càd deux lettres quelconques du mot, espacées de p positions sont identiques.
On note $\boldsymbol{P(w)}$ l'ensemble des périodes d'un mot w. Par convention la longueur d'un mot en est toujours une période, donc $|w| \in P(w)$.
Une période ne divise pas nécessairement la longueur dans mot.
Un mot w dont p est une période se factorise sous la forme $w = (uv)^k u$ avec $k \geq 1$, $p = |uv|$ et $v \neq \varepsilon$.
Un **mot primitif** est un mot qui ne s'écrit pas $w = u^k$ avec $k \geq 2$. Autrement dit un mot est primitif ssi sa longueur est l'unique période qui divise sa longueur.
Factorisation primitive : Tout mot w s'ecrit de manière unique $w = u^k$ avec $k \geq 1$ et u primitif. ($k = 1$ ssi w primitif)
Exemples : Si $w = 010101,\ P(w) = \{2,4,6\},\ w = (01)^3$ n'est pas primitif.
Si $w = 01001010010,\ P(w) = \{5,8,10,11\}$, or $|w| = 11$, donc w est primitif.
Fine et Wilf, 1965. Si p et q sont deux periodes d'un mot w de longueur $\geq p + q - p \wedge q$, alors $p \wedge q$ est aussi une période de w. Cette borne minimale est optimale.

La **suite des mots de Fibonacci** est définie par $f_0 = 1,\ f_1 = 0, f_{n+2} = f_{n+1}f_n\ \ \forall n \geq 0$ et ont beaucoup de propriétés remarquables.

Guibas et Odlyzko, 1981*. Tout mot sur un alphabet quelconque, a les mêmes périodes qu'un certain mot sur un alphabet binaire. La preuve est effective, on peut en déduire un algorithme qui trouve en temps $O(|w|)$ le mot $w' \in \{0,1\}^*$ tel que $P(w') = P(w)$.

Exemple : Pour six mots x, y, z, x', y', z' tels que $xyz \neq x'y'z'$, alors $\exists k_0 \in N\ \forall k \geq k_0\ \ xy^k z \neq x'y'^k z'$

1.3.2. Mots infinis

Un **mot infini sur un alphabet** correspond à une suite (infinie) de lettres de l'alphabet.

L'ensemble des mots infinis sur un alphabet A est note $\boldsymbol{A^\omega}$

Un morphisme de mots finis $\mu: A^* \to B^*$ peut être étendu en **morphisme de mots infinis** $\mu: A^\omega \to B^\omega$ en posant pour tout mot infini $x = x_0x_1x_2 \ldots$, $\boldsymbol{\mu(x)} = \mu(x_0)\mu(x_1)\mu(x_2) \ldots$ cette définition est sensée car chaque $\mu(x_k) \in B^*$ est de longueur finie.

Si μ est effaçant alors l'image d'un mot infini peut être un mot fini.

Pour un endomorphisme de mots $\mu: A^* \to A^*$, l'endomorphisme $\boldsymbol{\mu^k}: A^* \to A^*$ est défini par $\mu^0(x) = x$, $\forall k \geq 1\ \mu^k(x) = \mu\left(\mu^{k-1}(x)\right)$

Si x est préfixe de $\mu(x)$, alors $\forall k \in N\ \ \mu^k(x)$ préfixe de $\mu^{k+1}(x)$. Si de plus $\left|\mu^k(x)\right| \to_{n\to\infty} \infty$, alors $\exists!\ \boldsymbol{\mu^\omega(x)} \in A^\omega\ \ \mu^k(x) \to_{k\to\infty} \mu^\omega(x)$, de plus $\mu^\omega(x)$ est un point fixe de μ.

Exemple : L'endomorphisme de mots défini par $\psi(0) = 01, \psi(1) = 221, \psi(2) = 2$, defini alors le mot infini $\psi^\omega(0) = a_0a_1 \ldots$ ou $\ a_0 = 0, a_k = 1$ si k carre non nul, $a_k = 2$ sinon.

Donc le mot infini binaire qui vaut 1 dans $\{k^2 : k \in N\}$ et 0 ailleurs est de la forme $\mu\left(\psi^\omega(0)\right)$ avec μ, ψ morphismes de mots. Plus généralement le mot infini binaire qui vaut 1 dans $\{P(k) : k \in N\}$ et 0 ailleurs est de la forme $\mu\left(\psi^\omega(0)\right)$ avec μ, ψ morphismes de mots.

Exemple : La suite des mots de Fibonacci peut se définir aussi par $\left(\mu^k(0)\right)_{k\in N}$ avec μ **le morphisme de Fibonacci** défini par $\mu(0) = 01$ et $\mu(1) = 0$.

Le **mot infini de Fibonacci** est $\mu^\omega(0)$ pour ce morphisme de Fibonacci.

1.3.3. Motifs inévitables

On suppose fixe un ensemble F de mots sur un alphabet A, donc $F \subseteq A^*$, et on s'intéresse à l'ensemble $A^* \setminus A^*FA^*$ des mots qui évitent F, c'est-à-dire qui ne contiennent pas comme facteurs des mots de F.

La question principale de cette partie est de déterminer si $A^* \setminus A^*FA^*$ est fini ou non.

Lemme : $A^* \setminus A^*FA^*$ est infini ssi il existe un mot infini sur A sans facteurs dans F.

Un mot **carré** est un mot de la forme uu pour un mot u.

Un mot **sans carré** est un mot qui ne contient pas de facteur carré (autre que le mot vide).

Sur un alphabet a 3 lettres, il existe des mots sans carré arbitrairement longs.

Sur l'alphabet binaire, les seuls mots sans carré sont 010 et 101 et leurs facteurs.

Un **chevauchement** est un mot de la forme $uvuvu$ avec u mot non vide et v mot quelconque.

Un **mot sans chevauchement** est un mot dont aucun facteur n'est un chevauchement.

Un mot contient un chevauchement ssi un de ses facteurs a 2 occurrences qui se chevauchent.

Le **morphisme de Thue-Morse** est l'endomorphisme de mot de $\tau: A^* \to A^*$ defini par $\tau(0) = 01, \tau(1) = 10$.

Le **mot de Thue-Morse** est le mot $\tau^\omega(0)$. La n-ieme lettre du mot de Thue-Morse est 0 si le nombre de 1 dans l'écriture de n est pair, et 1 sinon. Le mot de Thue-Morse est sans chevauchement.

Par définition du morphisme de Thue-Morse, $\forall n \geq 1\ \ \tau^n(0) \in (01 + 10)^*$

Si $x \in (01 + 10)^*$ alors $0x0$ et $1x1$ n'appartiennent pas a $(01 + 10)^*$

Thue 1906. Le mot $\sigma^{-1}\left(\tau^\omega(0)\right)$ est sans carré.

Un **alphabet de motif** est un alphabet X distinct du principal alphabet étudié A.

