MINISTÈRE DE L'ÉDUCATION NATIONALE, DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE

ANNALES ISLAMOLOGIQUES

en ligne en ligne en ligne en ligne en ligne en ligne en ligne en ligne en ligne en ligne

AnIsl 44 (2010), p. 5-38

André Jaccarini

# De l’intérêt de représenter la grammaire de l’arabe sous la forme d’une structure de machines finies.



**Dernières publications**

9782724711622 *BIFAO 126*
9782724711059 *Les Inscriptions de visiteurs dans les Tombes thébaines* Chloé Ragazzoli
9782724711455 *Les émotions dans l’Égypte Ancienne* Rania Y. Merzeban (éd.), Marie-Lys Arnette (éd.), Dimitri Laboury, Cédric Larcher
9782724711639 *AnIsl 60*
9782724711448 *Athribis XI* Marcus Müller (éd.)
9782724711615 *Le temple de Dendara X. Les chapelles osiriennes* Sylvie Cauville, Oussama Bassiouni, Matjaž Ka?i?nik, Bernard Lenthéric
9782724711707 *????? ?????????? ??????? ???? ?? ???????* Omar Jamal Mohamed Ali, Ali al-Sayyid Abdelatif
*??? ???? ?? ??????? ??????? ?? ???????? ???????? ????????????*
*?????????? ????????? ?????? ?????? ?? ??? ??????? ?????:*

ANDRÉ JACCARINI

# De l'intérêt de représenter la grammaire de l'arabe sous la forme d'une structure de machines finies

Cette étude est à la fois un développement théorique de l'article présenté à Malte au congrès Lrec 2010, un bilan critique de notre travail dans le domaine du traitement automatique de l'arabe entrepris dès les années 1980 et une présentation d'un programme d'études en linguistique arabe à court, moyen et long terme.

Nous nous situons d'emblée dans le cadre de la théorie des « machines abstraites ». Les automates et les transducteurs, auxquels nous avons recours pour décrire la langue arabe, sans la donnée d'un lexique défini *a priori*, constituent une catégorie importante de ces machines, équivalentes aux machines de Turing*. L'une des idées importantes de cet article est que l'arabe, voire les langues sémitiques en général, présente un niveau d'« algorithmicité » / grammaticalité de surface, particulièrement élevé (à condition toutefois de clarifier la relation «procédure/structure»). Les caractéristiques *structurales* de l'arabe sont aussi des caractéristiques *algorithmiques*. Cette dualité est aisément traduisible dans le cadre de la théorie algébrique des automates et offre des perspectives applicatives intéressantes (la correspondance se fait dans les deux sens), facilement spécifiables mathématiquement. L'unification théorique, opérée grâce à la théorie algébrique des automates offre la possibilité d'établir solidement les Études arabes dans l'univers des connaissances qui est aujourd'hui le nôtre, à savoir celui de Turing*. Nous nous inscrivons ainsi dans une double tradition française : d'abord celle du traitement de

Je remercie vivement Sylvie Denoix, directrice des études et responsable de la section d'arabe de l'Ifao, d'avoir lu de très près cet article.

* Les astérisques renvoient à des articles fiables de Wikipédia.

l'arabe par informatique qui, ne l'oublions pas, a été lancé par David Cohen (1960) et ses amis de l'institut Blaise-Pascal, et ensuite celle de la tradition *algébrique* en informatique, initiée par M.P Schutzenberger (cf. *infra*).

La réactualisation du logiciel *Sarfiyya*** permet à l'équipe de prouver l'efficacité du traitement de l'arabe par automates. On trouvera dans le site, associé à l'article, une présentation des résultats d'un extracteur de citations en arabe ainsi que la démonstration de la détection de certaines relations discursives. Les résultats sont très encourageants. Nous démontrons ainsi que notre travail ne se limite pas à la morphosyntaxe mais s'étend également à la sémantique. Nous espérons dans un proche avenir être en mesure d'établir sérieusement des protocoles de comparaison avec d'autres expériences pour pouvoir avancer scientifiquement dans le domaine de l'évaluation ; ce qui suppose un minimum de coopération.

Notre objectif à court terme est de constituer une bibliothèque minimale d'opérateurs sur lesquels peuvent s'effectuer un certain nombre d'opérations en vue de synthétiser de nouveaux opérateurs selon nos besoins. En ce qui concerne les applications, nous nous intéressons à la recherche d'information et à son filtrage. À plus long terme, nous sommes intéressés par la construction d'une banque d'opérateurs (les *tokens***), laquelle serait aussi une base grammaticale facilement interrogeable.

Comme il s'agit d'une représentation *algorithmique* de la grammaire arabe nous insistons sur l'aspect essentiel et fondamental de la notion d'algorithme en rappelant les conditions historiques de son émergence (§6). L'approche algorithmique de l'arabe, qui est un sujet d'actualité, nécessite la définition d'un cadre scientifique cohérent. La recherche de cohérence théorique a toujours été associée à la recherche « expérimentale » d'algorithme. Le rappel de certains résultats fondamentaux concernant les propriétés intrinsèques des machines de Turing (existence de machines dites universelles, équivalences, problème d'indécidabilité, notamment en ce qui concerne le problème de l'arrêt, …) ne constitue pas simplement une énumération de certaines « curiosités » intellectuelles. Il s'agit surtout de caractéristiques logico-algébriques ayant une incidence directe au niveau de l'organisation de notre travail de programmation et de la construction d'un modèle algorithmique et grammatical de la langue arabe.

## 1. Historique du champ disciplinaire

Le programme Tala (traitement par automates de la langue arabe), piloté par la section des études arabes de l'Ifao, ainsi que par les services informatiques de l'Ifao et de la Mmsh, est le fruit de la collaboration entre plusieurs chercheurs de formations et de cultures différentes. Aussi est-il nécessaire de préciser le cadre logique dans lequel ces recherches se développent. L'approche purement algorithmique et minimaliste qui a caractérisé jusqu'ici notre approche de la grammaire arabe me conduit aussi à rappeler un certain nombre de résultats fondamentaux.

** Les doubles astérisques renvoient au site htpp://automatesarabes.net
Ce site est actuellement « passif », au sens où les résultats figurent dans des fichiers PDF ; mais les automates pourront être mis directement en action par les utilisateurs dans la future version de l'outil en ligne *Kawâkib Pro* (cf. *infra*, l'article de Christian Gaubert).

En effet, l'un des arguments importants dans cet article est que l'approche abstraite, dans le domaine du traitement automatique de l'arabe, est plus efficace que l'approche pragmatique et cela, de prime abord, peut sembler paradoxal. En effet plutôt que de programmer des applications particulières (ce que j'appelle la programmation *ad hoc*), il est plus rentable de construire un système plus général (et plus abstrait) qui puisse les engendrer. La situation actuelle en est une parfaite illustration. La présentation historique de la notion d'algorithme, qui figure ci-dessous (§6), a l'avantage important de faire apparaître clairement la fécondité du point de vue abstrait et notamment celle de machine abstraite (dont font partie les automates). Rappeler que l'informatique, c'est-à-dire la « civilisation » dans laquelle nous vivons, a été directement engendrée par le « théorème métamathématique de l'énumération universelle » (1936), avant même l'invention de l'ordinateur (1944), me semble intéressant pour illustrer cette « manière de voir ». Cette présentation de l'histoire de la logique ferait ainsi apparaître l'informatique comme une branche de la « métamathématique appliquée » (ce qui peut sembler quelque peu « oxymorique »). L'informatique théorique n'a pas résulté d'une théorisation de la pratique informatique, mais, à l'inverse, c'est l'ordinateur qui est né de la résolution d'un problème important de logique qu'énonça David Hilbert* dans son fameux « programme » qui avait pour objectif d'établir la consistance de l'ensemble de l'édifice mathématique, à la suite de l'« inquiétude » suscitée par l'apparition de certains paradoxes logiques et les constructions « transfinitaires » de Georg Cantor.

Pour en venir à la langue arabe, le programme de recherche sur les automates arabes a pour origine un programme écrit, en APL* vers 1980[1], de tri de formes arabes, qui attribuait à chacune d'entre elle une valeur numérique représentant approximativement le « degré de complexité » de cette forme relativement à l'opération d'extraction de la racine. Les critères retenus étant la « solidité » des consonnes, leurs positions relatives et la longueur du mot. Les opérations mentales consistant à extraire la racine arabe d'un mot (nécessaire pour accéder à la quasi-totalité des dictionnaires arabes depuis Sibawayh) me semblaient pouvoir donner lieu à un calcul (*i.e.* un programme) rendu possible grâce à la structure particulière de l'arabe (voire des langues sémitiques). J'attire l'attention sur le fait que ce qui me semblait une caractéristique importante de l'arabe est que cette « opération mentale » puisse être rendue par algorithme et non par un algorithme particulier, autrement dit que cette opération fût récursive, pour s'exprimer comme les logiciens, ce qui n'empêche pas cependant l'ambiguïté de se produire, surtout en cas de « dévoyellisation » (= dévocalisation), puisque, comme nous le savons l'écriture standard de l'arabe est de nature sténographique. Pour éviter le renouvellement d'un malentendu qui a pu se produire, y compris avec les arabisants, la mise en évidence d'un algorithme particulier (par exemple, un analyseur pour automate fini, ou pour transducteur, ou bien un réseau de neurones, un programme APL, ou bien même un algorithme sans structure particulière, etc.) ne signifie aucunement que notre esprit procède de la manière décrite par l'algorithme spécifié (et mis en œuvre). Nous échappons ainsi à la critique de réductionnisme fruste, de contingence, de « simulation » gratuite sur un ordinateur, etc. Quel

1. Voir *La Recherche*, n. 111, mai 1980.

intérêt par exemple, pourrait-on se demander, de simuler sur machine ce que l'esprit humain accomplit bien plus facilement (quand il connaît l'arabe [2] !), surtout quand se poseront les redoutables problèmes de complexité, d'ambiguïté, de bruits, de silence dont on était d'ailleurs conscient dès le départ[3]. De manière résumée : je cherchais à établir simplement la possibilité de résolution du problème d'accès au dictionnaire par algorithme, ce qui en soi représente un intérêt épistémologique (et même cognitif, mais à condition de ne pas être naïf) et d'en tirer un certain nombre d'avantages pratiques (nous y reviendrons). Par ailleurs, pour répondre à d'autres critiques, mais qui en un certain sens pouvaient rejoindre la précédente : celles de plaquer – artificiellement – sur la langue un formalisme étranger à cette dernière et à son mode de fonctionnement (à supposer que la langue « fonctionne ») – que ce soit la théorie des automates, que nous avons privilégiée, ou tout autre formalisme comme le lambda calcul : Lisp (ou un autre langage applicatif), les réseaux de neurones formels (fort à la mode dans le TAL à la fin des années 80 et dans les années 90), etc. – nous rappelons au paragraphe 6, les théorèmes d'équivalences qui sont très importants et même impressionnants (surtout pour ce qui est de celui de W.McCulloh* et Pitts* (1943, voir *infra*).

Ces résultats, non seulement répondent fortement aux critiques potentielles (pourquoi ce formalisme plutôt qu'un autre, ce langage de programmation plutôt qu'un autre, etc.), mais renforce considérablement, en relevant son « statut épistémologique », l'idée même d'activité intellectuelle « mécanisable ». Par ailleurs il me semble important de rappeler aussi qu'en ce qui concerne « la racine », mon point de vue est purement « phénoménologique », je constate simplement – d'un point de vue « mathématique » – que, dans un sous-ensemble important de la langue arabe (duquel est exclu, dans un premier temps, la morphologie non saine), toute permutation, dans une phrase, des triplets consonantiques – obtenus à partir de filtres déduits par « inversion » d'une liste donnée d' « opérateurs » appelé « schèmes » – n'affecte que faiblement la grammaticalité de la phrase. De la même manière on peut aussi remarquer que, pour ce qui est des mots graphiques (formés de lexèmes « agglutinés » à d'autres éléments), les règles de concaténation (respectivement de segmentation) demeurent invariantes par changement de « racines », d'où la notion de « congruence » (voir *infra*).

Autrement dit la « racine » était définie relativement à ces opérations « abstraites » et non dans l'« absolu » ; il s'agissait donc en quelque sorte d'une définition « en creux ». Il était nécessaire de préciser ce point, car la question a donné lieu à trop de débats inopportuns sur la « réalité » de la racine, notamment avec les « historiens de la langue ».

2. Pour les apprenants, on pourrait organiser les dictionnaires de manière alphabétique (exemple : *El-Mawred*), mais cela ne résoudrait pas pour autant le problème dit d'agglutination qui est important en arabe c'est-à-dire celui de la concaténation de plusieurs éléments dont plusieurs ne peuvent constituer des mots graphiques (entités séparées par deux blancs). La réorganisation des dictionnaires ne suffirait donc pas, loin de là, à résoudre le problème et il y faut naturellement ajouter celui, non négligeable, de la « dévocalisation » (dévoyellisation). Signalons enfin que l'organisation classique (par racine) a une justification logique et « structurale » puissante (voir à ce propos Jaccarini, « Approche algorithmique de la grammaire arabe » ; chapitre 2).

3. Voir Aubebert, Jaccarini, « À la recherche du *Ḥabar* » **.

## 2. De l'utilisation des automates

Un aspect important est le distinguo établi entre le système de base de la morphologie, ou morphologie saine, et le système général, auquel on peut passer à partir du premier au moyen d'une transduction finie ; cette dernière pouvant se définir comme une relation fonctionnelle, ce qui, en soi, constitue une « formalisation » et « donne un sens précis » au principe selon lequel la morphologie générale de l'arabe est obtenue par « déformation » d'un système originel « stable » et qu'elle se trouve en quelque sorte « lissée » pour répondre aux règles de compatibilité phonologique ; Fleish parle de « système de départ et de système d'arrivée[4] » ; d'autres linguistes évoquent un « principe de régularité » mais qu'ils ne précisent pas plus que ne le fait Fleish dans son étude assez complète mais non formalisée. La formalisation par « transduction » est une modélisation puissante qui constitue à la fois une justification de l'emploi de la théorie des automates pour rendre compte des particularités de la morphologie arabe et surtout de son système particulier d'écriture (concaténation d'éléments non nécessairement « lexicaux[5] »). Cette justification constitue l'un des thèmes de cet article, qui se veut aussi un article programmatique pour les *études arabes* dans le domaine de la linguistique et plus spécifiquement de la grammaire de l'arabe à l'heure de Turing*.

