自然言語処理 31－3
（1982. 5. 21）

ヨーロッパにおける計算言語学の様子

辻井 潤一
（京大・工学部）

1. はじめに

CNRS（フランス国立科学研究所）の客員研究員として，1981年6月より9ヶ月間，グルノーブル大学の機械翻訳研究グループ（GETA）でMT研究に従事する機会を得た。本報告では，この滞在中に知り得たヨーロッパ，とくにGETAでのMT研究の現況について報告する。GETAの組織的概要，および，GETAのMT方式の基本構成等については，同じGETAに滞在しておられた大分大学・岡田先生の報告[1]があるので割愛し，本報告では，GETAのシステムを使ってみた個人的感想，および，GETAの全体的な動向についての印象を，いくつかの観点から整理して述べることにする。

2. 全体的な動向と印象

GETAのソフトウェアシステムは，MTシステムの第2世代としては，最も良く整理の行き届いたものの1つであり，第2世代の典型である。これが，システムを使ってみた第一の印象である。MTにとって必要と思われる機能が，ATEF・ROBRA・TRANSF・SYGMORの各プログラム言語の仕様に適切に反映され，また，文法作成者が対話的に言語モデルを作成するのに便利な様々なコマンドが，MT研究用のモニタARIANE78に用意されている。20年以上に渡る彼等の研究成果が，これらソフトウェア群に結実している。しかしながら，その一方で，この間に米国・日本等で研究されてきた自然言語理解研究の成果，例えば，意味処理や文脈処理の成果は，MTには時期尚早であるとして，ほとんど取り入れられていない。単に言語モデル中に取り入れられていないというだけでなく，ソフトウェアの仕様そのものが，これらの処理に対する配慮を欠いている。この点でも，ARIANE78は第2世代の典型である。

SYSTRANに代表される第一世代システムが，多くの欠点を抱えながらも，大規模な辞書データを蓄積し，実用化レベルでの評価の時期を迎えているのに対して，第2世代システムは，TAUM計画・EUROTRA計画にみられるように，研究室レベルでの検証を終了し，大規模化の時期を迎えつつある。研究から開発への移行の時期を迎えているといっても良い。GETAにおいても，露仏翻訳を中心として，いくつかの言語対（仏－英，独－仏，英－中・日・マレー）を対象としたシステムが現在なお作成されつつあるが，報告者の印象では，ARIANE78を使ったこれらの研究は，新しい考え方を模索するというよりも，近い将来の大規模化に備えて，GETAの現在の方式が，多くの言語についてうまく働くかどうかを検討するための予備実験的な性格が強いように思えた。したがって，これらの言語対を対象とするシステムの作成においても，言語モデルと処理ストラテジーを各システムが出来るかぎり統一することに格別の注意が払われている。すなわち，解析・トランスファ・生成の各段階でどのような種類の情報をどのような構造を使って表現するか［言語モデル］，及び，これらの構造や情報をどのような方法で，また，どのような順序で活用するか［処理ストラテジー］，の両面に渡って

規格化しようとしている。最も長い歴史を持つ露仏翻訳システムにおいても，これまでトランスファ段階で処理されていた多くのことを，多言語間翻訳を意識して，解析・生成の段階に移し変える等，新しい処理手法の開発よりも，システム概念をより明確にすることに努力が傾けられている。実用化・大規模化の観点から，現在計画中の次の2つのプロジェクトは注目に値する。

(1) PILOT：現在のGETAの言語モデル・処理ストラテジーを言語専門家に教育し，大量の文法作成者・辞書記述者を養成するなど，システム開発のための専門機関を作ろうとする計画。
(2) ARIANE X：4年間の使用経験から，ARIANE78の問題点が明らかになってきたので，これらを踏えて新しいソフトウェアを開発しようとするプロジェクト。民間のソフト会社の技術者も参加しており，完成後は，実用的な移行性のあるソフトとして流通させようと考えている。現在，最終仕様を書く作業を進めている。

上記2つの計画は，本来GETAがEUROTRA計画によって遂行しようとしていたものであるが，諸般の事情から，GETAはEUROTRAのグループとは一線を画し，独自に計画を進めようと考えているようである（GETAがEUROTRAを開発プロジェクトと把えていたのに対し，最近のEUROTRAは，かなり研究的色彩の強いものになりつつある。プロジェクトに対するこの基本的態度の差が，GETAが離脱しつつある原因の1つであるように思えた）。

3. ARIANE78

これまでにもしばしば紹介されてきたように，GETAのMTシステムは，形態素解析（AM）・構造解析（AS）・単語トランスファ（TL）・構造トランスファ（TS）・構造生成（GS）・形態素生成（GM）の6つの段階からなり，各段階は，それぞれATEF（AM）・ROBRA（AS，TS，GS）・TRANSF（TL）・SYGMOR（GM）の4つのプログラム言語で書かれたプログラムによって処理される。プログラムといっても，実際には計算機をほとんど知らない言語専門家によって作成されるもので，ATEF・TRANSF・SYGMORによる記述は，それぞれ，形態素解析用辞書・2言語間の単語対照辞書・形態素生成辞書の形式をとり，また，ROBRAによる記述も，適用順序など多少計算機処理的色彩があるが，通常の意味でのプログラムというよりは，記述形式が厳密な文法規則の集合といった感じが強くなる。このような各言語で書かれた辞書や文法を実行形式にコンパイルしたり，実際に翻訳を実行したりするための会話モニタが，ARIANE78である。ARIANE78は，次のような機能を果している。

(1) 辞書・文法のコンパイル
(2) 各言語で開発された部分システムの結合と実行
(3) 各処理段階での中間結果の保持
(4) 文法・辞書について，いくつもの版を保持し，これを自由に利用者コマンドにより組み合せる。（例えば，独立に開発された解析部分と生成部分を組み合せることにより，多言語間MTシステムを比較的簡単に実験できる。）