Un **motif** est un mot d'un alphabet de motif.

Sur un alphabet A, **un mot w contient un motif p** d'un alphabet de motif X, ssi w contient un facteur $\mu(p)$ pour un certain morphisme de mots $\mu: X^* \to A^*$ non effaçant.
Dans le cas contraire on dit que **le mot w évite le motif p.**
Un mot w sur un alphabet A **matche un motif p** d'un alphabet de motif X, ssi w est de la forme $\mu(p)$ pour un certain morphisme de mots $\mu: X^* \to A^*$ non effaçant.
Un mot carre est un mot de motif xx.
Un chevauchement est un mot de motif $xyxyx$ si v vide, ou de motif xxx si $v = \varepsilon$
Un **motif p est k-inévitable** ssi l'ensemble des mots d'un alphabet A a k lettres qui évitent p, est fini.
Un **motif p est inévitable** ssi il est k inévitable pour tout $k \in N$. La référence sur ce sujet est [Cas02].
Lemme pour construire des motifs inévitables. Pour un motif inévitable p, et une lettre x de motif qui n'apparait pas dans le motif p, alors le motif pxp est inévitable.
La **suite des mots de Zimin** sur un alphabet infini $X = \{x_0, x_1, \dots\}$ est définie par $\boldsymbol{z_0} = \varepsilon,\ \forall n \geq 0\ \boldsymbol{z_{n+1}} = z_n x_n z_n$. Chaque terme de la suite de Zimin sur un alphabet de motif, est un motif inévitable.
La suite de Zimin converge vers un mot infini z, **le mot de Zimin**.
Pour $n \geq 0$, la n-ieme lettre du mot de Zimin est x_k si 2^k est la plus grande puissance de 2 qui divise n.
Il existe un algorithme du a Zimin, qui détermine si un motif donne est inévitable ou non. [Cas02]
Aucun algorithme n'est connu à ce jour pour déterminer si un mot est k-inevitable pour k fixe.

1.3.4. Codes (référence [BPR 08])

Un **code** est un langage dont deux suites de mots qui sont égales, sont identiques càd que tout mot admet au plus une décomposition sur le code. Autrement dit, X **code** sur A ssi $X \subseteq A^*$ et $\forall x_1 \dots x_n \in X^n\ \forall y_1 \dots y_m \in X^m\ \ x_1 \dots x_n = y_1 \dots y_m \Rightarrow n = m$ et $\forall i\ x_i = y_i$.
Exemple : Sur $A = \{a, b\}$, $\{aa, baa, ba\}$ est un code, mais $\{a, ab, ba\}$ pas un code car $aba = (ab)a = a(ba)$ admet 2 décompositions.
Rappel : Un morphisme de mots peut se voir comme une application d'un alphabet A vers un monoïde libre B^*.
Une application d'un alphabet vers un alphabet induit donc un morphisme de mots.
Une application d'un alphabet B vers un langage vu comme un alphabet X, sur un alphabet A définit donc un morphisme de mots $\mu: B^* \to X^*$.
Lorsque l'alphabet B est en bijection avec le langage X, cette application est bijective ssi le langage X est un code. Dans ce cas X^* est monoïde-isomorphe a B^*, et X^* est un sous-monoïde libre de A^*.
Un sous-monoïde M de A^* est **libre** ssi $\forall u, v \in A^*\ \ uv \in M, vu \in M$, et $v \in M \Rightarrow u \in M$
ssi M est l'étoile d'un code sur A.
Lemme : Pour un sous-monoïde M de A^*, $(M \setminus \varepsilon) \setminus (M \setminus \varepsilon)^2$ est un ensemble minimal de générateurs.
Une intersection quelconque de sous-monoïdes libres est un sous-monoïde libre.
Il s'ensuit que tout langage L est contenu dans un plus petit sous-monoïde libre, appelé **enveloppe libre du langage L**.
L'enveloppe libre d'un ensemble fini de mots X qui n'est pas un code, est engendrée par un ensemble Y tel que $|Y| \leq |X| - 1$
Pour deux mots u, v on a les équivalences : leur paire $\{u, v\}$ n'est pas un code $\Leftrightarrow$ u et v sont des puissances d'un même mot w : $u, v \in \{w\}^* \Leftrightarrow$ ils commutent : $uv = vu$
L'enveloppe libre d'un langage rationnel est un langage rationnel*.

1.4. Un peu d'ordre.

Soit R une relation binaire sur une classe X.
Une relation est **réflexive** si tout élément est en relation avec lui-même : $\forall x \in X, xRx$
Une relation est **irréflexive** si aucun élément n'est en relation avec lui-même: $\forall x \in X, \neg xRx$
Une relation est **transitive** si la relation s'hérite linéairement : $\forall x, y, z \in X, xRy$ et $yRz \Rightarrow xRz$
Une relation est **symétrique** si la relation ne dépend pas de l'ordre: $\forall x, y \in X, xRy \Rightarrow yRx$
Une relation est **antisymétrique** si la relation dans les deux sens implique l'égalité : $\forall x, y \in X, xRy$ et