Le principe d'invariance constitue une première illustration de la fécondité de la « méthode abstraite » car il est alors possible de déduire directement – dans certains cas – des algorithmes optimisés[6].

Par ailleurs, la modélisation par transduction du système morphologique général à partir du système de base fournit la possibilité mathématique de minimaliser les « actions déformantes » auxquelles fait allusion par exemple Fleish, dans les chapitres susmentionnés (d'un point de vue « procédural », les « règles » de « déformation » (règles phonologiques) peuvent se définir comme simple effet de bord d'un automate sous-jacent; d'un point de vue structural, elles peuvent se définir par des « équations » élémentaires).

Pour en revenir donc à la notion d'« algorithmicité » ou encore d' « effectivité » de l'opération mentale d'extraction de la racine qui, au départ, ne m'avait été inspirée que par simple introspection (s'observer lorsque l'on cherche dans le dictionnaire), elle fut considérablement confortée par la découverte du travail remarquable et pionner de David Cohen[7]. Nous aurons

4. Cf. dans Bibliographie**, le chapitre 1 de ref. 3 : Christian Gaubert, « Stratégie et règles minimales pour un traitement automatique de l'arabe ».

5. Par « éléments non lexicaux » nous entendons ici, ceux qui ne constituent pas des entrées lexicales, par exemple certaines prépositions « *li* », « *bi* », « *ka* », l'interrogatif : « *Hamza de l'istifhām* », la particule de corroboration, les coordonnants, etc. ; citons dans cette optique, comme cas particulier intéressant, la concaténation de deux éléments non lexicaux produisant un mot graphique licite ; exemple : *bihi*, *lahum*, etc.

6. Cf. Jaccarini, « Modélisation linguistique et théorie des automates. Méthodes de l'obtention de l'algorithme optimal par variations des grammaires. Application à l'arabe »**, chapitre 4.

7. « Vers un traitement automatique de l'arabe » ; cette étude, effectuée en 1960 (et relue pour ce qui est de l'aspect « informatique » par André Lentin), a été publiée dans « Études de linguistique arabe et sémitique ». Je dois à Maurice Gross, non seulement de m'avoir mis en contact avec D. Cohen vers 1978, mais d'avoir également attiré mon attention sur le travail de Greenberg concernant les compatibilités entre groupes entiers

AnIsl 44 (2010), p. 5-38 André Jaccarini
De l'intérêt de représenter la grammaire de l'arabe sous la forme d'une structure de machines finies.

l'occasion de revenir plus bas sur cette étude lorsque nous énumérerons les avantages de la représentation des données morphologiques arabes par des automates, sans recours à un lexique (§ 7, § 8 et § 9).

Inversement, toute spécificité structurale me semble intéressante à expliciter en vue de l'« algorithmisation » c'est-à-dire le traitement automatique.

## 3. « Algorithmicité » et forte « grammaticalité » : une spécificité de la langue arabe ?

La production foisonnante mais trop anarchique dans le traitement automatique de l'arabe de ces dernières années a affecté son image. Par ailleurs, l'excès de bruit qui règne dans ce domaine brouille sérieusement la communication scientifique et altère surtout la qualité de l'indispensable dialogue entre les disciplines impliquées (linguistique, mathématiques, informatique théorique et appliquée, psycholinguistique, etc.). Le TAL est à la fois riche et désorganisé. L'« hyperactivité » qui règne dans le monde informatique – où la pérennité des applications ne semble pas être le souci majeur des acteurs les plus importants, émetteurs de normes qui, souvent, d'un point de vue strictement scientifique, peuvent sembler quelque peu contingentes – s'est communiquée naturellement au monde du TAL. La recherche d'une illusoire « efficacité » sans souci de la cohérence est malheureusement trop souvent la règle. La trop grande production d'applications *ad hoc* se fait alors souvent au détriment d'une recherche algorithmique plus structurée[8].

Pour répondre par l'affirmative à la question posée dans le titre de ce paragraphe, le plus simple serait naturellement de construire un programme ne nécessitant qu'un recours minimal au lexique, voire même de se dispenser totalement de ce dernier (principe du dictionnaire vide **). Or, force est de constater que ce passage à la limite ne provoque aucune explosion d'ambiguïtés si on compare les analyseurs produits avec d'autres fonctionnant à l'aide d'un lexique. Ouerserghini** (équipe Diinar, Lyon 2) ne fournit pas de mesures exactes de celles-ci, mais reconnaît qu'elle demeure très élevée. Il va jusqu'à annoncer une moyenne de 6 ; néanmoins nous ne disposons d'aucun décompte précis.

Malgré les bruits et les silences, l'ambiguïté demeure confinée ; malgré ce que semble suggérer, à première vue, la présentation de Christian Gaubert dans son étude : « Règles et Stratégies

de consonnes « homorganiques » dans les triplets R1 R2 R3 constituant les « racines » en sémitique. Rappelons, par exemple, que non seulement R1 ne peut jamais être identique à R2 mais que les deux premières radicales ne peuvent jamais appartenir au même groupe homorganique (deux laryngales par exemple). Cette étude porte sur environ 6 000 racines et recèle des statistiques précieuses.

8. Cet état de choses présente de graves inconvénients car on se retrouve alors en présence de logiciels qui, pour être spectaculaires et fort ergonomiques, sont souvent non extensibles et totalement inadaptés pour peu que le contexte dans lequel ils ont été conçus ait quelque peu évolué. Tout mépris d'universalité et de pureté méthodologique occasionne, généralement, des coûts exorbitants. Des programmes importants contenant des centaines de milliers de lignes de code (voire des millions) ont dû être entièrement réécrits pour s'adapter à l'évolution du contexte informatique.

minimales pour un traitement automatique de l'arabe » **. Dans ce travail, l'ambiguïté n'est pas toujours hiérarchisée. Un effort supplémentaire d'ergonomie serait de nature à diminuer « l'impression » de forte densité de cette dernière. Cependant, la question d'ergonomie est relativement secondaire au niveau actuel de notre recherche, car nous n'en sommes pas encore à définir un produit final prêt à être commercialisé.

Il nous a toujours semblé que le recours au contexte, plus que l'appel à un lexique, était de nature à faire le plus baisser l'ambiguïté. C'est la thèse principale que Claude Audebert et moi-même avons défendue dans : « À la recherche du *Ḥabar*. Outils en vue de la constitution d'un programme d'enseignement de l'arabe assisté par ordinateur »**. Nous y reviendrons.

L'étude de Christian Gaubert ne tient, à dessein, pas compte du contexte, et porte sur de l'arabe non « voyellé » (non vocalisé). Il est tout à fait normal, que, dans ces conditions draconiennes (études de cas limites), l'ambiguïté atteigne un certain niveau. C'est plutôt le cas inverse qui aurait suscité de sérieux doutes sur l'ensemble de la méthode, et sur l'honnêteté de l'entreprise. Cette étude se voulait critique ; elle se proposait de « mesurer » la pertinence de l'hypothèse du travail sans lexique, en en soulignant à la fois forces et faiblesses.

Relativement à l'étude de Christian Gaubert, il reste quelques questions importantes à évoquer.

1. S'il s'agit de « mesurer » le niveau, supposé élevé, d' « algorithmicité » / grammaticalité de l'arabe, de nature à renforcer l'hypothèse de l' « existence » d'un langage squelette de l'arabe[9] (existence d'un langage quotient, *i.e.* dans quelle mesure la langue demeure-t-elle invariante par changement de racine ? L'idée est naturellement à nuancer, mais c'est le « schéma de pensée » qui nous intéresse ; un « schéma » dont la fécondité serait peut-être, dans un premier temps, plus facilement perçue par un physicien – ou un mathématicien – que par un arabisant de tradition exclusivement littéraire), Christian Gaubert se situe – tout comme je l'avais fait précédemment en recourant au Lisp de l'Inria[10] – dans le cas le plus défavorable : celui du *parsing*. Les grammaires formelles (dont celles de Chomsky) sont généralement conçues comme des mécanismes ; ou simplement des spécifications de ces mécanismes sans références à leurs mises en « exécution » ; auquel cas on pourrait leur associer des systèmes d'équations (aspect structural : voir notamment le Gross et Lentin) ; or ces procédures sont généralement perçues dans le sens de la génération et ce n'est point un hasard si l'on a parlé de grammaire *génératives* (et de *générativisme* ; l'hypothèse « cognitiviste » de Chomsky étant la faculté humaine « innée » d'engendrer le langage, *infini*). Or l'opération de « lecture », ou plus généralement de « réception », est une opération inverse. Dans le cas général (celui des grammaires transformationnelles) cette opération s'est avérée très difficile, susceptible même de provoquer des explosions combinatoires, ce qui a entraîné d'ailleurs dans les années 70, le courant d'un certain « réalisme linguistique » qui préconisait notamment un rapprochement de la modélisation linguistique avec la psycholinguistique. C'est ce courant qui a favorisé un grand intérêt pour les ATN (Augmented Transitive Network) de Woods qui ne sont en fait

9. Pour ce qui est du « statut ontologique » du langage L/RAC, nous y reviendrons dans un autre contexte.
10. Cf. Audebert, Jaccarini, « Méthode de variation de grammaire et algorithme morphologique » **.

que des *transducteurs*. Ces transducteurs peuvent être formalisés dans le cadre des grammaires de Knuth* (à attributs « synthétisés » et « hérités » ; cf. *infra*).

En effet, l'opération inverse d'analyse recèle de mauvaises surprises (à tous les niveaux), si bien que – à la différence des langages informatiques que l'on définit toujours en sorte qu'il présente la caractéristique d'être déterministe** – l'un des « attributs » les plus importants du langage naturel est justement son ambiguïté formelle (au sens mathématique).

2. L'étude de Chr. Gaubert se limite – toujours à dessein – aux seuls « automates finis ». On s'interdit de leur adjoindre la moindre mémoire. Les filtres et les circuits correcteurs opèrent *a posteriori* mais on ne s'interdit pas l'adjonction de « micro lexique » (qui ne sont pas en fait des lexiques mais de simples listes de racines attestées ou de schèmes). C'est une opération de « débruitage » nécessaire pour rendre compte de la « faisabilité » : un effort de visibilité qui ne devrait toutefois pas brouiller la méthode (l'objectif étant de comparer la chute du bruit par mécanisme d'interprétation ou de contrôle *a posteriori* : sur ce point la discussion est encore ouverte).

3. Cette étude montre en creux la nécessité de recourir au contexte et corrobore donc l'hypothèse développée dans « À la recherche du *Ḥabar* »**, dont nous allons parler au paragraphe suivant. Cet aspect n'a peut-être pas été suffisamment souligné, d'autant plus que les données statistiques très riches que recèle l'étude de Chr. Gaubert nécessitent un travail considérable d'interprétation, qui constituerait à lui seul une thèse consistante.

## 4. L'hypothèse d'un processeur *abstrait* et *idéal* de décodage : la méthode descendante dite « *top-down* »

En 1986, lors de la publication de l' article « À la recherche du *ḥabar*. Outils en vue de la constitution d'un programme d'EAO de l'arabe »**, la situation était la suivante :

1. Des modules APL* d'une très grande concision avaient été écrits en vue de l'analyse morphologique et de la constitution de programmes de tri (APL est un langage d'opérateurs qui peuvent effectuer des tâches globales sur des structures algébriques « généralisées » : tableaux à plusieurs dimensions sur lesquels peuvent être appliquées des opérations de nature très diverses : produit interne et externe mettant en jeu différents types d'opérateurs logiques algébriques, des rotations, des inversions, etc.). La nécessité, pour des raisons d'optimisation, liée à la nature même du langage APL, amène naturellement à regrouper dès le départ les formes à décoder en groupes relativement homogènes présentant des similitudes, relativement aux différentes opérations à effectuer. Ainsi se trouvent naturellement isolées, d'entrée de jeu, des formes ne possédant pas de racines : des mots figés, lesquels obéissent néanmoins à certaines règles d'agglutination ;

2. Or, il se trouve que ces mots, qui sont en quelques sortes des « atomes » (ils sont morphologiquement indécomposables) recoupent l'ensemble des mots considérés comme peu « informatifs » dans une optique d'informatique documentaire ; ces derniers sont d'ailleurs regroupés dans des « anti-dictionnaires ». Ils coïncident aussi avec ce que les linguistes appellent,

AnIsl 44 (2010), p. 5-38 André Jaccarini
De l'intérêt de représenter la grammaire de l'arabe sous la forme d'une structure de machines finies.

faute de mieux, les « mots-outils »[11]. Ce constat amène à faire le lien avec l'étude historique de Claude Shanon*: « Mathematical Theory of Communication » où se trouve définie la notion importante d' « entropie d'un langage » (la notion d'entropie « passe » ainsi de la thermodynamique à la théorie de la communication et du langage).

L'idée est simple : dans une suite de symboles, de longueur quelconque, la valeur « informative » d'un symbole donné est inversement proportionnelle à sa probabilité d'occurrence (en fonction du segment initial déjà lu). S'il existe par exemple un « procédé » (un algorithme) pour engendrer la succession des symboles (c'est le cas par exemple de la suite infinie et non périodique de décimales définissant le nombre π) la valeur informative de chacun de ces symboles sera strictement nulle. Si en revanche, il n'en existe aucun (mais comment démontrer « effectivement » l'absence : d'où la difficulté, par exemple, de définir la fonction « random » en informatique : « engendrer » du hasard est antinomique) la valeur informative sera maximale. Les niveaux de liberté, dans un flux informationnel, varient au fur et à mesure que s'effectue la lecture. Étant donné la valeur intrinsèquement « informative » de la langue, il est évident que son « entropie » ne peut être nulle (absence de liberté) ; elle ne peut en revanche être maximale (aucune règle). Shanon a calculé que l'entropie de l'anglais était égale à ½ ; ce qui concrètement signifie que, dans tout message correctement rédigé en anglais, un symbole sur deux est superflu et que l'expression correcte des contraintes (c'est-à-dire la spécification de la grammaire) pourrait ainsi diviser par deux la longueur des messages[12]. Il corrobore son calcul, probabiliste, par une remarque fort astucieuse : il note que s'il est possible de concevoir *ad infinitum* des mots croisés en anglais en dimension deux, cela n'est plus possible en dimension trois ;

3. Pour en revenir au « décodage » de l'arabe, on aurait pu attaquer le problème à différents niveaux. Le principal intérêt des échanges avec Claude Audebert et du travail (*i.e.* l'expérimentation mentale[13]) fait en commun, a été d'établir des correspondances entre ces niveaux. S'intéresser aux éléments qui, dans un flux d'information, font le plus baisser le « niveau entropique », *i.e.* induisent les contraintes les plus importantes, présente les intérêts suivants, qui, quoique appartenant à des champs disciplinaires différents, sont en quelque sorte « homomorphes » :

– Ces éléments induisent des contraintes plus importantes (bien que non encore mesurées mathématiquement c'est-à-dire statistiquement : travail de quantification qui reste à faire pour asseoir « scientifiquement » la méthode, mais non nécessaire pour convaincre un arabisant) que les autres lexèmes ; ils peuvent donc être vus soit comme des révélateurs de structures, des « indicateurs » de la présence de « sous arbres » particulièrement intéressants dans l'arbre général (ou indicateur syntagmatique, pour avoir recours à la terminologie chomskyenne) par lequel on définit la structure générale de la phrase ; soit comme induisant des attentes plus fortes (des probabilités plus élevées) de certaines sous structures, d'où l'intérêt (et c'est là un

11. On remarquera aussi que dans la « procédure » précédemment évoquée, proposée dans l'étude de David Cohen, ces « mots-outils » sont à reconnaître en premier.

12. Signalons que Shanon travaillait pour le compte d'une société de téléphone !

13. Au sens où, en philosophie analytique, on parle d'expérience de pensée (exemple d'expérience de pensée : le célèbre paradoxe EPR (Einstein, Podolsky, Rossen) censé réfuter la mécanique quantique (1935).

AnIsl 44 (2010), p. 5-38 André Jaccarini
De l'intérêt de représenter la grammaire de l'arabe sous la forme d'une structure de machines finies.

aspect qui intéresse la didactique, *i.e.* la cognition) de les repérer en premier afin d'émettre des hypothèses, plus ou moins, probables sur les structures attendues, plutôt que d'analyser linéairement les formes rencontrées lors de l'opération de lecture (avec une production considérable de bruit et d'ambiguïté) et un résultat très hasardeux. Au cours d'expériences en didactique que Claude Audebert a effectuées au Deac – qu'elle venait de fonder – entre 1982 et 1986, il a été remarqué que les étudiants (d'un niveau de 200 à 250 heures) qui lisent en avançant le plus possible dans la phrase (sans vraiment la comprendre), mais qui émettent en cours de route des hypothèses sur sa structure (notamment syntaxique) pour ne chercher dans le dictionnaire la signification des lexèmes que dans un deuxième temps et dans un ordre non linéaire, réussissent mieux et plus rapidement à décoder correctement la phrase que ceux qui procèdent mot à mot ;

– Ces stratégies, pour peu que l'on réussisse à les formaliser (même de manière incomplète et imparfaite), font inévitablement penser – autre homomorphie – aux méthodes de *parsing* descendantes (*top-down*) et montante (*bottom-up*) en informatique, dans le domaine de l'analyse du langage naturel [14] ;

– La difficulté d'accéder au dictionnaire (c'est le côté paradoxal de l'apprentissage de l'arabe) nous oblige à réfléchir à un modèle didactique.

Sur le plan épistémologique, ce travail suppose plusieurs hypothèses sur les machines de Turing impliquées**, à différents niveaux, et dont l'existence est *supposée*. Cette méthodologie intrinsèquement descendante mérite un exposé indépendant [15].

14. Il faut rappeler que l'analyseur « classique » pour grammaire *context free*, d'Earley (1968-1970) est un compromis entre analyses *top-down* et *bottom-up* et que ce fait « calculatoire » peut *aussi* induire, en retour, des idées sur le compromis « humain » idéal entre analyse montante et descendante. Cet analyseur est en en effet quasi optimal : il opère, dans le cas le plus défavorable, en $n^3$ ; il a été démontré par Vallient – en utilisant des méthodes « monstrueuses » d'analyse numérique, concernant l'optimisation du calcul matriciel – qu'il existait une limite $n^{2.75}$.

15. On peut se représenter cette « *tentative de modélisation* » par le cas *limite* suivant : celui d'un Martien ne disposant que d'une machine de Turing* à mémoire non limitée (le ruban de mémoire dans les deux sens), et ne possédant pas par conséquent de « connaissances » *a priori* : il se trouve en présence d'un corpus de taille indéterminée dont il essaie de dégager les régularités (la structure) mais non le sens, en vue de « spécifier » ce langage par la description d'une autre machine de Turing. En fait, il ne pourrait, dans un premier temps que saisir des fragments de machines de Turing (décrivant certaines régularités de surface du texte) qu'il essaierait d'agréger ensuite en une machine globale (qu'il ne pourra en fait jamais atteindre, mais simplement approcher). De manière encore plus générale (donc abstraite) il serait amené à tester et tenter de hiérarchiser les différents fragments et sous fragments (des machines partielles) en vue de synthétiser une machine globale (mais non figée ; nous reviendrons plus loin sur cet aspect important de non figement de la grammaire ; cf. le paragraphe ci-dessous « pourquoi privilégier la théorie des automates ? »). Il est évident que ce Martien ne dispose pas *a priori* d'un dictionnaire (*i.e.* de données statiques) mais d'intelligence (*i.e.* de capacités de définir et de mettre en œuvre des procédures (attention nous n'avons aucun *a priori* sur l'« intelligence », à ce sujet voir dans Mosconi, « l'hypothèse forte et l'hypothèse faible de l'intelligence artificielle » dans « La constitution de la théorie des automates »**).
La mise en parallèle avec l'apprenant (dont l'intelligence lui serait en quelque sorte communiquée par l'enseignant) est instructive et peut donner lieu à des expériences de pensée dont toute la deuxième partie de l'article mentionnée au début de ce paragraphe : « À la recherche du *Ḥabar* »** donne une idée assez claire.

Dans cet article, on suppose résolu le problème morphologique (*top-down*) et l'on imagine à l'aide d'un programme fictif l'optimisation du processus morphologique[16].

Bref, l'idée principale émise ici est que la stratégie consistant à avancer dans le texte – en recourant le moins possible au dictionnaire – est la plus efficace, à condition toutefois de savoir repérer les éléments les plus contraignants : les *tokens*[17].

L'expression de ces contraintes sous forme d'automates s'imbriquant les uns dans les autres découle de l'hypothèse de l'existence d'un « processeur » abstrait et optimal de décodage[18].

Or celui-ci, s'il existe, ne peut être atteint que par approximations successives et sa mise au point progressive ne peut se faire qu'en procédant par rétroaction continue** (*feedback*)*. C'est là tout le sens de notre travail.

## 5. Méthode de construction de grammaires par agrégation de fragments

Quel que soit le niveau auquel on se situe : cognition, didactique, description linguistique, algorithmique – entre lesquels, comme nous venons de le dire, nous établissons des correspondances – nous procédons par description de fragments (de grammaires ou programmes) ;

16. Les personnes familières de la programmation dite récursive ou descendante (genre Lisp ou Prolog) seront moins étonnées par ce type de démarche. La méthode récursive constitue un paradigme logique puissant. Ne pas confondre par ailleurs la « Récursivité » comme discipline, qui désigne de manière générale la théorie de la calculabilité, avec le raisonnement récursif.

17. Ces éléments, qui sont les plus contraignants, coïncident justement avec les éléments figés (les atomes morphologiques). Cette coïncidence a une explication simple : la cardinalité de leur classe de congruence syntaxique ** est très limitée (souvent égale à l'unité) contrairement à celles des autres lexèmes, lesquels peuvent commuter avec d'autres éléments appartenant à la même catégorie (ou classe de congruence syntaxique) sans remettre en cause la grammaticalité de la phrase, ni son type (pour ce qui est du rapport entre la congruence syntaxique et la congruence induite par les schèmes, voir dans « Approche algorithmique de la grammaire arabe »** le chapitre premier intitulé « Système morphologique et monoïde syntaxique ». Une caractérisation algébrique des *tokens* y est donnée : ce sont les invariants « lexicaux » de la projection du langage sur son squelette (autrement dit les invariants de la projection L → L /RAC). Le mot « *token* » a été choisi en référence à ce que les concepteurs de langages informatiques désignent par ce terme : des associations de symboles qu'il faut considérer comme figés (exemple : BEGIN, GOTO, LOOP, ….etc.), lesquels induisent naturellement des *attentes* particulières.
Signalons également que le terme « *token* » est traduit par certains informaticiens francophones par « lexème » (on pourrait d'ailleurs les considérer, en quelque sorte comme les « lexèmes » du langage quotient L /RAC; nous n'avons pas retenu cette traduction pour éviter les risques de confusion. Il est évident par ailleurs que nous n'utilisons pas ce terme dans le sens de Pierce par exemple. Notre souci premier était simplement d'attirer l'attention sur l'analogie avec les langages formels. L'objet L/RAC est un langage semi-formel obtenu, par projection, à partir d'un langage naturel. Il présente la particularité de posséder un lexique très limité. C'est ce fait qui nous semble le plus caractéristique du système sémitique (la forte grammaticalité/algorithmicité) dont la langue arabe est l'un des représentants les plus caractéristiques.

18. Que l'apprenant « idéalisé » mettrait en marche dans sa tête avec l'aide de l'observateur : le professeur. À ce sujet, voir les remarques dans « À la recherche du *Ḫabar* ».

plus précisément par tentatives de description que l'on affine progressivement en les mettant en œuvre informatiquement, en les confrontant ensuite systématiquement aux corpus et en bouclant rétroactivement jusqu'à obtenir un niveau jugé satisfaisant.

Cette méthode nous astreint à « spécifier » mathématiquement nos programmes et nos grammaires sous peine de perdre entièrement le contrôle sur notre travail. Une légère transformation peut rendre l'ensemble incohérent. Les transformations de grammaires seront nécessaires pour rendre compte des points de vue divers que l'on porte sur la langue. Notre souci d'universalité, et surtout de cohérence méthodologique, questions sur lesquelles nous insistons tout au long de cet article, font que ces « programmes/grammaires » doivent être structurés en sorte que l'on puisse effectuer des calculs en vue de les manipuler, voire les transformer en d'autres programmes/grammaires qui leurs seront équivalents ou proches (d'où l'intérêt de pouvoir définir des topologies voire des « métriques », *i.e.* des distances dans des espaces abstraits). La variation des points de vue (c'est-à-dire des différents programmes/grammaires) est d'autant plus important pour nous que nous nous intéressons justement aux invariants. J'ai déjà eu l'occasion de signaler dans la bibliographie commentée** l'un des intérêts que représente – de mon point de vue – ce travail dans le domaine des études arabes automatisées : celui de pouvoir atteindre le programme optimalisé par « variation de grammaires ».

Cet article est à la fois un plaidoyer pour l'abstrait et pour « la machine ». La langue arabe – et non seulement la manière d'interagir avec elle, de l'appréhender ou de la traiter automatiquement comme on dit communément – est, elle-même, considérée comme une « machine abstraite ».

Cette étude s'inscrit, au niveau le plus théorique qui soit, dans celui des « Études arabes ». Par ailleurs elle relève aussi de l'univers de la machine, c'est-à-dire du TAL (Traitement automatique du langage). Il est souhaitable de concilier ces deux aspects car la possibilité d'une unification théorique de ces points de vue, apparemment étrangers l'un à l'autre, est une réalité.

Cette « chance », c'est la théorie algébrique de Schutzenberger et de son école qui va nous l'offrir. L'importance que revêt pour nous la « spécification » algébrique de notre travail aussi bien au niveau de l'arabe qu'au niveau informatique s'inscrit dans le souci constant que nous avons eu de « regrouper [dès le début] l'ensemble des connaissances sur lesquelles [il] se fonde, même dans ses aspects les plus pratiques (méthodes de programmation), en un corps de doctrine unique ». Cet esprit, qui caractérise notre entreprise, nous ancre dans une certaine tradition ; il procède en effet d'une certaine « philosophie » que J. Pin ** a été le premier, sauf erreur, à discerner et qui, dans le monde de l'informatique théorique, est souvent désignée par l'expression « *french school* » ; laquelle ne manque pas de prestige, grâce notamment à la qualité de ceux qui y participent et surtout l'incroyable richesse de la personnalité et du génie de son fondateur : M.P. Schutzenberger*.

Les avantages considérables que présente une théorie algébrique des automates se transfèrent en quelque sorte à l'arabe. L'une des illustrations de cette thèse est constitué par le calcul du monoïde de transition du groupe nominal arabe déterminé, qui nous fournit du même coup l'automate minimal d'acception** (l'application à la morphologie n'est qu'esquissée**).

AnIsl 44 (2010), p. 5-38 André Jaccarini
De l'intérêt de représenter la grammaire de l'arabe sous la forme d'une structure de machines finies.

## 6. L'énumération universelle[19]

Rappelons que l'informatique n'est constituée que de programmes, indépendamment des supports physiques sur lesquels ils s'exécutent (circuits électroniques inscrits dans du silicone, neurones, cellules vivantes ! etc.). « L'objet "ordinateur" sans son système d'exploitation – qui est un ensemble de programmes – n'est qu'une page blanche[20]. »

L'informatique a été en effet « inventée » avant que Von Neuman n'eût conçu le premier ordinateur dans le cadre du projet Manhattan, dans un contexte d'ailleurs bien différent de celui qui vient d'être évoqué. Cette « invention », à moins qu'il ne s'agisse d'une « découverte », s'est produite dans le cadre d'une recherche philosophico-logique touchant aux « fondements » même des mathématiques.

Dans le cadre de son « programme » de fondation des mathématiques, Hilbert a posé le problème suivant : « Existe-t-il une procédure générale effective permettant de déterminer si une formule du calcul des prédicats est démontrable ou non ? » En répondant par la négative, en 1936, Alan Mathisson Turing ne se doutait sans doute pas qu'il inaugurait du même coup une nouvelle ère dans l'histoire de l'humanité[21]. En effet, dans son célèbre article : « On Computable Numbers, with an Application to the *Entscheidung problem* [22]», Turing est amené à proposer une définition rigoureuse, mathématique, de ce qu'est un algorithme. Car ce que l'on entend par « procédure finie », ou même par « calcul », est intuitif. Aussi étrange que cela puisse paraître de prime abord la notion même de calcul, contrairement à celle de nombre, n'est pas mathématique, elle est métamathématique. Turing en propose donc un modèle qui est une

19. Ce paragraphe historique est important car il constitue une introduction logique à la justification, au paragraphe suivant, du traitement (au sens le plus large que recouvre ce terme) de la langue arabe par « automates abstraits ».