$yRx \Rightarrow x = y$
Une relation est **asymétrique** si la relation n'est possible que dans un sens : $\forall x, y \in X, xRy \Rightarrow \neg yRx$ / ssi la relation est irréflexive et antisymétrique.
Sous irréflexivité, les notions antisymétrie et asymétrie coïncident.
Une **relation d'équivalence** est une relation réflexive, transitive, symétrique.
Deux éléments $x, y \in X$ sont **comparables** pour la relation, s'il y a au moins une relation entre les deux : xRy ou yRx
Une relation est **totale** si tout couple est comparable : $\forall x, y \in X, xRy$ ou yRx
Une relation est **trichotomique large** si tout couple est comparable ou égal : $\forall x, y \in X, xRy$ ou $x = y$ ou yRx
Une relation est **trichotomique stricte** si tout couple vérifie exactement une des 3 propositions xRy, $x = y$, ou yRx.
Alternativement, une relation est **trichotomique stricte** ssi elle est trichotomique large et asymétrique.
Pour une relation binaire notée $\leq$ on peut appelle une **relation stricte induite** notée $<$ définie par $x < y \Leftrightarrow x \leq y$ et $x \neq y$
Pour une relation binaire notée $<$ on peut appelle une **relation large induite** notée $\leq$ définie par $x \leq y \Leftrightarrow x < y$ ou $x = y$
Une relation stricte induite est toujours irréflexive.
Une relation large induite est toujours réflexive.
Une relation binaire est réflexive ssi elle coïncide avec la relation large induite de sa relation stricte induite.
Une relation binaire est irréflexive ssi elle coïncide avec la relation stricte induite de sa relation large induite.
Ces propriétés montrent qu'il y a une dualité entre relation binaire réflexive et irréflexive. Une telle relation peut être vue dans son sens large (version réflexive) ou dans son sens strict (version irréflexive). Fixer l'une revient à fixer l'autre.
Soit $\leq/<$ une relation binaire dans sa version réflexive et irréflexive sur une classe X.
$<$ asymétrique $\Leftrightarrow$ $<$ antisymétrique
$\leq$ totale $\Leftrightarrow$ $<$ trichotomique faible
$<$ trichotomique $\Leftrightarrow$ $<$ a(nti)symétrique et trichotomique faible
$\leq$ **préordre = quasi-ordre /** $<$ **préordre strict**: $\leq$ transitive / $<$ transitive.
On dit que
$\leq$ ordre (partiel) / $(X, \leq)$ ordoné (partiellement) / $<$ ordre strict / $(X, <)$ ordonné strictement lorsque :
$$\begin{cases} \begin{cases} (\leq \text{ réflexive}) \\ \leq \text{ transitive} \end{cases} / \leq \text{ préordre} \\ \leq \text{ antisymétrique} \end{cases} / \begin{cases} \begin{cases} (< \text{ irréflexive}) \\ < \text{ transitive} \end{cases} / < \text{ préordre strict} \\ < \text{ a(nti)symétrique} \end{cases}$$
On dit que
$\leq$ ordre total / $(X, \leq)$ ordonné totalement / $<$ ordre total strict / $(X, <)$ ordonné strict totalement lorsque :
$$\begin{cases} \leq \text{ ordre} \\ \leq \text{ total} \end{cases} / \begin{cases} \begin{cases} (\leq \text{ réflexive}) \\ \leq \text{ transitive} \end{cases} / \leq \text{ préordre} \\ \leq \text{ antisymétrique} \\ \leq \text{ total} \end{cases} / \begin{cases} \begin{cases} (< \text{ irréflexive}) \\ < \text{ transitive} \end{cases} / < \text{ préordre strict} \\ < \text{ a(nti)symétrique} \\ < \text{ trichotomique faible} \end{cases} / < \text{ trichotomique}$$
Exemples : La relation de divisibilité sur N est un ordre, la relation « divise une certaine puissance du nombre » est un préordre sur N.
La relation d'équivalence $\approx$ **induite par un préordre** $\leq$ est definie par $x \approx y \Leftrightarrow x \leq y$ ou $y \leq x$
Un préordre sur un ensemble, devient un ordre si on quotient par la relation d'équivalence qu'il induit.

Une relation d'équivalence sur un ensemble, est un quasi-ordre dont la relation d'équivalence induite est justement la relation en question.
Un **élément $x \in X$ est irreductible/en forme normale pour une relation binaire $\rightarrow$ sur X** (dans un contexte ou $\rightarrow$ symbolise une réduction) ssi il n'existe pas de $y \in X$ tel que $x \rightarrow y$
Une **chaine sur un ensemble préordonné** correspond à une suite finie ou infinie dont pour tout deux termes consécutifs, le suivant est plus petit ou égal au précèdent.
Une **relation binaire $\rightarrow$ est noethérienne** ssi il n'existe pas de chaine infinie $x_0 \rightarrow x_1 \rightarrow \cdots$
Une **chaine stricte sur un ensemble préordonné** correspond à une suite finie ou infinie dont pour tout deux termes consécutifs, le suivant est strictement plus petit que le précèdent.
Un préordre est **bien-fondé** ssi il n'admet pas de chaine stricte infinie, càd ssi sa relation stricte induite sur le quotient $\frac{E}{\approx}$ est noethérienne.
Une **antichaine sur un ensemble préordonné** correspond à un ensemble d'éléments incomparables deux a deux.
Un **idéal d'ordre sur un ensemble préordonné** correspond à une partie dont tout élément n'admet pas de supérieur en dehors de la partie.
Une **base d'un idéal d'ordre sur un ensemble préordonné** correspond à une sous-partie de l'idéal tel que tout élément de l'idéal admet au moins un inférieur ou égal dans cette sous-partie.
On dit que **l'idéal est engendré par** une partie, ssi c'est une base de cet idéal.
Higman. Etant donné un préordre $\leq$ sur un ensemble E, on a les équivalences :
1. Tout idéal d'ordre admet une base finie
2. Toute chaine croissante (pour $\subseteq$) d'idéaux $I_0 \subseteq I_1 \subseteq \cdots$ est stationnaire.
3. Toute suite infinie d'éléments de E contient une sous-suite infinie croissante.
4. Toute suite infinie d'éléments de E contient une sous-suite croissante de longueur 2.
5. Toute chaine stricte est finie, et toute antichaine est finie.
6. Tout préordre $\leq'$ qui prolonge le préordre considéré $\leq$ est bien fondé

Un **bon-préordre** est un préordre satisfaisant les conditions précédentes.
Une relation d'équivalence est un bon préordre ssi elle a un nombre fini de classes.
$\leq$ bon préordre sur N, mais n'est pas bien-fondé sur Z.
Sur N, la divisibilité est un ordre bien-fondé mais pas un bon préordre car les nombres premiers forment une antichaine infinie. Donc il y a une différence entre ordre bien fondé et bon préordre.
Si $\leq_1$ préordre sur E_1, $\leq_2$ préordre sur E_2, alors $(x_1, x_2) \leq (y_1, y_2) \Leftrightarrow x_1 \leq_1 y_1$ et $x_2 \leq_2 y_2$ est un préordre sur $E_1 \times E_2$
Lemme Nash Williams. Si $\leq_1$ bon préordre sur E_1 et $\leq_2$ bon préordre sur E_2 alors $\leq$ est bon préordre sur $E_1 \times E_2$
L'ordre naturel sur les entiers induit un ordre naturel sur les k-uplets d'entiers defini par $(m_1, \dots, m_k) \leq (n_1, \dots, n_k)$ ssi $\forall i \ \ m_i \leq n_i$.
Lemme Dickson. Tout sous-ensemble de N^k a un nombre fini d'elements minimaux.

1.4.1. Préordres sur les mots.

Un préordre $\leq$ sur un alphabet A permet de définir le **préordre des sous-mots** $\leq^*$ sur A^* par $a_1 \dots a_m \leq^* a'_1 \dots a'_m$ ssi il existe une suite croissante $1 \leq j_1 < \cdots < j_k \leq n$ d'indices tels que $\forall i \in \{1, \dots, m\} \ \ a_i \leq a'_{j_k}$
Pour une relation R on note parfois R^* la clôture réflexive et transitive de R. Mais ici $\leq^*$ n'est pas du tout la clôture de $\leq$ (qui est en fait $\leq$). l'étoile fait seulement référence a A^*.
Le préordre des sous-mots $\leq^*$ est aussi le plus petit (au sens inclusion) préordre sur A^* tel que
1. Pour tous mots $u, v \in A^*$ et toute lettre $a \in A$, $\quad uv \leq^* uav$
2. Pour tous mots $u, v \in A^*$ et toutes lettres $a, b \in A$, $\ uav \leq^* ubv$