20. Voir Jean-Louis Krivine : « Mathématique des programmes et programme des mathématiques »* ; ainsi que «Ensembles et Preuves»* et d'autres articles de cet auteur et chercher aussi « correspondance de Curry-Howard »*. La théorie « philosophique» de Krivine est lumineuse : il répond notamment à cette question que s'était posée Einstein (d'autres, naturellement, se la sont posée avant lui ; toutefois signalons que celle ci l'obsédait tout particulièrement) : « Comment se fait-il que les mathématiques, qui sont un pur produit de mon esprit, s'appliquent si excellemment au réel ? »

21. « Lorsque Alan Turing publie en 1936 son fameux article, il n'a certes pas conscience d'énoncer les prolégomènes d'une nouvelle science de l'esprit », Jean-Pierre Dupuy : « Aux origines des sciences cognitives », La Découverte, 1999. Quand on parle de science de l'« esprit » c'est au sens de « *mind* » bien entendu, tout comme on parle de « philosophie de l'esprit » (laquelle est une lointaine descendante de la philosophie analytique » anglo-saxonne). À cette occasion nous recommandons vivement la lecture de cette excellente étude où se trouve décrite, avec une grande intelligence, la richesse des confrontations interdisciplinaires conduites au sein des fameuses « conférences Macy »*, au cours desquelles, discutèrent, de 1946 à 1953, certains des plus grands esprits du XX$^{e}$ siècle, dont certains furent des génies dans leurs disciplines (mathématiques, biologie, neurophysiologie, ingénierie, sociologie, linguistique, économie,…. ). Pour juger de la qualité des participants, il suffit de citer quelques noms: Von Neuman*, Wienner*, Warren McCulloh*, H. Simon*, Von Hayek*, Rosenblueth*, etc. ; le linguiste Jacobson ne fit qu'y passer. Ce groupe se donna comme nom de code « Cybernétique » (terme inventé par André Marie Ampère pour désigner la science du *gouvernement*).

22. Entscheidung = décision .

machine « abstraite » (la « machine de Turing »)[23]. En fait à cette époque on dispose déjà d'autres « caractérisation » de ce qu'est une « fonction calculable». Gödel avait proposé la récursivité générale et Alonzo Church avait déjà inventé le lambda-calcul[24]. D'autres formalismes furent élaborés plus tard pour essayer de « capter » la notion de fonction « calculable », dont l'un des plus impressionnants est le modèle que le neurophysiologiste McCulloh, en association avec le logicien Pitts, proposa en 1943 (donc avant l'invention de l'ordinateur) : celui des neurones formels. Toutes ces « tentatives » (qui sont aussi des *modèles*) sont très différentes les unes des autres, elles procèdent en tout cas d'intuitions variées dont on a du mal, de prime abord, à saisir le « dénominateur commun ». Que ces formalisations aient toutes été prouvées comme étant coextensives est un fait remarquable. C'est cette équivalence qui nous fait accepter sans trop de difficulté la thèse de Church-Turing (1936) :

– Toute fonction effectivement calculable peut l'être à l'aide d'une machine de Turing ;

ou bien :

– Tout ce qui est calculable, au sens intuitif, peut être représenté par des fonctions récursives ;

ou bien encore :

– par des réseaux de neurones ;

– par des automates à deux piles de mémoire ;

– par des automates cellulaires[25].

Dans le même article, Turing énonce un résultat remarquable : *le théorème de l'énumération universelle* : « Existe une machine singulière, dite universelle, capable de simuler le fonctionnement de toute autre machine de Turing. »

– C'est ce qu'on appelle le *principe de fonctionnement des ordinateurs*[26]. Il ne reste donc plus qu'à créer ces derniers. Ce fut chose faite en 1944 (Von Neuman) ; ces derniers qu'on appelle aussi quelquefois – dans la mesure où ils sont séquentiels – machine à la Von Neuman[27], par opposition par exemple au Lisp Machine aux machines parallèles ; neuronales, etc. ; aux automates cellulaires.

Dans le même article, Alan Turing démontre un autre résultat fondamental qui, après celui de Gödel sur l'incomplétude (1931), va très sérieusement ébranler le programme de Hilbert : « Aucune machine de Turing ne peut prévoir si une machine de Turing quelconque va s'arrêter ou non. » Le problème de l'arrêt est *indécidable*. Il entraîne entre autres conséquences l'indécidabilité du langage des prédicats du premier ordre. Ce qui est pour le moins gênant pour le programme de fondation des mathématiques.

23. Sur la notion de modèle : voir le principe *Verum et factum convertuntur* dans Jean Pierre Dupuy « Aux origines des sciences cognitives », 1er chapitre : La notion de modèle,.

24. Le lambda-calcul est le paradigme du langage informatique Lisp (McCartney, MIT, 1960).

25. Il existe plusieurs autres machines abstraites équivalentes aux machines de Turing.

26. On peut consulter l'excellent article de Bernard Jaulin sur la récursivité dans *l'Encyclopedia Universalis*,

27. Von Neuman est le premier à avoir distingué le matériel du logiciel (*hardware* / *software*). On dit aussi que, si Alan Turing a été le père de l'ordinateur, Von Neuman en a été la sage-femme.

Le langage des prédicats n'est que semi-récursif (ou récursivement énumérable). Un langage est dit récursif s'il est semi-récursif et que son complémentaire l'est aussi. Par exemple il existe une machine de Turing (ou tout autre mécanisme équivalent) capable d'engendrer tous les théorèmes du calcul des prédicats, mais il n'en existe pas une capable d'engendrer tous les non-théorèmes.

Un langage est récursif si et seulement si il existe :

1. Une machine de Turing capable d'engendrer tous les énoncés grammaticaux ;

2. Une machine de Turing capable d'engendrer son complémentaire c'est-à-dire tous les énoncés non grammaticaux.

Un tel langage présente alors la particularité d'être *décidable* : soit x un énoncé quelconque, alors il est toujours possible de décider si cet énoncé est grammatical ou non puisqu'il sera forcément produit par l'une des deux machines susmentionnées.

Dans la hiérarchie de Chomsky-Schutzenberger qu'on évoquera au paragraphe suivant, en liaison avec les automates correspondants (avec 0, 1 ou 2 piles de mémoires), toutes les grammaires sont récursives à l'exception de celles de plus haut niveau (équivalentes aux grammaires transformationnelles de Chomsky) qui ne sont que récursivement énumérable (ou semi-récursives) c'est-à-dire qu'elles sont capables d'engendrer des langages indécidables.

Ce résultat a pour corollaire immédiat que le calcul des prédicats du premier ordre ne peut être rendu que par une grammaire de Chomsky de niveau le plus élevé (laquelle correspond à la capacité de calcul la moins contrainte d'une machine de Turing).

Ce fait a eu aussi une autre conséquence : il a entraîné certains à penser que les grammaires transformationnelles de Chomsky, sans restriction, constituaient peut-être un formalisme trop puissant (du point de vue de leur capacité de génération mathématique) étant donné qu'il était difficile d'imaginer dans une langue donnée la question de la «grammaticalité » d'une phrase comme étant indécidable[28] !

## 7. Pourquoi avoir privilégié la théorie des automates pour la représentation de l'arabe ?

Nous avons vu au paragraphe précédent que toutes les tentatives pour définir la notion métamathématique de « calcul », plus précisément de « fonctions calculables par une machine », avaient toutes abouties à des modèles équivalents (tout au moins jusqu'à présent[29] !). Bien plus, lorsqu'on a construit – dans un domaine, à l'origine tout à fait indépendant de la logique mathématique et du programme de Hilbert, celui de la *neurophysiologie* – une représentation de l'activité électrique cérébrale, on a réussi à *prouver* son équivalence avec le modèle des machines

28. D'aucuns pensent néanmoins que cela est discutable.

29. Rappelons que si l'on parle de « *thèse* de Church » ou de « Church-Turing » et non de *théorème* c'est que cette dernière n'est pas *prouvable*, pour la simple raison que la notion même de calcul (effectuable sur machine) est métamathématique. Au paragraphe précédent, nous n'avons cité que quelques modèles, la liste est loin d'être exhaustive.

AnIsl 44 (2010), p. 5-38 André Jaccarini
De l'intérêt de représenter la grammaire de l'arabe sous la forme d'une structure de machines finies.

de Turing ! (cette fois-ci il s'agit d'un *théorème*[30] ; McCulloh et Pitts, 1943) ; ce qui naturellement ne pouvait que susciter – chez certains – des fantasmes faustiens (le cerveau artificiel !)[31].

Pour en venir à l'essentiel, en ce qui concerne la représentation du langage, et de la langue arabe en particulier (comme procédure et comme structure), je vois surtout dans la théorie des automates (classiques[32]) les avantages suivants :

1. L'aspect de linéarité est prépondérant aussi bien dans la production que dans la reconnaissance du langage. Dans la phrase (voire le « mot » dans le cas de l'arabe[33]), les symboles se succèdent (linéarité) avec des probabilités variables, tout comme une succession d' « événements », se produisant les uns à la suite des autres avec des probabilités dépendantes des symboles qui se sont déjà produits ; certains d'entre eux pouvant boucler un nombre indéfini de fois. C'est là l'aspect « processus » : le *temps* y joue un rôle essentiel (tout comme dans un programme qui s'exécute), à la différence de la structure qui est donnée, *indépendamment du temps*, (en linguistique, il s'agit en général d' « indicateurs syntagmatiques », des arbres formellement équivalents à des systèmes de « parenthèses »). Ces successions d'événements peuvent être décrits par des graphes** lesquels peuvent être facilement *spécifiés** mathématiquement par des langages rationnels (dit réguliers**,*) ;

2. Le *théorème de Kleene* (1956) est essentiel : il affirme que la classe des langages « engendrés » (ou reconnus) par la catégorie la plus « simple » d'automates, c'est-à-dire ceux d'entre eux qui sont sans piles de mémoire, dits « automates à état finis », coïncide avec celle des langages rationnels (réguliers). Il est possible d'établir une correspondance entre des expressions régulières (qui sont des formules appartenant à un « langage » que l'on peut très simplement définir récursivement par des propriétés de clôture) et des automates finis (qui sont des programmes). Des calculs peuvent être effectués avec des expressions régulières dont les règles sont déduites de la définition même du langage : des identités, des résolutions d'équations, etc., auxquelles correspondent autant de *transformation* d'automates. C'est là un avantage qui est loin d'être négligeable. À toute manipulation et transformation d'automates (dits « à état fini ») on peut faire correspondre des opérations algébriques, des résolutions de systèmes d'équations. Cette *spécification* mathématique des « opérations », que l'on peut accomplir sur des programmes, est naturellement *essentielle*. Nous pouvons citer une multitude de telles opérations : transformer des automates (ou sections d'automates) en automates déterministes, inclure dans un réseau de

30. En ce sens que tout « réseau de neurones formels » peut être « simulé » (au sens de Turing) par une machine de Turing universelle (*i.e.* un ordinateur théorique) et vice-versa.

31. L'ordinateur ne sera inventé que l'année suivante. Dans les années 50 on désignait encore, souvent, cette machine par le terme « *cerveau* électronique » ; cependant, que l'on ne s'inquiète guère : un simple rappel d'échelle de grandeur nous ramènera très vite à la réalité : notre système nerveux comporte 100 milliards de neurones dont 10 milliards environ dans notre cerveau. La « complexité » de cet organe en état de fonctionnement n'est donc pas $10^{10}$, mais 2 à la puissance $10^{10}$ (deux à la puissance dix milliards autant dire *l'infini*. D'aucuns (comme D$^{r}$ Schutz) parleraient de « complexité ultra-cosmique ».

32. Par opposition aux automates cellulaires, sans compter que l'automate le plus célèbre est la « machine de Turing » elle même.

33. En arabe, dans la mesure où on peut avoir des agglutinants, nous entendons par « mot », le mot graphique, soit toute suite de lettres séparées par deux blancs. Exemple : *fakatabahu*.

tels automates d'autres automates, les sectionner, les raccorder, les minimiser, en synthétiser de nouveaux à partir de déjà existants et de sections de textes,[34]…

La spécification mathématique est essentielle car elle va rendre contrôlables toutes ces opérations et nous met à l'abri du danger d'erreurs qui est considérable dans ce domaine (sectionner des programmes, les transformer pour les recoller ensuite en vue de les optimiser n'est pas un exercice de toute facilité). Ces « langages réguliers » constituent une structure mathématique que l'on peut *enrichir* tout comme en algèbre on procède à des « extensions de structures » en vue de rendre possible des résolutions d'équations qui ne le sont pas dans les structures à partir desquelles elles sont construites ;

3. On trouvera dans « Modélisation Linguistique et Théorie des Automates »** (chap. 3) la description précise de la construction d'un corps non commutatif (qui peut être étendu en une « algèbre » non commutative ») où il devient possible non seulement d'effectuer des calculs ayant pour objets des automates finis mais où des transducteurs minimisés et optimisés sont également synthétisés, à partir de la résolution de systèmes d'équations. Une méthode de construction d'un programme Lisp récursif de reconnaissance de formes arabes est ainsi obtenu par le seul calcul (chap. 4)**. Or ces calculs peuvent être eux-mêmes automatisés (au sens où l'on peut les effectuer, à leur tour, par transduction) produisant des programmes optimisés avec en prime *la preuve* (mathématique) de leur validité (ils ne contiennent pas de *bugs*, du moins à ce niveau !) et l'évaluation de leur complexité. Cette manière de programmer m'a semblée assez originale pour faire l'objet des deux chapitres susmentionnés. Il est également possible, par la donnée d'une simple correspondance de type « homomorphique » définie par un ensemble de 4 règles, de définir un analyseur « linéaire » pour automate fini non déterministe. Cet analyseur peut ensuite être enrichi modulairement (*i.e.* l'on enrichit le programme originel d' « actions », en sorte de conserver sa structure de base) pour se transformer en analyseur pour des automates contenant des transitions conditionnelles et des « actions » : des tests supplémentaires sous forme de « prédicats logiques[35] » peuvent être défini, ainsi que des « effets de bord[36] ». Cet analyseur enrichi est utilisé pour effectuer aussi bien l'analyse syntaxique que morphologique ;

4. La possibilité de concevoir un traitement *global* de l'arabe par une grammaire unifiée (dans le cadre des automates et des transducteurs) me semble caractéristique de l'arabe (voire du sémitique). Le recours au lexique est remplacé par des appels de sous automates morphologiques dont le vocabulaire de base est tout simplement l'alphabet, auquel on adjoint toutefois les mots figés : les *tokens* (nous en avons déjà parlé). La possibilité théorique de définir une grammaire formelle ayant pour vocabulaire de base des lettres et une suite d'environ 300 mots figés est une spécificité forte du système arabe[37]. Comme la syntaxe est généralement définie par des grammaires formelles de type *Context-free* (voir hiérarchie de Chomsky-Schutzenberger)

34. Voir à ce sujet le dernier chapitre de « Modélisation Linguistique et Théorie des Automates »** : « Vers un atelier de grammaire » (résumé dans Bibliographie **), ainsi que Taln 03, Medar 09.
35. Toute fonction (programme) ne pouvant donner pour résultat que les valeurs 1 ou 0 , « vrai » ou « faux ».
36. Des programmes peuvent être associés aux transitions de l'automate. Cet aspect est important comme nous l'avons vu au §2.
37. Généralisable au sémitique ?

renforcée[38] (appelée aussi grammaire à attributs sémantiques*, grammaires de Knuth ou encore schéma de traduction guidée par la syntaxe), ces dernières peuvent être traduites naturellement en transducteurs (voir par exemple la correspondance entre les ATN de Woods et les grammaires de Knuth* dans « Modélisation Linguistique et Théorie des Automates »**, chap. 2) ;

5. Les « pertinences » des grammaires formelles se testent, en règle générale, par les résultats produits par le programme d'analyse (dite opération de *parsing*), qui rappelons-le est une opération inverse (voir plus haut le point 1 du § 3). Or le modèle le plus naturel pour représenter un flux d'informations (que ce soit en lecture ou écriture d'ailleurs) est bien celui des automates augmentés (ils sont nécessairement augmentés pour tenir compte des opérations de « *backtrack* », de va et vient, généralement nécessaire pour le décodage d'où la nécessité de registres de mémoire sur lesquels on peut effectuer certaines opérations). Ils simulent l'opération d'avancement dans le texte associée à la construction de la représentation de la « structure de la phrase » (généralement un arbre « décoré »**,*). L'aspect de linéarité reste dominant ;

6. Un problème fort intéressant et spécifique à l'arabe se pose alors. En l'absence de lexique (ce qui est d'ailleurs, rappelons-le, le cas limite de l'apprenant au lexique réduit ; voir plus haut (§4, §3), il existe un flux directeur de plus haut niveau, celui des mots ; lequel s'appuie sur des sous-flux : ceux des lettres.

Or, comment s'opère justement l'interaction entre syntaxe et morphologie (qui est caractéristique du décodage de l'arabe) ? Comment éviter les blocages, les cercles vicieux (« *deadlock* » entre processeurs, abstraits, morphologiques et syntaxiques en attente l'un de l'autre) ? La représentation par automates s'appelant réciproquement est avantageuse en ce sens qu'elle peut se traduire en termes d'évitement de cercles vicieux dans les grammaires de Knuth ; or, dans ce domaine les algorithmes existent[39].

Une remarque est toutefois nécessaire : nous cherchons le plus souvent des heuristiques*. Les solutions « calculatoires » qui existent déjà pour des problèmes homomorphes (ou « quasi homomorphes[40] ») ne sont censées que nous donner des idées. Nous ne cherchons pas toujours des solutions précises à certains problèmes qui peuvent quelquefois provoquer des explosions combinatoires. La question de la complexité est fondamentale lorsqu'on aborde les problèmes que pose la modélisation du processus d'appréhension du texte ;

7. Les automates peuvent être de simples spécifications ; ils reflètent des contraintes qui existent entre des mots au niveau de la syntaxe ou entre des graphèmes, au niveau de la « morphographie ». Il est vrai que nous avons mis au point des analyseurs pour transducteurs non déterministes arabes – nous l'avons déjà dit, nous les avons même *calculés,* ce qui signifie la

38. Afin de tenir compte de certains aspects « contextuels » de la syntaxe mais aussi afin d'établir une correspondance plus aisée avec la représentation sémantique. Pour ce qui est de la puissance – mathématique – d'engendrement des grammaires Hpsg ou autres voir Torris et Millers (« Les nouveaux formalismes pour l'analyse syntaxique », Hermès, 1990).

39. Les « grammaires à attributs »* ont été introduites par Knuth en 1968 : première définition des attributs synthétisés et hérités avec la description d'un algorithme d'évitement de cercles vicieux.

40. Nous reviendrons, dans un autre contexte, sur ce concept un peu flou. Pour l'instant nous demandons au lecteur de l'accepter dans son sens intuitif.

possibilité de leur génération automatique (*i.e.* par automates ; voir plus haut le point 3) – mais ces réalisations ne préjugent en rien des algorithmes qui seront finalement retenu pour le décodage. La contrainte de *linéarité* peut en effet représenter, dans certains cas, un handicap pour l'optimisation du décodage (nous le savons d'autant mieux que nous avons déjà programmé en APL, où l'on procédait sur des textes entiers et non mot à mot (voir plus haut) et que, par ailleurs , sur le plan cognitif, en didactique, nous explorons l'hypothèse d'« avancement dans le texte avec recours minimal au lexique, et émission d'hypothèses quitte à revenir ensuite à un décodage morphologique plus rigoureux dans un deuxième temps et dans un ordre non forcément linéaire[41] »). Ces outils (les analyseurs) peuvent toutefois nous servir à rechercher d'autres *outils* plus efficaces (voir à ce propos dans les derniers chapitres de « Modélisation linguistique et théorie des automates »**, ainsi que dans « Approche algorithmique de la grammaire de l'arabe »**, certaines idées d'expériences). En effet, nous avons la possibilité d'effectuer des expériences sur des programmes, *i.e.* sur des calculs ; nous reviendrons plus bas sur cette *qualité épistémologique extrêmement importante* qu'offre l'ordinateur : il fait du Calcul (dans le sens le plus large que revêt ce terme ) un objet d'*expérience* tout comme la matière est un objet d'expérience pour le chimiste ou le physicien ; il transforme donc en quelque sorte un « objet » intellectuel en objet d'expérience (modulo un minimum d'interactivité bien sûr, pour rester réaliste)[42] ;

41. Voir Audebert, Jaccarini, « À la recherche du *Ḥabar* » ** et Jaccarini, « Vers une théorie du moniteur syntaxique» **.

42. Il faut être conscient qu'il existe des seuils, notamment dans le domaine de la complexité : à partir d'un certain niveau, il est par exemple infiniment plus simple de donner le plan de câblage de l'automate plutôt que de décrire ses différentes configurations : il s'y produit comme une sorte de renversement. Dans des cas de grande complexité (en météorologie, en économie, il peut se produire une inversion dans le rapport du réel au modèle : il peut être, par exemple, beaucoup plus facile d'observer les lois du marché que de les prévoir mathématiquement (*idem* pour la météo) ; dans ce cas, ce sont l'économie ou la météo qui viennent au secours des mathématiques plutôt que l'inverse (car il faudrait alors résoudre en un temps, *plus rapide que celui du déroulement du phénomène*, des dizaines de milliers d'équations différentielles)). Cela fait penser à la superbe phrase que Borgès attribue à un auteur de son invention, concernant les cartes d'échelle 1/ 1. Or il se trouve que ce genre de situation peut à nouveau se reproduire non pas seulement en économie, en météorologie, dans les sciences humaines ou celles du vivant mais dans le domaine du Calcul lui même dont l'extrême foisonnement et la profusion peut égaler celui du Vivant (voir à ce sujet l'article « la riche zoologie des automates cellulaires » *in* : *Pour la Science*, décembre 2003 »). Ces problèmes de complexité non maîtrisable, nous allons aussi les rencontrer très tôt en linguistique. Certains automates d'apparence anodine peuvent être en fait des «monstres cachés». On acceptera ainsi plus facilement cette idée, qui n'est peut-être pas très intuitive au départ, que le Calcul lui-même peut être objet d'expérience. Nous y reviendrons. Elle conforte, en tout cas, notre méthode expérimentale de construction d'automate par rétroaction continue, ainsi que certaines critiques visant le travail de trop grand raffinement des grammaires formelles qui obéissent presque toujours à la loi 20-80 que nous avons évoquée dès l'introduction de notre ouvrage « Approche algorithmique de la grammaire arabe »**. Il faut savoir à partir d'un certain moment lâcher prise et se contenter d'heuristique* (nous en avons déjà parlé). La méthode de comparaison des grammaires par comparaison systématique des configurations que prend chacun des automates associés (voir ci-dessous méthodes de comparaison initiée dès 1992 (voir Audebert, Jaccarini, « «Méthode de variation de grammaire et algorithme morphologique» **) nous conduit naturellement à l'idée de la nécessité de définir des « métriques » associées à des « normes », lesquelles définissent la manière d'intégrer les critères retenus en en privilégiant certains, en fonction du but

8. Un *éditeur structurel* de grammaires arabes, a été défini en Lisp, plus précisément un éditeur de transducteurs arabes (équivalents à une sous-classe de grammaires de Knuth : les arbres « sémantiques » associés étant des arbres de lambda-expressions[43] ; la limite étant que seules sont pris en compte, ce que dans la terminologie des grammaires de Knuth on appelle, « les attributs synthétisés » par opposition aux « attributs hérités[44] »). Un éditeur structurel est un éditeur qui a connaissance de la nature « syntaxique » de l'objet qu'il traite[45]. Or cet éditeur est lui-même un transducteur (on peut donc le définir dans lui-même !), ce qui offre la possibilité d'améliorer indéfiniment ses performances (par rétroaction) en fonction des besoins nouveaux qui ne manqueront pas de se présenter au fur et à mesure de l'avancement de la recherche linguistique[46] .

Il s'agit là d'une illustration frappante d'une qualité singulière de la théorie des automates: la possibilité de faire du « calcul » sur du « calcul » tout en restant rigoureusement dans le même cadre théorique ; ce qui offre non seulement un bel exemple de méthodologie récursive, mais permet aussi, en un certain sens, à mieux cerner la dualité « Structure, Procédure ». La « richesse et le pouvoir d'unification » de la théorie des automates n'est pas une formule creuse.

## 8. L'étude de David Cohen et la théorie des automates

Nous isolons ce 9e argument en faveur des automates pour la modélisation de la langue arabe, en un paragraphe autonome, car il me semble que sa force est proportionnelle à sa grande simplicité.

recherché. Très formellement, c'est-à-dire à la fois *abstraitement* et du point de vue de l'économie intellectuelle, nous avons ainsi la possibilité de définir une application informatique dans le domaine de la linguistique arabe, comme étant la simple donnée d'une *norme* (dans un espace abstrait), laquelle hiérarchise les critères que l'on a préalablement fixés par expériences « rétroactives » (voir, plus bas, le paragraphe concernant la transparence de l'analyseur et ses conséquences). La réalisation d'une application informatique de l'arabe revient ainsi à choisir le point de vue idoine, pour peu que l'on ait réussi à faire apparaître le programme recherché comme résultant d'une grammaire donnée, laquelle représente un point de vue (voir « Modélisation linguistique et théorie des automates. Méthode de variation de grammaires en vue de l'obtention de l'algorithme optimal») à laquelle on applique l'analyseur adéquat, lequel, ne l'oublions pas, est déduit d'un simple calcul effectué dans la même structure que celle où l'on a défini notre grammaire.

43. *I.e.* des « fragments de programmes » Lisp, si je puis dire : des « actions ». Nous cherchons d'ailleurs à « minimaliser » ces actions en vue justement de valider le postulat de régularité (voir § 2) et obéir au principe d'économie et de *minimalité*.

44. Il existe d'autres contraintes afin de préserver la structure mathématique évoquée précédemment (point 3), en sorte que les calculs possibles demeurent valides (ce qui nous permet des synthèses de transducteurs à partir de fragments). Des exemples de tels calculs sont fournis dans les ouvrages susmentionnés.

45. Par exemple un éditeur Lisp a connaissance de la grammaire qui régit les systèmes de « parenthésage », si bien qu'il peut par exemple signaler un défaut dans le système de parenthèses ou déceler d'autres erreurs de syntaxe (par un voyant ou une sonnerie ; proposer même des corrections).

46. Voir dans Bibliographie** (réf. 2), le chapitre intitulé « Un éditeur structurel de grammaire pour le traitement de l'arabe».

L'ensemble de l'étude pionnière de David Cohen – soit une trentaine de pages au moins – peut être spécifiée tout entière par un simple automate fini non déterministe ne comportant que six états ! Cette extrême concision est symptomatique ; elle représente un intérêt épistémologique considérable, du moins si l'on accepte le principe selon lequel « toute économie d'écriture représente un approfondissement conceptuel », généralement admis chez les logiciens. *Attention* (ce qui va sans dire va encore mieux en le disant) : nous ne disons pas que nous avons effectuée une analyse plus « profonde conceptuellement » que celle de D. Cohen : nous « glorifions » ici l'automate à état fini appliqué à l'arabe (ou tout au moins à sa structure graphique de surface), mais sans « diminuer » pour autant D. Cohen ; au contraire nous nous fondons sur l'ensemble de son étude ; nous n'élaguons rien ni ne rectifions ; nous ne faisons que « représenter », donc nous modélisons grâce aux langages rationnels (qui codent les automates ; voir théorème de Kleene cité au point 2 du paragraphe précédent). Nous aboutissons ainsi à une extrême concision. Notre thèse est que cette concision est significative. Dans Audebert, Jaccarini, « Variations de grammaire et algorithme morphologique. Vers un extracteur de racine en arabe » ** ; nous fournissons dans une première partie une spécification ensembliste de cette étude[47], que nous traduisons ensuite en un automate minimal non déterministe.