Le préordre des sous-mots peut être étendu en $\leq^{\omega}$ sur A^{ω} par $(a_n)_{n\geq 0} \leq^{\omega} (a'_n)_{n\geq 0}$ ssi il existe une suite strictement croissante d'indices $(i_n)_{n\geq 0}$ telle que $\forall n \geq 0 \;\; a_n \leq a'_{i_n}$

Higman 1952. Si $\leq$ est un bon préordre sur A alors $\leq^*$ est un bon préordre sur A^*. Faux pour $\leq^{\omega}$ sur A^{ω}

1.4.2. Ordres sur les mots

Le **plus long préfixe commun à deux mots w, w'** est noté $\boldsymbol{w \wedge w'}$. w prefixe de w' ssi $w = w \wedge w'$

La fonction $d(w, w') = |w| + |w'| - 2|w \wedge w'|$ est une distance sur A^*

Un ordre total $\leq$ sur un alphabet A induit un **ordre lexicographique $\leq_{lex}$** sur A^* par $w \leq_{lex} w'$ ssi w préfixe de w' ou $\exists a, b \in A \; \exists p, u, v \in A^* \;\; w = pau, \;\; w' = pbv, \;\; a \leq b$. Dans ce cas $p = w \wedge w'$.

L'ordre lexicographique induit par un ordre d'alphabet total, est un ordre total mais il n'est pas bien-fondé des que l'alphabet a au moins deux lettres. Exemple sur l'alphabet $\{0,1\}$ la suite de mots $(0^n 1)$ est infinie strictement décroissante.

Un ordre total $\leq$ sur un alphabet A induit un **ordre hiérarchique $\leq_{hie}$** ou les mots sont d'abord classés par longueur puis par ordre lexicographique : $w \leq_{hie} w'$ ssi $|w| \leq |w'|$ ou $\big(|w| = |w'|$ et $w \leq_{lex} w'\big)$

Cet ordre a l'avantage d'être total et bien fondé.

1.4.3. Préordres sur les arbres

Le théorème de Higman admet une généralisation aux arbres finis due à Kruskal. La construction de bon préordres sur les arbres est essentielle pour la terminaison des systèmes de réécriture de termes, souvent utilises pour donner une sémantique aux langages de programmation fonctionnels.

Un **domaine d'arbre** est un ensemble de suites finies d'entiers ($D \subseteq N^*$), **clos par préfixe** càd que tout préfixe d'une suite du domaine est encore dedans, et **clos par valeur inferieures** càd que pour une suite dans le domaine, alors toute suite identique sauf la dernière lettre qui est un entier inférieur, est encore dedans $ui \in D \Rightarrow \forall j \in \{0, \dots, i\} \; uj \in D$. Un domaine d'arbre contient en particulier la suite vide ε.

Un domaine d'arbre correspond à un graphe d'arbre. Chaque suite du domaine correspond à un sommet du graphe, et correspond à l'unique chemin de ses prédécesseurs jusqu'à la racine représentée par ε. « clos par préfixe » assure que les prédécesseurs font bien partie du graphe, et « clos par valeur inferieures » assure que les arêtes qui émanent d'un sommet, sont bien numérotées de 0 a k.

Un **arbre étiqueté d'alphabet $\boldsymbol{A}$** correspond à une fonction t d'un domaine d'arbre $D \subseteq N^*$ vers l'alphabet A.

Un arbre étiqueté correspond à un graphe d'arbre dont chaque sommet est affecté d'une lettre de l'alphabet A.

Le **domaine d'un arbre** est noté $\boldsymbol{dom(t)}$

La **taille d'un arbre $\boldsymbol{t}$** notée $|\boldsymbol{t}|$ est le cardinal de son domaine.

Un mot fini $w = a_0 \dots a_n \in A^*$ correspond à un arbre étiqueté t d'alphabet A de domaine $D = \{\varepsilon, 0, 0^2, \dots, 0^n\} = (\varepsilon + 0)^n$ défini par $\forall k \in \{0, \dots, n\} \; t\big(0^k\big) = a_k$. (C'est un graphe linéaire).

Pour un domaine d'arbre D, et $w \in D$, le **domaine du sous arbre enraciné en $\boldsymbol{w} \in \boldsymbol{D}$** est $w^{-1}D$

On peut étendre cette définition pour un mot entier $w \in N^*$ qui $\notin D$ mais alors $w^{-1}D = \emptyset$

Pour un arbre étiqueté t, le **sous-arbre enraciné en $\boldsymbol{w}$** est $\boldsymbol{w^{-1}t}$ defini par $(w^{-1}t)(u) = t(wu)$

Un préordre sur un alphabet A, induit le **préordre naturel d'arbres étiquetés $\leq^{\Delta}$** sur les arbres finis:

Deux arbres finis étiquetés t, t' de domaines D, D' vérifient $t \leq^{\Delta} t'$ ssi $\exists f: D \to D'$ injective telle que pour tous $w, w' \in D, \;\; t(w) \leq t'\big(f(w)\big), f(w \wedge w') = f(w) \wedge f(w')$, et $w \leq_{lex} w' \Rightarrow f(w) \leq_{lex} f(w')$

La fonction f et la 1ere condition généralise la suite croissante d'indice dans la définition du préordre des sous-mots $\leq^*$. Les deux autres conditions assurent que f est bien un plongement de t dans t'. La 2e préserve la relation de parente, et la 3e préserve l'ordre des fils.

Kruskal 1960. Pour un bon préordre sur un alphabet A, alors le préordre d'arbre étiquetés $\leq^{\Delta}$ est un bon préordre sur l'ensemble des arbres finis étiquetés A.

Il est possible de définir des arbres ou l'ordre des fils n'a pas d'importance et Kruskal reste vrai.

Les théorèmes de Higman et Kruskal peuvent être généralisés aux graphes.

Un arbre étiqueté, est **monochrome** ssi $\forall w, w' \in dom(t)\ \ t(w) = t(w')$ ses sommets sont affectés du même label.
On note $\{\mathbf{0}, \mathbf{1}\}^{\leq \boldsymbol{n}}$ l'ensemble des mots de longueur $\leq n$ sur $\{0,1\}$.
Tout arbre étiqueté de domaine $\{0,1\}^{\leq 2n}$ sur l'alphabet $A = \{a, b\}$, est plus grand qu'un arbre étiqueté monochrome de domaine $\{0,1\}^{\leq n}$, où plus grand signifie $\geq^{\Delta}$ induit par l'ordre $=$ sur l'alphabet A.