Au niveau procédural, en raison de son non-déterminisme, cet automate ne peut apparaître comme un *programme* mais plutôt comme une « pré-procédure ». Il ne s'agit que de la simple description d'une structure de surface. Pour la rendre « effective »[48] il faut :

– soit transformer cet automate en automate déterministe [49] qui, en général, comporte un grand nombre de transitions, mais dont l'exécution est très rapide, ayant néanmoins l'inconvénient de cesser d'être pertinent sur le plan linguistique ou simplement conceptuel (ce qui

47. Nous fournissons notamment en exemple, par une traduction en une simple expression rationnelle (régulière) très courte, une spécification de l'ensemble des formes arabes ne contenant aucune consonne solide (les consonnes solides sont celles qui sont toujours radicales ; il en existe 16 soit 4/7 de l'alphabet) mais dont l'extraction de la racine ne suppose *aucun* calcul ! (cela est dû à la position relative, particulière, de certains graphèmes).

48. C'est-à-dire la transformer en algorithme.

49. La démonstration du théorème qui établit l' « équivalence faible »** des automates déterministes et non déterministes est traduisible en un algorithme ; celui-ci est raisonnable car, dans le programme que nous avons mis au point, nous évitons de calculer les états inaccessibles : les nouveaux états sont, en fait, des regroupements d'états de l'automate de départ non déterministe, ce qui fait que, dans les cas les plus défavorables, mais qui, dans notre champ d'étude, sont plutôt rares, pour des raisons que nous n'expliciterons pas ici, le nombre d'états possibles peut être égal à 2 à la puissance du nombre d'états de l'automate initial (ce qui fait que pour un automate d'une trentaine d'états nous pouvons obtenir un résultat de l'ordre du milliard !). Or, ce sera justement le cas pour le deuxième automate, moins fruste, que nous présentons dans cette étude : il comporte 31 états. Cependant ce dernier peut être rendu déterministe (ce qui suppose néanmoins quelque heures de calcul sur machine, mais qui ne sont à effectuer qu'une seule fois : nous ne sommes pas en temps réel, l'automate est ensuite stocké en mémoire. Il faut toutefois noter que dans ce cas il ne s'agit que d'un « accepteur » de mots arabe bien formés : d'un filtre, lequel n'est pas absolu non plus, c'est-à-dire que si toute les formes rejetées ne peuvent constituer des mots graphiques licites, celles qu'il accepte peuvent être quelquefois illicites ; le filtre le moins grossier est celui qui est obtenu par intersection de tous les filtres précédemment construits d'où par exemple l'intérêt du théorème suivant : La classe des langages rationnels (ceux qui sont acceptés par des automates finis, en vertu de Kleene) est stable, relativement à l'opération ensembliste d'intersection : nous

AnIsl 44 (2010), p. 5-38 André Jaccarini
De l'intérêt de représenter la grammaire de l'arabe sous la forme d'une structure de machines finies.

signifie que l'on ne peut plus attacher aux états ou aux transitions entre les états des catégories linguistiques ou simplement des catégories significatives) ;

– soit mettre au point un analyseur dont nous avons déjà parlé au point 3 du paragraphe précédent ; dans ce cas l'exécution est plus lente mais on peut rester dans un cadre linéaire (nous avons déjà fait allusion à la spécification *algébrique* de cet analyseur au paragraphe 8, qui est importante car elle nous permettra de démontrer facilement ses propriétés. J'y reviendrai d'ailleurs très rapidement en 9.4)[50].

## 9. La transparence des analyseurs, ses conséquences

### 9.1. *La comparaison des grammaires*

Une des particularités frappante des automates (classiques, cellulaires, neuronaux, machines de Turing,….) est de posséder à la fois des définitions très simples, de pouvoir être décrits par un très petit nombre de règles de base, de représenter des mécanismes d'une grande simplicité, voire frustes et de donner lieu néanmoins à des ensembles de configurations pouvant atteindre un niveau de complexité redoutable. Par exemple, au sujet de la définition de la machine de Turing, Jean-Pierre Dupuy a tout à fait raison de noter que : « Suivant le cas ou l'humeur, on en trouvera la description compliquée ; ou bien, au contraire, on s'*émerveillera* qu'un être aussi *primitif* puisse prétendre à représenter de façon synthétique tout ce que nous entendons intuitivement par procédure finie, effective ou décidable[51]. » Il en est de même des automates finis (ou même de ceux qui sont renforcés par l'adjonction d'une ou deux piles de mémoire, ces derniers possédant, rappelons-le, toute la puissance des machines de Turing : tout programme, tout calcul aussi complexe soit-il, par exemple tout l' « univers informationnel » d'Internet, peut être représenté par un simple automate à deux piles de mémoires).

sommes ainsi assuré que le filtre ainsi produit (par intersection de tous les filtres précédents) est lui même un automate fini. Il peut être atteint par approximations successives.

50. Nous avons exposé en détail ces résultats au premier congrès sur la langue arabe que nous avions organisé à l'Atala (Association pour le traitement automatique du langage), en janvier 1993, avec le concours d'Everhard Ditters de l'université de Nimègue et auquel assistait André Lentin (lequel, rappelons-le, avait lu de près l'étude de David Cohen, cf. *supra*). Ils ont été résumés dans l'article susmentionné (Audebert, Jaccarini, «Méthode de variation de grammaire et algorithme morphologique ») et largement développé par la suite dans le deuxième chapitre d'« approche algorithmique de la grammaire arabe ». L'un des principaux intérêts de ces développements est la démonstration de la possibilité de comparer les grammaires grâce à la transparence des analyseurs.

51. Dans la phrase citée (c'est moi qui ai souligné), j'aurais préféré au mot « compliquée » celui de « contingente ». En effet ce qui peut sembler « désagréable » de prime abord est bien une certaine impression de contingence (pourquoi cette machine particulière, à la fois fruste et étrange, plutôt qu'un autre mécanisme ?), la définition semble un peu arbitraire et l'objet est finalement « primitif » ; c'est la définition formelle de cette machine, contingente et très simple, qui apparaît alors « compliquée », voire un peu laborieuse ; je reviendrai plus bas sur cette impression gênante lorsque j'exposerai les avantages de la présentation algébrique de la théorie des automates.

Le fonctionnement de l'analyseur pour automate non déterministes (voire pour une classe de transducteurs ayant pour réseau sous-jacents des automates finis, enrichis d'actions obéissant à des conditions rigoureuses, permettant de préserver certaines propriétés mathématiques essentielles en sorte que la généralisation des calculs de transformation sur ces nouveaux objets reste possible) est rendu transparent. Cet analyseur est défini par un homomorphisme auquel nous avons fait allusion en 8.3. Il suppose le « fonctionnement » d'une « pile » à double entrée et à fond unique dont tous les états apparaissent, au fur et à mesure que s'exécute l'analyse. Aux successions des états de cette double pile sont associées systématiquement (la correspondance est un homomorphisme) là ou les différentes tentatives de parcours dans l'automate, qui sont effectuées en parallèle. Pour peu que l'automate ait été conçu en sorte que les états et/ou les transitions correspondent à des catégories ou « pseudo-catégories[52] », alors nous obtiendrons le déroulement de toutes les hypothèses explorées (et souvent abandonnées) en cours de la tentative d'analyse. Ces hypothèses sont menées de front. Si l'on aboutit en fin de processus à un seul état terminal de la double pile (réduite au symbole représentant son fond) auquel correspondrait un seul parcours dans l'automate depuis l'état initial jusqu'à un état de sortie lequel est donc associé à une description linguistique unique, on dira que la forme ou le segment étudié est non ambigu[53] ; en revanche, s'il en existe n (avec n> 1) on dira qu'il est accepté avec un coefficient d'ambiguïté égal à n. Si n = 0 le segment est rejeté (la pile n'a pu parvenir à un état terminal).

Chacune des étapes représente une configuration de la machine à un moment donné. Deux ou plusieurs machines (*i.e.* deux ou plusieurs grammaires) pourront donc être comparées relativement à une suite de symboles (s'il s'agit de morphologie, il s'agira de mots graphiques) en fonction de plusieurs critères : le nombre d'étapes, c'est-à-dire les différents états de la pile, le nombres d'hypothèses à chaque étape, les fonctions représentant le nombre de ces hypothèses : croissantes, décroissantes, alternées, périodiques, etc., et enfin la pertinence linguistique de chacune de ces hypothèse ce qui nous renvoie au point 10. 3 ci-dessous.

Ces critères doivent être identifiés au cours d'expérimentation et il faut établir leurs indépendances pour constituer une base.

Ce travail a été rendu possible grâce à l'analyseur Lisp mis au point dans les années 90.

52. C'est-à-dire des classes d'éléments auxquelles on ne peut conférer le statut de classes de congruences syntaxiques, qui n'obéissent donc pas au principe de commutation (nous sommes dans une logique distributionnelle). En revanche, ces éléments peuvent avoir une signification « relative », ils peuvent être définis par leur position dans l'automate sans que l'on ne soucie par exemple du fait qu'ils sont solidaires de l'ensemble du lexème (dans le cas de la morphologie). Dans ce dernier domaine, les préfixes, infixes et suffixes de schèmes appartiennent, par exemple, à cette classe de « pseudo-catégorie » ; elles entraînent une autre classe de cette nature qui revêt une grande importance : celui du segment radical, soit le plus petit segment contenant la racine (laquelle, rappelons-le, est souvent discontinue ; ces deux notions ne se confondant donc pas).

53. Ce qui ne signifie pas qu'en cours de parcours, il n'ait eu la possibilité d'explorer à certaines étapes plusieurs hypothèses à la fois, l'automate n'étant pas déterministe.

AnIsl 44 (2010), p. 5-38 André Jaccarini
De l'intérêt de représenter la grammaire de l'arabe sous la forme d'une structure de machines finies.

L'esquisse d'un atelier de grammaire entièrement défini en Lisp a été donnée dans le livre « Modélisation linguistique et théorie des automates »**. Ce travail nécessaire est aujourd'hui à reprendre entièrement étant donné le changement du contexte informatique.

Ces critères étant établis, il s'agit maintenant de définir une norme qui nous permette de les intégrer en sorte que l'on puisse affecter à chaque grammaire une valeur.

Les grammaires sont acceptées, évaluées, ou refusées en fonction du comportement de l'analyseur, bref il faudra bien alors leur « accorder une note », sauf que cette dernière sera relative. Elle sera relative au but recherché, ce qui est en accord avec notre point de vue, que nous avons déjà eu l'occasion d'exposer : la nécessité de faire varier les grammaires (ce qui nous a résolument poussé, dès le départ, à nous situer dans un cadre algébrique précis : « construire une application c'est définir une norme, choisir un point de vue » ; cette formule est importante car elle résume la philosophie du travail qui nous reste à accomplir et celui de notre souci de constituer pour l'avenir une banque d'automates, relationnelle, s'inscrivant dans un environnement informatique stable et solide nous permettant de tirer le meilleur parti des descriptions linguistiques normalisées. Sur ce point, les choix sont encore ouverts. J'y reviendrai, à une étape ultérieure.

À notre stade (étape 1 du programme : 2010-2012) nous cherchons à démontrer la « puissance » de la technologie des automates minimaux, dans le domaine de la recherche d'information (R.I), la fouille textuelle et la « caractérisation » des textes.

## 9.2. *La classification des formes arabes selon leur niveau de complexité*

La considération des configurations prises par l'automate à l'analyse de chacune des suites de symboles présente un grand intérêt. Considérons par exemple la grammaire G1 minimale et non déterministe à laquelle se trouve associée l'étude de D. Cohen précitée. On peut remarquer ainsi que certaines analyses se font en une seule passe. Nous voulons dire par là que jamais, à quelque étape de la double pile que ce soit, plus d'une hypothèse n'est explorée, ce qui donne à la suite des configurations de la machine (reconstitution de tous les parcours), après l'analyse de la forme, l'aspect d'un parfait triangle rectangle ; c'est le cas par exemple de **ḤiBR** (encre), de ʿa**MaL** (travail), **q**i**Ṭ**ʿā (pièce), **Q**i**Ṭ**āʿ (secteur), ʿa**Ğ**ū**Z** (vieux). Nous marquons en majuscules les graphèmes principaux (non diacritiques) et en gras ceux qui sont radicaux (le symbole ʿ solide (*i.e.* qu'il est toujours radical, quelque soit sa position), est utilisé pour désigner la lettre « *ʿayn* » qui est un graphème principal). Il n'en est pas de même, en revanche, pour Mu**ʿĞ**i**Z**a (même racine que la forme précédente), en raison de la non-solidité du premier graphème « *mim* », quoique la forme générale diffère assez peu des précédentes (voir figure dans Annexes d'Audebert, Jaccarini, « Méthode de variation de grammaire et algorithme morphologique »**). Dans le premier cas **ḤiBR**, la non-solidité du **B** radical est compensé par sa position puisqu'il se trouve inséré entre deux solides et ne peut constituer un infixe lesquels n'appartiennent qu'à un petit ensemble constitué de 4 graphèmes : *Alef, Ta, Waw* et *Yā*[54]. En comparant toutes ces

54. Nous envisageons le cas d'une lecture strictement linéaire (parseur classique d'automate, lequel ne sera pas forcément retenu dans tous les cas ; puisque la longueur de la forme, réduite ici à trois graphèmes principaux,

configurations, c'est-à-dire en essayant de dégager des critères et définir ensuite des normes pour calculer les « degrés » ou de manière plus nuancée « l'allure » que peuvent prendre ces déformations par rapport à l'hypoténuse rectiligne des 4 premiers cas et l'évolution du nombre des hypothèses, on peut aboutir à classifier les formes selon leur degré de complexité et disposer ainsi d'un instrument de « mesure » de la perspicacité de nos premières tentatives APL et les justifier ainsi mathématiquement.

Ces mêmes formes sont ensuite étudiées de manière similaire par une grammaire moins fruste, laquelle n'est pas forcément plus intéressante à mettre en œuvre dans tous les cas, surtout si l'on tient compte de critères complémentaires pour établir des comparaisons. L'idée étant d'établir le rapport qualité/prix en fonction des objectifs à atteindre. Par exemple, une hiérarchisation des ambiguïtés s'impose (voir §3). Selon que l'ambiguïté porte sur la racine, le lexème, sur certaines marques de nombre morphologiques externes, etc., elle n'aura pas la même incidence sur l'environnement.

Une boîte noire scellée eût rendu toutes ces expériences impossibles.

### 9.3. *Le parallélisme cognitivo-algorithmique*

La transparence de l'analyseur est indispensable si l'on désire établir des parallèles pertinents avec le « mécanisme naturel » d'appréhension du langage et l'opération de lecture et de décodage par des apprenants – de niveaux différents (voir : Jaccarini, « Vers une théorie du moniteur syntaxique »**).

Toute les tentatives de parcours dans l'automate, auxquelles correspondent des interprétations linguistiques (grâce à un homomorphisme) – c'est-à-dire des hypothèses linguistiques – sont produites, je l'ai déjà dit, au fur et à mesure de l'avancement de la tête de lecture. Or certaines de ces hypothèses, si ce n'est la plupart dans de nombreux cas, sont abandonnées en cours d'analyse, plusieurs d'entre elles étant franchement absurdes sur le plan linguistique ou même au niveau logique. L'idée d'un « rétrocontrôle » : c'est-à-dire d'un contrôle par l'interprétation, vient alors assez naturellement à l'esprit, de même que l'analogie entre langage de haut niveau et de bas niveau[55] ; le langage de (très) haut niveau se situant dans notre cas au niveau de « l'interprétation

constitue un critère important ; l'indéterminisme est maximal pour les mots d'une longueur intermédiaire (5 à 8 graphèmes : « principe de l'accordéon » ; les courbes n'ont pas été établies, *qui auront dû l'être*, n'était-ce le développement intempestif des environnements informatiques – qui ont rendus nos programmes Lisp et McLisp obsolètes).

55. C'est-à-dire ceux plus ou moins proche de ce que les informaticiens appellent le « langage machine » ; les langages de haut niveau étant les langages informatiques, les plus courants (Lisp, C, APL, Fortran (1956), Java, …, la liste est inépuisable) plus ou moins proche du fonctionnement de l'esprit humain ; le langage dit « assembleur » étant intermédiaire entre les deux. Les compilateurs et/ou interpréteurs étant des programmes qui convertissent les langages de haut niveau en des « langages machine » directement exécutables par la machine. Ce qui fait dire à Jean-Louis Krivine (« Lambda-calcul, types et modèles » (Masson, 1990), Introduction, p. 8) que programmer consiste à écrire dans une colonne une suite d'instructions directement exécutables par la machine et dans la colonne d'en face des *commentaires* expliquant ce que sont censées faire les instructions qui sont en face et en garder autant que possible le contrôle. « Compiler » un programme,

linguistique ». Introduire (par des règles linguistiques, logiques, ou probabilistes) des contraintes sur le déroulement de la double pile c'est s'assurer un contrôle sur son évolution.

Une interface linguistique qui irait beaucoup plus loin que la simple interprétation des parcours et des tentatives de parcours serait donc nécessaire si l'on désirait introduire des critères d'élagage ou instaurer même des critères permettant l'évitement de l'exploration d'hypothèses absurdes, ce qui aurait pour effet non seulement de diminuer le bruit mais d'accélérer l'analyse[56].

Il est évident que l'exploration du « parallélisme cognitivo-algorithmique » est bien plus intéressante au niveau de la syntaxe que s'il demeure simplement confiné au niveau de la morphologie, étant donné qu'il y a plus de chance que l'esprit humain fonctionne au niveau des mots ou segments de phrase plutôt que de lettre à lettre. Cette option d'étude fort instructive sera de nouveau d'actualité, vu son intérêt, dès que nos outils de « calcul » seront restaurés (reprogrammation de notre analyseur transparent en Lisp[57]).

## 9.4. *La spécification mathématique de l'analyseur*

C'est dans ce paragraphe que la remarque lapidaire et quelque peu « paradoxale » de Jean-Louis Krivine évoquée, relative à l'« oubli », prendra toute sa résonance (cf. note 55).

En effet, si l'on porte dans une colonne tous les états de la double pile (c'est-à-dire la pile à double entrée et fond unique), dont l'évolution des configurations représente le fonctionnement de l'analyseur pour automate fini non déterministe, et que dans la colonne d'en face on effectue dans la structure d'algèbre non commutative, déjà évoquée, un calcul de quotient (plus précisément si l'on calcule le produit de l'inverse de la forme à analyser par le symbole du vocabulaire auxiliaire représentant l'état initial de la grammaire), on remarquera un phénomène pour le moins curieux : si l'on efface un certain nombre de lignes de calcul, à chaque étape de ce « développement équationnel », pour ne garder à chaque fois que la dernière d'entre elles et que l'on ne retient dans cette ligne que certains symboles d'états de l'automate, ces derniers ne seront alors rien d'autre que les numéros des états de l'automate figurant de part et d'autre du symbole de fond de la pile à double entrée au cours de son évolution durant l'analyse – dans la colonne d'en face (des exemples très simple seront fournis dans le site susmentionné dans « exemple de fonctionnement de l'analyseur »).

en vue de son exécution, consiste alors, dit-il, d'une manière on ne peut plus lapidaire, tout simplement à « oublier les commentaires » (c'est moi qui souligne). Je reviendrai en 9.4 sur cet aspect apparemment paradoxal qui suggère déjà un renversement de perspective (voir Krivine 1993 ; « Mathématique des programmes et programme des mathématiques »*).

56. À condition naturellement que le processus d'optimisation ne soit pas plus complexe que le processus à optimiser : situation classique et récurrente en informatique.

57. Vers 2001, il était impossible de passer dans les deux sens de Lisp à C (langage dans lequel *Sarfiyya* a été développée) ; si bien que nous avions même envisagé de réécrire en C l'éditeur structurel de transducteurs arabes; et d'unifier nos deux approches (celle de Chr. Gaubert et la mienne). De toute manière tout aurait dû être réécrit, étant donné la mutation du contexte informatique.

Le fonctionnement se trouve ainsi entièrement spécifié par le calcul et en plus cette spécification est mathématique (algébrique). Nous avons ainsi non seulement un programme écrit dans un langage de très haut niveau : celui de notre raisonnement humain, puisqu'il s'agit simplement d'effectuer un calcul algébrique mais, en plus, nous disposons d'une preuve de programme – nous avons la preuve que le programme s'achève toujours, à la seule condition toutefois que l'automate de base ne contienne de boucle étiquetée par le symbole ε, lequel représente le mot vide (auquel cas, le niveau d'ambiguïté de la grammaire serait infini[58])– ; qu'il nous fournit tous les parcours possibles c'est-à-dire les coefficients d'ambiguïté de chaque forme. Bien plus, un calcul dans une structure de corps non commutatif est un objet bien identifié qui obéit à des règles strictes de développement : nous savons donc calculer sa complexité ; elle est quasi linéaire et maximise donc (étant donnés les effacements) celle du programme[59].

## 10. Le transfert entre structures et procédures (langue arabe <–> automates)

Ce transfert s'effectue dans les deux sens. Il est possible en effet, sur le plan théorique d'établir un lien formel entre les propriétés structurales et algébriques par lesquelles nous avons caractérisé le système arabe d'une part et les automates et transducteurs sur lesquels nous avons fondé notre approche algorithmique de l'arabe d'autre part. Ces automates et transducteurs constituent une classe de machines équivalentes à celles de Turing, sur lesquelles nous avons néanmoins introduit des conditions de minimalité liées à une hiérarchie rigoureuse.

Les propriétés structurales arabes mises en avant portent sur la commutation des opérateurs de « catégorisation » et de « projection ». La « catégorisation » revient à construire un

58. Dans une structure de corps cela reviendrait à commettre l'« irréparable » c'est-à-dire diviser par 0.

59. Par ailleurs, il existe d'autres spécifications du fonctionnement de cette double pile, c'est à dire de l'analyseur pour automate non déterministe (généralisable d'ailleurs en analyseurs pour une sous-classe importante de transducteurs finis non déterministes, dont nous prouvons qu'ils sont suffisants pour rendre compte des contraintes grammaticales de l'arabe). Ces algorithmes ne sont pas triviaux. Les astuces de fonctionnement de cette double pile ne sont pas tout à fait évidentes. Il est alors très intéressant de remarquer que ces spécifications qui sont différentes les unes des autres représentent néanmoins la même chose (l'évolution de la double pile) et qu'en ce sens elles sont équivalentes. Parmi ces différentes spécifications, celle qui est la plus facile à saisir pour l'esprit humain est celle de plus haut niveau (au sens où les informaticiens parlent d'un langage de haut niveau) : celle que nous venons de décrire, puisqu'elle revient à accomplir un simple calcul dans une structure algébrique rigoureuse que l'on a construite par extension afin de pouvoir justement résoudre toute sorte d'équations (le corps voire l'algèbre non commutative). Il est infiniment plus facile d'effectuer le calcul d'un quotient dans une structure adéquate que d'inventer le « fonctionnement » d'un « objet » (la pile à double entrée) puisque cet objet constitue, à lui seul, une originalité : une sorte de synthèse entre deux structures de données usuelles en informatique, un hybride ; ces deux structures de données étant respectivement la pile (dernier rentré, premier sorti) et la file d'attente (premier rentré, premier sorti) ; en anglais, respectivement LIFO et FIFO (*Last In First Out* ; *First In First Out*). Cet objet est non usuel ; sa structure même nous est pourtant dictée par la nature même du calcul très simple effectué dans le corps non commutatif. Il est tout de même assez joli de constater que « le calcul d'un quotient » est équivalent à un analyseur pour automate non déterministe ; ce qui constitue un résultat non trivial.

ensemble quotient constitué des « classes de congruence syntaxique » sur le monoïde libre des symboles du vocabulaire initial (des mots formels dans le cas de la syntaxe), ou « catégories syntaxiques ». La relation de congruence impliquée formalise le principe distributionnel des linguistes (Bloomfield) : elle exprime que deux mots peuvent être considérés comme équivalents si, et seulement si, ils peuvent commuter sans mettre en cause la grammaticalité de la phrase. La projection revient à réduire le mot à son schème (*pattern*), qui induit également une partition en classe de congruence : les concaténations (réciproquement les segmentations) demeurent invariantes par changement de racines. Cette dernière congruence est compatible avec la précédente en ce sens que la relation de congruence syntaxique est la plus « grossière » que l'on puisse définir sur le monoïde libre constitué à partir des graphèmes de l'arabe et qui sature l'ensemble constitué par l'ensemble des mots arabes graphiques, ce qui a pour corollaire que tout schème, considéré ici comme une classe donnée de mots et non comme un opérateur les engendrant (dualité Structure/ Procédure), se situe nécessairement à l'intérieur d'une catégorie syntaxique dans le cadre de l'ensemble des mots graphique arabes. Nous avons inscrit la modélisation de la morphosyntaxe arabe dans le cadre général d'un *principe d'invariance* dérivant de la propriété morphographique arabe précédente, qui est évidente (invariance par changement de racine dans le cadre de la morphologie) en la généralisant à la syntaxe, à savoir que :

1. Établir d'abord des classes syntaxiques que l'on sous partitionne en schèmes ;
2. Ou bien, à l'inverse, établir des schèmes que l'on regroupe ensuite en classes syntaxiques ;

revient au même.

Supposons que Π et SC désignent respectivement les homomorphismes canoniques associés aux congruences considérées plus haut : la congruence syntaxique et celle associée aux schèmes (catégorisation et projection), alors le principe d'invariance pourra s'exprimer de manière encore plus concise : celle de la commutation de ces deux derniers « opérateurs » :

$$\Pi.SC = SC.\Pi.$$

En vertu de ce principe, nous catégoriserons donc en sorte que la construction de la grammaire ne soit pas affectée par l'opération qui consiste à réduire la langue à ses seuls paradigmes (les schèmes (*patterns*) + les *tokens*).

La possibilité de construire un programme informatique fonctionnant sans lexique n'est qu'une conséquence de la propriété susmentionnée selon laquelle il devrait être indifférent de catégoriser d'abord et de projeter ensuite ou vice versa.

Par ailleurs nous avons vu que la définition des machines de Turing pouvait apparaître quelque peu contingente – tout en s'étonnant qu'un mécanisme aussi fruste puisse représenter tous les calculs effectuables sur machine et même qu'il puisse (théorème de l'énumération, universalité oblige !) simuler tout ordinateur ou ensemble d'ordinateurs (y compris Internet). Les automates à deux piles de mémoires (on en a pas besoin d'un plus grand nombre, ce qui représente en soi une propriété remarquable) en sont équivalents. Ces automates sont fondés sur ceux de moindre complexité, sans piles de mémoire : les automates finis dont la définition

peut susciter le même sentiment de « malaise » évoqué au sujet des machines de Turing et en même temps le même émerveillement devant le fait que des mécanismes aussi élémentaires puissent engendrer des configurations aussi complexes.

L'adaptation d'un point de vue plus abstrait, plus algébrique, nous permet à la fois d'éviter ce malaise de contingence (voir la critique évoquée dès le §1 sur le fait de plaquer artificiellement un formalisme étranger au « fonctionnement » de la langue) et de donner un sens à l'extension du principe d'invariance du niveau linguistique au niveau informatique, d'unifier ainsi le cadre théorique tout en offrant des perspectives pratiques intéressantes. En effet, le calcul du monoïde de transition M(L) du langage L revient à construire directement l'automate déterministe minimal acceptant ce langage[60]. L'automate correspondant à l'étude de David Cohen sera reconstruit en utilisant cette même méthode (où l'on aboutira à la constitution d'un automate de 13 états et 102 transitions) en suivant une « chaîne entièrement automatisée », si l'on peut s'exprimer ainsi, ou plutôt « automatisable ». Cela dit, il n'est nullement nécessaire de reconstituer en Lisp le programme. N'importe quel langage « Turing complet » pourrait faire l'affaire.

Toute séquence d'un langage peut en effet être considérée comme une application d'un segment initial de N dans lui même et dire qu'un langage est reconnaissable par un automate fini c'est, en fait, définir une congruence sur ce langage dont l'ensemble des classes est fini.

Les théorèmes qui établissent explicitement les liens entre les notions de monoïde syntaxique, de congruence et la notion classique d'automates, tels que nous les utilisons pour notre analyse de l'arabe, figureront également dans le site. Au monoïde de transition d'un langage est associé un automate déterministe minimal pouvant reconnaître ce langage. L'automate peut être produit grâce à un transducteur[61]. Le monoïde de transition (= syntaxique) peut être obtenu automatiquement.