1.5. Langages rationnels

1.5.1. Expressions rationnelles

La classe R des **langages rationnels sur un alphabet fini $\boldsymbol{A}$** est la plus petite famille de langages telles que :
L'ensemble vide en fait partie, toute singleton lettre en fait partie, la famille est close par les opérations rationnelles (l'union, le produit et l'étoile de Kleene).
Autrement dit, inductivement, $\emptyset$ est un langage rationnel, $\forall a \in A\ \ \{a\}$ est un langage rationnel, l'union/le produit/l'étoile de langages rationnels est un langage rationnel, aucun autre procédé ne donne un langage rationnel.
$\{\varepsilon\}$ est un langage rationnel car $\{\varepsilon\} = \emptyset^*$
Le langage A est rationnel car $A = \bigcup_{a \in A}\{a\}$
Le langage des mots de longueur paire est rationnel car c'est $(AA)^* = (A^2)^*$
Le langage des mots de longueur impaire est rationnel car c'est $A(AA)^*$
Le langage des mots qui contiennent un facteur aba est rationnel car c'est A^*abaA^*
Dans la suite, la notion d'expression n'est pas formellement définie, les parenthèses sont encore utilisées pour lever les ambiguïtés.
La classe des **expressions rationnelles/régulières sur un alphabet fini $\boldsymbol{A}$, muni d'un symbole de concaténation ., d'un symbole d'union $+$, et d'un symbole étoile** $*$ est définie inductivement par :
$\emptyset$ est une expression rationnelle
Toute lettre $a \in A$ est une expression rationnelle
Pour deux expressions rationnelles E, E' alors $E + E', E \cdot E',\ E^*$ sont des expressions rationnelles.
Aucune autre expression n'est rationnelle.
Ici $+, \cdot, *$ sont vus comme des symboles inertes faisant partie de l'expression et non des opérations.
Pour simplifier l'écriture, le $\cdot$ est généralement implicite, et une expression $x_1 + \cdots + x_n$ s'écrit parfois juste X lorsqu'on a aussi défini dans le contexte un ensemble $X = \{x_1, \dots, x_n\}$.
Par exemple on écrit A l'expression régulière $a_1 + \cdots + a_n$, avec $a_1, \dots, a_n$ les lettres de l'alphabet A.
Exemples d'expression régulières : $A, A^*, aA^*, A^*a, (b + ab)^*(a + \varepsilon), a^* + b^*, (aa + b)^*, (ab^*a + b)^*$
Un langage rationnel correspond à une expression rationnelle puisqu'il s'obtient par application d'un nombre fini d'opérations rationnelles.

1.5.2. Automates.

Un **automate/machine à états (fini, non déterministe)/NFA** correspond à un (Q, A, E, I, F) avec A alphabet fini, Q ensemble fini d'**états**, $I \subseteq Q$ ensemble **d'états initiaux**, $F \subseteq Q$ ensemble **d'états finaux**, $E \subseteq Q \times A \times Q$ ensemble de **transitions** étiquetées par lettres, fini car Q et A le sont.
Une autre façon de modéliser les transitions est de remplacer E par une **fonction de transition** $\delta: Q \times A \to P(Q): (q, a) \mapsto \{v \in Q \mid (q, a, q') \in E\}$.
Un automate définit un graphe fini (multiple) dont les sommets sont les états, et chaque transition $(q, a, q') \in E$ represente un arc (q, q') étiqueté par la lettre a. On écrit souvent une transition : $\boldsymbol{q} \to^{\boldsymbol{a}} \boldsymbol{q'}$
Un **automate** est **déterministe** ssi il n'a qu'un seul état initial $I = \{\boldsymbol{q_0}\}$, et $(p, a, q) \in E$ et $(p, a, q') \in E \Rightarrow q = q'$ càd que la fonction de transition est déterministe et partielle $\delta: D \subseteq Q \times A \to Q$
Une **configuration externe d'un automate** correspond à un (q, u, v) avec q un état courant, $u \in A^*$ les lettres qui ont été lues à gauche de la tête, $v \in A^*$ les lettres restantes à lire à droite de la tête, le mot scanné étant $w = uv$. Un automate non déterministe peut avoir plusieurs configurations à la fois.
Une **configuration interne d'un automate $\boldsymbol{M}$** correspond juste à un état courant $q \in Q$

Calcul. On définit une **étape de calcul/calcul élémentaire** comme correspondant à (un couple de configurations et une lettre $\in A$) qui correspondent à une transition.
On peut la qualifier d'externe et la noter $\boldsymbol{C} = (\boldsymbol{q}, \boldsymbol{u}, \boldsymbol{av}) \vdash_{\mathbf{M}} \boldsymbol{C'} = (\boldsymbol{q'}, \boldsymbol{ua}, \boldsymbol{v})$ si on considère dans le contexte un mot scanné fixé, ou d'interne sinon, auquel cas on peut l'identifier à la transition $\boldsymbol{q} \rightarrow^{\boldsymbol{a}} \boldsymbol{q'} \in E$.
Un chemin/calcul/une exécution dans un automate $\boldsymbol{M}$ correspond à une suite consécutive finie d'étapes de calcul. On peut la qualifier d'externe et la noter $\boldsymbol{C_0} = (\boldsymbol{q}, \boldsymbol{u}, \boldsymbol{wv}) \vdash^*_{\mathbf{M}} \boldsymbol{C_n} = (\boldsymbol{q'}, \boldsymbol{uw}, \boldsymbol{v})$ si on considère un mot particulier, ou d'interne sinon, auquel cas on peut noter $\boldsymbol{q} \rightarrow^{\boldsymbol{w}} \boldsymbol{q'}$ avec $w \in A^*$
Une exécution dans un automate correspond donc à une suite finie de transitions consécutives $q_0 \rightarrow^{a_1} q_1 \rightarrow^{a_2} \dots \rightarrow^{a_n} q_n$ avec $q_0, \dots, q_n \in Q$, $a_1, \dots, a_n \in A$, donc correspond à un chemin du graphe.
Une **configuration est initiale (resp. finale)** ssi son état courant l'est.
Une **exécution d'un automate est <u>acceptante</u>** ssi son premier état est initial et son dernier état est final.
Un mot w est **accepté/reconnu** par un automate M s'il est l'étiquette d'une exécution <u>acceptante</u> de l'automate càd ssi $\exists q_0 \in I \; \exists q_n \in F \;\; (q_0, w, \varepsilon) \vdash^*_{\mathrm{M}} (q_n, \varepsilon, w)$
Le langage (accepté) d'un automate $\boldsymbol{M}$, est l'ensemble $\boldsymbol{L}(\boldsymbol{M})$ des mots de A^* qu'il accepte.
Deux automates sont **équivalents** ssi ils acceptent le même langage.
Kleene 1956. Un langage est rationnel ssi c'est le langage d'un automate.
Une **configuration $\boldsymbol{C}$ est accessible** ssi il existe une configuration initiale C_i telle que $C_i \vdash^*_M C$ càd ssi il y a une exécution d'une configuration initiale jusqu'à C.
Une **configuration $\boldsymbol{C}$ est coaccessible** ssi il existe une configuration finale C_f telle que $C \vdash^*_M C_f$ càd ssi il y a une exécution de C jusqu'à une configuration finale.
Une exécution <u>acceptante</u> passe par un état ssi cet état est accessible et coaccessible.
Un automate est **émondé** ssi par tout état passe au moins une exécution acceptante.
On peut faire 2 BFS sur le graphe pour déterminer l'ensemble des états membres d'une exécution acceptante. Les états par lesquels ne passe aucune exécution acceptante peuvent être retirés sans changer le langage de l'automate. On supposera donc souvent wlog que l'automate est émondé.
Un automate est **normalisé** s'il possède un unique état initial avec une seule transition sortante, et un unique état final avec une seule transition entrante.
Si l'état initial et l'état final d'un automate normalisé coïncident, aucune transition n'est adjacente et le langage de l'automate est vide.
Tout automate M admet un **automate normalisé** M' dont le langage est le même, mais sans le mot vide $L(M') = L(M) \setminus \{\varepsilon\} = \alpha\big(L(M)\big)$. Il suffit de contracter les états initiaux en un seul, les états finaux en un seul <u>distinct</u> (ce qui empêche donc ε d'être accepté).
Preuve sens direct de Kleene : Avec $\varepsilon(L) = L \cap \{\varepsilon\}, \alpha(L) = L \setminus \{\varepsilon\}$, on a les formules : $\varepsilon(L + L') = \varepsilon(L) + \varepsilon(L'), \varepsilon(LL') = \varepsilon(L)\varepsilon(L'), \varepsilon(L^*) = \varepsilon, \alpha(L + L') = \alpha(L) + \alpha(L'), \alpha(LL') = \alpha(L)\alpha(L') + \varepsilon(L)\alpha(L') + \alpha(L)\varepsilon(L'), \alpha(L^*) = \alpha(L)^+$ qui permettent de calculer inductivement α de tout langage rationnel, on ne s'intéresse qu'a α, pour les automates normalisés, mais pour α il faut ε.
Ensuite on montre comment construire à partir d'automates normalisés de langages $\alpha(L), \alpha(L')$ on peut construire l'automate normalisé « union/produit/étoile » ce qui permet de générer le langage union/produit/étoile $\alpha(L + L'), \alpha(LL'), \alpha(L^*)$, et ainsi on genere recursivement tout langage rationnel L. (A la fin on a $\alpha(L)$ mais on rajoute le mot vide si nécessaire en ajoutant un état initial final).
Sens indirect Kleene.
Algorithme McNaughton Yamada. Facile à programmer, mais trop calculatoire a la main.
Méthode par élimination. TODO
Lemme d'Arden. Permet de résoudre des équations linéaires entre langages.
Soit l'équation $X = KX + L$ d'inconnue X un langage
Si K ne contient pas le mot vide, l'équation admet pour unique solution $X = K^*L$. Sinon les solutions sont de la forme $X = K^*(L + Y)$ avec $Y \subseteq A^*$. On l'utilise en général que dans le premier cas.