## 11. Vocalisation automatique et transduction

Un dernier point mérite d'être isolé, étant donné son importance. Comme on le sait, on l'a vu, l'écriture standard de l'arabe est sténographique. Les voyelles brèves ne sont pas notées, ce

60. On trouvera dans le site le développement de ce type de calcul (sur un exemple de syntaxe tiré de « Modélisation linguistique et théorie des automates »). Cette illustration offre aussi bien un intérêt théorique (ramener un ensemble éventuellement infini de phrases à un nombre fini de configurations) que pratique la construction « automatique » de l'automate déterministe minimal correspondant.

61. Nous l'avions programmé en Lisp ; sa restauration se pose aujourd'hui en termes d'opportunité, d'utilité, de calendrier, de temps de travail, etc. Donc, là aussi, affaire à suivre. Pour l'instant, cette tâche n'est pas prioritaire. Il est également possible d'enrichir le transducteur (minimal) en vue de déterminer les relations de base, qui, associées aux générateurs, définissent le monoïde de transition (isomorphe au monoïde syntaxique). Il peut, en effet, être intéressant d'avoir la possibilité de définir un langage infini (groupe nominal déterminé ou non déterminé, conditionnelles, etc.) par un petit ensemble d'égalités portant sur ce langage de longueur limitées, plutôt que par des règles de réécriture. Par exemple, dans l'exemple évoqué dans le site (un petit sous-ensemble du groupe nominal déterminé), pour vérifier qu'une séquence quelconque appartient à ce langage, il suffit d'examiner ses sous-séquences de longueur 3.

AnIsl 44 (2010), p. 5-38 André Jaccarini
De l'intérêt de représenter la grammaire de l'arabe sous la forme d'une structure de machines finies.

qui a pour conséquence d'augmenter considérablement l'ambiguïté et les difficultés de lecture. De plus, les cas sont marqués souvent par des voyelles brèves, s'il s'agit du singulier, et leur « calcul » ne se trouve pas toujours d'une extrême simplicité [62].

Notre propos n'est pas de discuter de la pertinence de ce système d'écriture mais de constater le phénomène tout en essayant de mesurer ses conséquences en terme d'ambiguïtés et de fournir des arguments objectifs pour les deux types de locuteurs : celui des « arabisants » dont les systèmes de translitération, ne laissent généralement aucun droit à l'erreur (les voyelles brèves ayant la même valeur que les consonnes pleines ou les voyelles longues – il n'y en a que trois) et le système usuel utilisé par les arabophones, qui laisse une certaine latitude [63], laquelle semble suggérer – un fait corroboré par l'expérience, mais qui reste à approfondir – que la lecture (sans signes diacritiques) en premier flux permet de percevoir globalement le sens de la phrase ; une analyse syntaxique plus fine, impliquant du *backtracking* (des retours en arrière) permettant dans un deuxième temps de lever l'ambiguïté.

Néanmoins, ces hypothèses doivent être évaluées. Le système de transduction fondé sur des automates sous jacents, auquel – rappelons-le – on peut faire correspondre des grammaires dite à « attributs sémantiques »* (grammaire de Knuth*), ou encore des « schémas guidés par la syntaxe », lesquels se trouvent associés à des attributs « synthétisés » et « hérités », est particulièrement bien adapté pour cette tâche (flux linéaire dominant « entravé », néanmoins, par du *backtracking* pouvant présenter dans les cas les plus dramatiques des cercles vicieux signifiant des impossibilités de vocalisation – ambiguïtés irréductibles qui sont des cas à étudier pour eux-même). Les attributs synthétisés sont des valeurs qui se propagent du bas vers le haut de l'arbre représentant la structure de la phrase (on dit que l'on décore l'arbre ou bien encore qu'on lui associe un arbre « sémantique ») et les attributs hérités sont ceux qui se propagent du haut vers le bas. Transposé au plan du flux de lecture, cela revient à dire qu'il existe des valeurs (ici on s'intéresse aux voyelles brèves) qui sont « synthétisées » au fur et à mesure de l'avancement de la tête de lecture, alors que certaines unités lexicales ne peuvent acquérir leurs valeurs définitive

62. Le grand grammairien Sibawayh cite un exemple d'ambiguïté irréductible, ressemblant à un gag, qui a aussi le mérite d'attirer l'attention sur le fait qu'en arabe littéraire l'emplacement des mots dans la phrase est relativement plus libre ; cette liberté étant compensé par un « marquage » morphocasuel plus important. L'exemple est le suivant : Akala (a mangé) ʿĪsĀ (Jésus) MūsĀ (Moïse) ; on ne peut savoir si c'est Jésus qui a mangé Moïse ou l'inverse étant donné l'incompatibilité phonologique de la marque du cas direct (voyelle *u* brève) avec le dernier phonème de ʿĪsĀ ou MūsĀ. L'ambiguïté demeurant naturellement la même par permutation de Issa et de Moussa (la marque de l'indirect étant le *a* bref).

63. Ce constat de l'usage de la graphie standard par les grammairiens arabes depuis des siècles, ainsi d'ailleurs que l'organisation de leurs dictionnaires nous incline naturellement à penser qu'ils percevaient (et continuent de percevoir) les consonnes comme étant les éléments d'un « squelette » qui serait le « support » principal du sens (plus spécifiquement les consonnes radicales, les autres pouvant appartenir à un schème (pattern), forcément discontinu, si l'on tient compte des voyelles brèves qui y interviennent, lesquelles ne peuvent jamais être radicales ; le schème dans son *intégralité*, qui est une forme non « concaténative », étant seul (avec la racine) susceptible de posséder une ou des valeurs sémantiques sans qu'évidemment cela signifie pour autant un sens précis- tout comme la racine d'ailleurs). Dans cette remarque nous nous situons dans le cadre de la morphologie dite *saine* (voir §2). Je ne signale que des faits et je me tiens volontairement à l'écart du problème des figements lexicaux.

(vocalisation ou « sens ») que par effet de retour, une fois que la lecture complète de la phrase a été accomplie. Knuth a étudié les cas de cercles vicieux et mis au point (1968) un algorithme pour les éviter. S'il y a impossibilité, on se retrouve alors dans la situation bien connue des informaticiens du « *deadlock* », qui se produit lorsque deux processus sont en attente l'un de l'autre. Nous en avons déjà parlé plus haut dans un autre contexte. Il s'agit d'ambiguïté intrinsèque [64].

La méthode de décoration d'arbre se trouve illustrée dans « Modélisation linguistique et théorie des automates »** et elle intervient directement dans le calcul du transducteur minimal et déterministe remplissant la fonction de synthétiseur d'analyseur morphologique descendant de formes arabes. Toutefois seuls des attributs synthétisés (qui se propagent du bas vers le haut ou de manière équivalente sont synthétisés parallèlement au flux de lecture) ont été considérés. Par ailleurs, il a été établi que le passage du système morphologique de base (morphologie saine) au système général [65] ne requérait pas d'attributs hérités (cf. http://automatesarabes.net).

Dans « Approche algorithmique de la grammaire de l'arabe », un transducteur morphologique *ambigu* a été exhibé, fonctionnant mot à mot (les voyelles dépendant du cas n'étant pas prise en compte, puisque le raccordement avec la syntaxe n'a pas été implémenté). Les coefficients d'ambiguïté variant de 1 (dans un nombre significatif de cas [66]) jusqu'à 12 ; plusieurs types d'ambiguïté (de différents niveaux) pouvant se combiner.

En effet, un raccordement avec la syntaxe est nécessaire non seulement pour faire baisser le niveau d'ambiguïté mais également pour pouvoir être en mesure de vocaliser la fin des mots.

De tels outils, qui sont à reprogrammer, peuvent avoir des applications intéressantes. Le rédacteur d'un texte en arabe peut être informé en temps réel du niveau d'ambiguïté de la forme introduite pour se voir suggérer un certain nombre de solutions (totales ou partielles) pour lever l'ambiguïté selon le niveau de l'utilisateur (EAO), rien qu'en cliquant. La technologie fondamentale existe déjà ; tout le reste n'est que question d'ergonomie et d'interface, ce qui, dans le domaine de la rédaction assistée et de l'EAO, est fondamental. Il va sans dire qu'il s'agirait d'un outil améliorable par introduction de la syntaxe et aussi par apprentissage.

L'outil conceptuel (le transducteur de vocalisation interactif) serait évidemment du plus grand intérêt pour répondre à la question que l'on s'était posée au début de ce paragraphe, à savoir celle de mesurer, ou plutôt de discuter scientifiquement de la pertinence des deux points de vue : respectivement celui des arabisants et, de l'autre, celui des arabophones.

64. Nous en avons parlé au sujet du « moniteur syntaxique » qui est censé optimiser l'analyse morphologique, en envisageant le cas extrême où les deux processeurs morphologiques et syntaxiques sont en attente l'un de l'autre.

65. Inspiré par le contexte, je préférerais les expressions de « morphologie sous-jacente » et de « morphologie enrichie », qui, par ailleurs, indépendamment de la théorie des automates, sont suffisamment parlantes par elles mêmes.

66. Cependant, aucune statistique n'a été établie ; il s'agissait d'une étude de faisabilité.

## 12. Le Retour aux *tokens* : de la syntaxe à la sémantique

Il est à noter que les éléments figés dont nous avons parlé ne sont pas simplement des révélateurs de structures syntaxiques mais peuvent aussi constituer les marqueurs de relations discursives, surtout si l'on ne s'interdit pas de considérer certaines suites de *tokens* ou encore des associations d'éléments figés non nécessairement contiguës. La fréquence de certains de ces marqueurs – qui peuvent donc être discontinus – ainsi que leurs combinaisons avec d'autres marqueurs, qu'il nous serait possible de détecter par des fonctionnalités de *pattern matching* (recherche de motifs), lesquelles sont aisément spécifiables par des expressions rationnelles (voir dans le site les expressions qui, dans *Kawâkib*, ont permis de spécifier la recherche de certains « motifs » (*patterns*) ainsi que celles que nous avons utilisées pour extraire différents types de négations, etc.) peuvent dans de nombreux cas être des révélateurs de relations sémantiques. Certaines de ces combinaisons sont particulièrement discriminantes si l'on cherche à « caractériser » les textes selon leurs types ou, tout au moins, à les trier selon les probabilités de leur appartenance à certaines classes sémantiques : il peut être intéressant par exemple de repérer les textes de type argumentatif ou de nature « polémique », etc., relativement à certains sujets ou thèmes : ces derniers étant alors plutôt repérés par des fréquences de lexèmes ou de racines particulières. La « nature » des textes, ainsi que l'existence de certaines relations discursives, sont ainsi distinguées du « contenu » à proprement parler (cf. *infra*, l'article de Claude Audebert).

Les expériences que nous avons menées dans ce domaine ont été concluantes (voir la rubrique « Résultats » dans http://automatesarabes.net). Un commentaire détaillé de ces résultats sera produit. La méthode expérimentale sur laquelle nous nous appuyons est celle de la rétroaction continue (*feedback*). À ce propos, on peut consulter Medar 09 et Lrec 2010 (voir Bibliographie**).

## 13. L'algorithmique associée à la recherche d'information et à son filtrage

L'attention du lecteur a déjà été attirée à plusieurs reprises sur les risques d'explosions combinatoires, d'ambiguïtés, et de complexité algorithmique. Nous avions justement remarqué que les automates sont des objets très « simples », pouvant cependant produire des « configurations » extrêmement complexes. Le monde du Calcul est aussi foisonnant que celui du Vivant. Dans ce même contexte, nous avions eu aussi l'occasion de souligner un aspect épistémologique essentiel de l'ordinateur : fondé sur le calcul, il permet de faire du Calcul un objet d'expérience. Notre recherche algorithmique est, elle aussi, grâce notamment au *feedback*, expérimentale.

À ce propos, rappelons que le « fonctionnement » d'un automate (d'une machine de Turing en général) n'est pas la chose très simple (car « mécanique » justement), que l'on pourrait s'imaginer de prime abord puisque aucune autre « machine » n'est capable de le prévoir ! (problème de l'indécidabilité de l'arrêt d'une machine de Turing).

Il est difficile de « prévoir » la longueur des développements des algorithmes ainsi que leur comportements. À partir d'une analyse fonctionnelle précise des besoins il est ardu d'établir

un calendrier rigoureux pour le développement algorithmique. Il n'y a peut être pas lieu de développer ici tous les arguments qui plaident en faveur d'un autre *renversement* de paradigme. Disons simplement qu'il peut être pénalisant, de manière générale, de définir et de fixer au départ des fonctionnalités linguistiques précises, et de chercher ensuite à les mettre en œuvre informatiquement en développant des algorithmes. Je suggérerais plutôt de travailler dans l'autre sens, c'est-à-dire : développer d'abord une algorithmique (linguistique) consistante et essayer d'imaginer ensuite les fonctionnalités correspondantes. On construit d'abord un programme dont on connaît la structure, le comportement, etc., et on « imagine » l'ensemble des fonctionnalités dont le programme constitue l'implémentation. En somme, une fois que l'on dispose du code (le programme) on essaie, dans un deuxième temps, d'imaginer l'ensemble des questions dont il est la réponse ! Ces considérations font inévitablement penser à J.-L. Krivine* que nous avons déjà cité. Concluons enfin en remarquant que cette approche peut aussi présenter un autre aspect intéressant : plusieurs fonctionnalités en apparence très différentes peuvent néanmoins appartenir à la même catégorie dans la mesure où elles reposeraient toutes sur le même algorithme ou famille d'algorithmes.

## Conclusion

La formalisation algébrique de la structure des opérateurs linguistiques arabes offre la perspective d'une programmation structurée. Nous avons critiqué tout au long de cet article la programmation *ad hoc* qui présente, entre autre, le défaut d'une complexité que l'on n'arrive pas toujours à maîtriser et à évaluer. La constitution d'une banque d'opérateurs est un objectif à long terme. Notre objectif à court terme est de construire une petite bibliothèque d'opérateurs en vue de la recherche d'information. Pour la bibliographie nous renvoyons le lecteur à http://automatesarabes.net. Rappelons également que les résultats figurant dans ce site, concernant l'extraction d'information, feront l'objet d'un examen détaillé et seront accompagnés de commentaires dont une synthèse sera présentée dans un prochain article.

AnIsl 44 (2010), p. 5-38 André Jaccarini
De l’intérêt de représenter la grammaire de l’arabe sous la forme d’une structure de machines finies.