Elimination de Gauss. Pour résoudre un système d'équations linéaires entre langages via lemme d'Arden. TODO
Un automate est **avec $\boldsymbol{\varepsilon}$-transitions** ssi ses étiquettes sont soit une lettre de A, soit le mot vide ε.
Un automate avec ε-transitions est équivalent à un automate sans ε-transitions.

1.6. Automates déterministes.

Un **automate** est **déterministe** ssi il n'a qu'un seul état initial $I = \{s_0\}$, et $(p, a, q) \in E$ et $(p, a, q') \in E \Rightarrow q = q'$ càd que la fonction de transition est déterministe et partielle $\delta: D \subseteq Q \times A \to Q$.
Un mot accepté par un automate déterministe, correspond a exactement une exécution acceptante.
Un préfixe d'un mot accepté par un automate déterministe correspond a exactement une exécution partant de l'état initial.
Tout automate non déterministe est équivalent à un automate déterministe. (il suffit de voir chaque ensemble d'états non déterministes, comme un état déterministe). Lorsqu'on fait ça sur un exemple précis, on essaye d'être plus fin pour éviter une explosion du nombre d'états a 2^n. En pratique : on fait une BFS de l'automate émondé depuis un état initial, pour identifier les arcs parallèles.
Lemme pour l'équivalence : Pour $w \in A^*$ il y a un chemin de I a P dans $\widehat{M}$ étiqueté w ssi $P = \{q | \exists i \in I\ i \to^w q \text{ dans } M\}$
Un **automate est complet** ssi tout état p et toute lettre a mene a un autre état, càd $\forall p \in Q\ \forall a \in A\ \exists q\ p \to^a q \in E$, càd la fonction de transition n'a jamais pour image le vide.
Ainsi, un automate déterministe est complet ssi sa fonction de transition est totale $\delta: Q \times A \to Q$.
On note alors, dans un automate déterministe complet : $\boldsymbol{qa}$ l'état $\delta(q, a)$
Cette notation peut être étendue à tout mot $w \in A^*$ par récurrence, donc $q \to^w \boldsymbol{qw}$
Cette notation $q \cdot w$ definit donc une action à droite du monoïde A^* sur l'ensemble des états Q.
Tout automate (déterministe) est équivalent à un automate (déterministe) complet. (on ajoute un état puits, et toutes les transitions manquantes vers le puits).
Clôture par complément. Le complémentaire dans A^* d'un langage rationnel, est un langage rationnel.
L'intersection de langages rationnels est un langage rationnel.

1.7. Automate minimal

1.7.1. Quotients

Les quotients à gauche sont utiles étudier les automates déterministes puisque leur nombre donne un minorant du nombre d'états. Ils donnent aussi une caractérisation utile des langages rationnels.
Le quotient à gauche d'un langage $\boldsymbol{L}$ par un mot $\boldsymbol{u} \in \boldsymbol{A}^*$ est le langage $\boldsymbol{u^{-1}L} = \{v \mid uv \in L\}$
Le quotient à gauche d'un langage $\boldsymbol{L}$ par un langage $\boldsymbol{K} \subseteq \boldsymbol{A}^*$ est le langage $\boldsymbol{K^{-1}L} = \bigcup_{u \in K} u^{-1}L$
De manière symétrique on peut définir les **quotients à droites**.
Pour toute lettre a, tous mots u, v, w et tous langages K, L on a :
$w^{-1}(K + L) = w^{-1}K + w^{-1}L$
$w^{-1}(KL) = (w^{-1}K)L + \sum_{uv=w} \varepsilon(u^{-1}K)v^{-1}L$
$w^{-1}L^* = \sum_{uv=w} \varepsilon(u^{-1}L^*)(v^{-1}L)L^*$
$a^{-1}(KL) = (a^{-1}K)L + \varepsilon(K)a^{-1}L$
$a^{-1}(L^*) = (a^{-1}L)L^*$
$(uv)^{-1}L = v^{-1}(u^{-1}L)$
Ces formules permettent de calculer tous les quotients à gauche d'un langage.
Lemme fondamental reliant quotient à gauche et automate déterministe :
Pour une exécution $i \to^u q$ de label u, partant de l'état initial, arrivant à l'état q dans un automate déterministe de langage L, le quotient a gauche $u^{-1}L$ est l'ensemble des labels de toutes les exécutions de q vers un état final, càd $u^{-1}L = \{v \in A^* \mid \exists f \in F\ q \to^v f\}$. (Il y a unicité du chemin préfixe $i \to^u q$)
Corollaire : Le nombre de quotients à gauche d'un langage est $\leq$ au nombre d'états de tout automate déterministe complet acceptant ce langage.
Théorème. Un langage rationnel est fini ssi il a un nombre fini de quotients à gauche.

L'automate minimal d'un langage rationnel $\boldsymbol{L}$ est l'automate $\boldsymbol{M_L}$ défini sur l'alphabet A par $Q = \{u^{-1}L : u \in A^*\}, I = \{L\}, F = \{u^{-1}L : u \in L\}, E = \{u^{-1}L \to^a (ua)^{-1}L : u \in A^*, a \in A\}$
L'automate minimal d'un langage accepte son langage. (par déf et récurrence sur longueur mot de L).
L'automate minimal d'un automate est l'automate minimal de son langage $M_{L(M)}$
Un **préordre est régulier** ssi $\forall u, v, u', v' \in A^*\ \ u \leq u'$ et $v \leq v' \Rightarrow uv \leq u'v'$
Un langage est rationnel ssi c'est un idéal d'un bon préordre régulier.

1.7.2. Congruence de Nérode.

Une **congruence sur un automate déterministe complet**, est une relation d'équivalence sur l'ensemble d'états Q telle que $q \sim q' \Rightarrow (q \in F \Leftrightarrow q' \in F)$ et $q \sim q' \Rightarrow \forall a \in A\ \ qa \sim q'a$. Autrement dit, chaque classe d'une congruence n'a que des états finaux ou que des états non finaux, et la congruence est compatible avec les transitions de l'automate. Rien ne concerne l'état initial dans cette définition.
Le **quotient d'un automate $\boldsymbol{M = (Q, A, E, \{i\}, F)}$ par une de ses congruences** $\sim$ est l'automate $\frac{\boldsymbol{M}}{\sim} = \left(\frac{Q}{\sim}, A, E', \{[i]\}, [F] = \{[f] : f \in F\}\right)$ avec $E' = \{[q] \to^a [qa] : q \in Q, a \in A\}$
L'automate quotient $\frac{M}{\sim}$ est encore déterministe car une classe $[qa]$ ne dépend que de la classe de q.
L'automate quotient $\frac{M}{\sim}$ accepte le même langage que celui de l'automate M.
La congruence de Nérode d'un automate déterministe complet est définie par $q \sim_N q'$ ssi $\big(\forall w \in A^*\ (qw \in F \Leftrightarrow q'w \in F)\big)$. La congruence de Nérode est une congruence d'automate.
Théorème. L'automate minimal d'un automate est son quotient par Nérode : $M_{L(M)} = \frac{M}{\sim_N}$

1.7.3. Calcul de l'automate minimal. On cherche à calculer sa congruence de Nérode. 1e méthode : $O(n^2)$, 2e méthode : $O(nlog(n))$ diviser pour régner -> étonnant dans ce cadre.
Méthode itérative. TODO $O(n^2)$
Algorithme de Hopcroft. TODO $O(|A| n \log n)$

1.8. Propriétés de clôture.

1.8.1. Operations booléennes

La classe des langages rationnels est close par union, produit, étoile, complément, intersection, et donc toutes les opérations booléennes. On peut donc étendre le concept d'expression rationnelle à celui d'expression booléenne pour décrire un langage rationnel.
Une expression rationnelle est en particulier une expression booléenne.

1.8.2. Morphisme et morphisme inverse

L'image d'un langage rationnel par un morphisme de mots est un langage rationnel.
L'image réciproque d'un langage rationnel par un morphisme de mots est un langage rationnel.
Exercices : TODO
Tout morphisme de mot a la propriété de sélection, càd que tout langage rationnel L admet un sous langage rationnel K tel que μ est injectif sur K et $\mu(K) = \mu(L)$

1.9. Lemme de l'étoile et ses variantes

Lemme de l'étoile. Un langage rationnel admet une longueur minimale n telle que tout mot du langage d'au moins cette longueur s'écrit uvw avec $v \neq \varepsilon$, et $uv^*w \subseteq L$
Sur un alphabet binaire, le langage $\{0^n 1^n : n \in N\}$ n'est pas rationnel.
Sur un alphabet binaire, le langage des mots dont le nombre de 0 = le nombre de 1 n'est pas rationnel.
Sur un alphabet binaire, le langage $\{0^n 1^n : n \in N\} \cup A^* 10 A^*$ vérifie la condition du lemme de l'étoile mais n'est pas rationnel.
L'ensemble des écritures en base 2 des nombres premiers n'est pas un langage rationnel.
Lemme de l'étoile fort. Un langage rationnel admet un entier n tel que pour tout mot du langage de la forme $uv_1 \dots v_n w \in L$ avec $v_i \in A^+$, alors $\exists i, j, 0 \leq i < j \leq n\ \ uv_1 \dots v_i \big(v_{i+1} \dots v_j\big)^* v_{j+1} \dots v_n w \subseteq L$
Malgré le renforcement de la condition, celle-ci n'est toujours pas suffisante. TODO

Ramsey 1929. Pour tout $(k,m,r) \in N^3$, il existe $N_{k,m,r} \in N$ tel que pour tout :
ensemble E tel que $|E| \geq N$, ensemble C tel que $|C| = m$, fonction $f: P_k(E) \to C$,
Alors il existe $F \subseteq E$ tel que $|F| \geq r$, $\left|f\left(P_k(F)\right)\right| \leq 1$.
les éléments de C sont appelés couleurs, $P_k(E)$ est l'ensemble des parties a k éléments de E. La condition $\left|f\left(P_k(F)\right)\right| \leq 1$ exprime que toutes les parties à k éléments ont la même couleur.
Un **langage L est σ_k** avec $k \in N$ ssi pour toute factorisation $f = uv_1v_2 \dots v_kw$ avec $\forall i\ v_i \in A^+$ alors $\exists i,j, 0 \leq i < j \leq k$ tel que $\forall n \in N\ \ f \in L \Leftrightarrow uv_1 \dots v_i\left(v_{i+1} \dots v_j\right)^n v_{j+1} \dots v_kw \in L$
Un **langage L est σ'_k** avec $k \in N$ ssi pour toute factorisation $f = uv_1v_2 \dots v_kw$ avec $\forall i\ v_i \in A^+$ alors $\exists i,j, 0 \leq i < j \leq k$ tel que $f \in L \Leftrightarrow uv_1 \dots v_iv_{j+1} \dots v_kw \in L$
Ehrenfeucht, Parikh, Rozenberg 1981.
Un langage L est rationnel ssi $\exists k \in N$ L est σ_k ssi $\exists k \in N$ L est σ'_k
1.10. Hauteur d'étoile
La hauteur d'étoile d'une expression rationnelle e notée $h(e)$ se définit par récurrence :
$\forall a \in A\ \ h(a) = h(\varepsilon) = 0$
$h(e+f) = \max\left(h(e), h(f)\right)$
$h(ef) = \max\left(h(e), h(f)\right)$
$h(e^*) = 1 + h(e)$
Les expressions $ab + ba, (ab + a)^*, (ab^*a + b)^*$ sont respectivement d'hauteur d'étoile 0,1,2.
Un même langage peut être décrit par plusieurs expressions rationnelles d'hauteur d'étoile différentes.
Par exemple $(a+b)^*$ et $(a^*b)^*a^*$
La **hauteur d'étoile d'un langage** est la hauteur d'étoile minimale d'une expression rationnelle décrivant ce langage.
Les langages de hauteur d'étoile 0 sont les langages finis.
Pour tout entier $n \geq 1$, sur un alphabet binaire $\{0,1\}$ le langage des mots dont le nombre de 0 est divisible par 2^n est de hauteur d'étoile n.
Il existe un algorithme qui calcule la hauteur d'étoile d'un langage rationnel donne, mais très complexe.
La **hauteur d'étoile peut être étendue aux expressions booléennes** en ajoutant $h(\emptyset) = 0, h(e^c) = h(e)$.
La **hauteur d'étoile généralisée d'un langage** est la hauteur d'étoile minimale d'une expression booléenne décrivant L.
C'est un concept distinct de celui de hauteur d'étoile. $(0+1)^* = \emptyset^c$ est de hauteur d'étoile 1, mais de hauteur d'étoile généralisée 0.
Un **langage sans étoile** est un langage de hauteur d'étoile généralisée valant 0.
Problème non résolu : existe-t-il des langages de hauteur d'étoile généralisée > 1 ?
1.11. Reconnaissance par morphisme
Un **morphisme de monoïdes** est une application f entre deux monoïdes compatible avec la structure de monoïde, c'est-à-dire telle que $f(\varepsilon) = \varepsilon'$ et telle que $f(uv) = f(u)f(v)$.
Une application $\mu: A \to M$ correspond à un morphisme de monoïdes $A^* \to M$ en posant $\hat{\mu}(a_1 \dots a_n) = \mu(a_1) \dots \mu(a_n)$
Un langage L est reconnu par un morphisme de monoïdes $\mu: A^* \to M$ ssi ce langage est une image réciproque du morphisme.
Un langage est reconnu par un monoïde M ssi il y a un morphisme $A^* \to M$ vers ce monoïde qui le reconnait.
Dans ce cas on peut prendre $P = \mu(L)$, donc symboliquement :
(μ, M) reconnait L ssi $\exists P \subseteq M$ $L = \mu^{-1}(P)$ ssi $\mu^{-1}\left(\mu(L)\right) = L$
Si μ n'est pas surjectif, on peut y remédier en remplaçant M par le sous-monoïde $M' = \mu(A^*)$.
Rationalité par morphisme. Un langage est rationnel ssi il est reconnu par un monoïde fini.

(Sens indirect : l'automate d'états le monoïde M, d'états final $\mu(L)$ de transitions $m \to^a m\mu(a)$, reconnait L.)
Une partie $M \subseteq X$ est un sous-monoïde de X ssi l'identité $M \to^{id} X$ est un morphisme de monoïdes.
Exemple : L'ensemble des parties rationnelles de A^* est un sous-monoïde de $P(A^*)$
Un **langage est préfixe** ssi $L \cap LA^+ = \emptyset$ cad ssi un préfixe strict d'un mot de L n'est jamais un mot de L.
L'ensemble des langages préfixes est un sous-monoïde de $P(A^*)$
Dans ce monoïde $K_1 \dots K_m = L_1 \dots L_n \Leftrightarrow m = n$ et $\forall i \in \{1, \dots, m\}\ K_i = L_i$
Un tel monoïde est dit libre car isomorphe a un monoïde de la forme B^*.
Un **quotient d'un monoïde** M correspond a un monoide Q muni d'un morphisme surjectif de $M \to Q$.
Un monoïde $\boldsymbol{M}$ **divise un monoïde** $\boldsymbol{M'}$ et on écrit $\boldsymbol{M} \lhd \boldsymbol{M'}$ ssi M est un quotient d'un sous-monoïde M_1 de M'. Cela correspond au diagramme : $\begin{matrix} M_1 & \to^i & M' \\ \downarrow_\pi & & \\ M & & \end{matrix}$
La relation de division est une relation d'ordre sur l'ensemble des monoïdes finis.
On perd l'antisymétrie de cette relation sur les monoïdes infinis.
Une **congruence de monoïde** est une relation d'équivalence sur un monoïde compatible avec le produit càd telle que $x \sim x', y \sim y' \Rightarrow xy \sim x'y'$. Condition suffisante pour la compatibilité d'une relation d'équivalence : $y \sim y' \Rightarrow \forall x \forall z\ xyz \sim xy'z$. Une congruence de monoïde permet de bien définir le **monoïde quotient par la congruence** $\frac{\boldsymbol{M}}{\sim}$ muni de la loi quotient $[x][y] = [xy]$.
Le contexte d'un mot $\boldsymbol{y}$ suivant un langage $\boldsymbol{L}$ est l'ensemble $\boldsymbol{C_L(y)} = \{(x, z) \in A^{*2} | xyz \in L\}$
La congruence syntaxique d'un langage $\boldsymbol{L}$ est la congruence de monoïde $\sim_{\boldsymbol{L}}$ définie par
$y \sim_{\boldsymbol{L}} y' \Leftrightarrow C(y) = C(y') \Leftrightarrow \forall x \in A^*\ \forall z \in A^*\ (xyz \in L \Leftrightarrow xy'z \in L)$
Le **monoïde syntaxique d'un langage $\boldsymbol{L}$** est le monoïde libre quotienté par la congruence syntaxique de ce langage $\frac{A^*}{\sim_L}$
Théorème. Un monoïde M reconnait un langage L ssi le monoide syntaxique de ce langage divise M.
Le monoïde syntaxique d'un langage rationnel L est égal au monoïde des transitions de l'automate minimal de ce langage L. Utile pour le calcul du monoïde syntaxique.
Exemple : TODO
Soit σ une substitution de A^* dans B^* et soit $K \subseteq B^*$ un langage rationnel. Les deux langages $\{w \mid \sigma(w) \cap K \neq \emptyset\}$ et $\{w \mid \sigma(w) \subseteq K\}$ sont rationnels.
1.12. Langages sans étoile (application du monoïde syntaxique).
TODO
1.13. Compléments
1.13.1. Conjecture de Cerny