(5) 各段階での辞書を相互参照し，各段階での未知語リストを出力する。
(6) 種々のモードでの処理過程のトレース

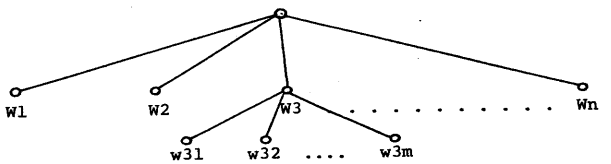
この他，ARIANEの機能ではないが，各プログラム言語のコンパイラも，かなりキメの細い cross reference を出力したり，コンパイル時のエラーチェック機能が強力で，実行前に文法の不統一を検出する等，文法を漸次的に開発するための多くの機能を持っている。しかしながら，ARIANE78自体は，あくまでMTシステム開発用の援助システムであり，実用化段階のMTシステムをサポートし，pre-editingやpost-editing を含む翻訳の各段階での人間の介在を許すようなシステムにはなっていない。GETAにおいても，実用化後のMTシステムのための man-machine 機能を開発することは，今後の研究課題となっている（現在，3ème cycle の学生2人 ── 日本の修士課程と博士課程の中間ぐらいか？ ──が，この課題にとり組み始めているが，ARIANE78及びそのもとでの各プログラム言語が，必ずしもこの種の機能を実現するのに適してはおらず，例えば，Bringham－Young大のITSのような介在すら容易ではなさそうである）。また，人間が与えた辞書記述・文法記述をコンパイルするという方式は，記憶効率（5000語の辞書項目 ── 形態素解析のもとになる項目数が5000，彼等の言によると普通辞書の項目数にすると3万項目に相当 ── を持った露仏システムが，辞書を含めて2Mbyteで動作できる）や処理速度の点で優れているが，柔軟性や修正の容易さの点で劣ること等，ヨーロッパでのハードウェア環境の悪さが，ARIANE78の設計思想に大きく影響しているようにも思えた。人間のMTシステムへの介在についても，ワードプロセシング技術が格段の進歩をとげている米国や日本の研究グループの方が有利な立場にあるといえよう。ARIANE78の規模を表1に示す。

4. 言語モデル

GETAの言語モデルで特筆すべきは，Multi-level Analysis Tree（MAT）の考え方である。解析・トランスファ・生成の各過程において，常に1本の木構造がその処理対象となる。入力単語列は，ATEFによるAMの結果，終端節点に入力単語とその辞書情報のついた平らな木構造（図1）に変換され，以後，この木構造がAS，TL以下の処理をうけて，出力文にまで変換される。MATは，節点が属性－属性値対の集合で修飾された木構造であり，基本的には，これまで拡張CFG文法で使われていたもの（Robinson[2]は，この種の構造をannotated tree str. と呼んでいる）と同じである。

• 規模	• Primitve instruction数にして，約200,000step
• 使用言語	• アセンブラ言語（80%）
	• PL360 （12%）
	• PL1 （8%）
	• CMSのCommand Language
• ソフトウェアの内容	
	• ROBRA，ATEF，SYGMOR，TRANSFのコンパイラ（70%）
	• 使用者との対話機能（15%）
	• 実行過程の管理機能（15%）
• 処理は，仮想記憶方式によって行なわれ，各ユーザは，2MBの領域で仕事を行なう。グルノーブル大学の最も大きな翻訳システム，ロシア語－仏語システムも実行時は，2MBで実行される。この2MB中には，翻訳に必要なすべての辞書も，ロードされている。辞書は，コンパイルされているために，かなりコンパクトになっているという（written text と比べ4～5倍小さくなっている）。	

表1



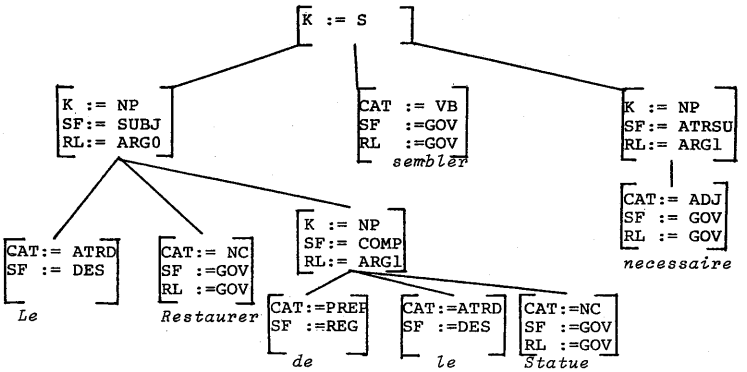
(注) W1，…Wnの節点には，辞書情報がつけられる。
W3には，辞書項目としてm個の曖昧さがあることを示す。

図1

しかしながら，他の多くのシステムで使われた木構造が，意味マーカ・数・性・格パターン等の，節点の代表する語や句の内在的な性質（そして，それらは理論的には非終端記号で表現することができる(3)）を属性－属性値対の集合で表現していたのに対して，GETAのMATでは，これらの内在的性質の他に，その語や句が構造中でどのような役割を果すか，という語や句の機能的な性質も，この属性－属性値対の集合で表現される。GETAにおいては，言語表現を構造的に記述するのに，

(1) 句のまとまり方の種類（Syntactic Category－K）
(2) 句の統語的役割（Syntactic Function－SF）
(3) 句の意味的関係（Semantic Relation－RS）
(4) 句の論理的関係（Logical Relation－RL）

の4つのレベルの記述を想定しているが，これらのすべてがMAT中の属性－属性値対で表現されることになる。



La restauration de la statue semble necessaire.

図2

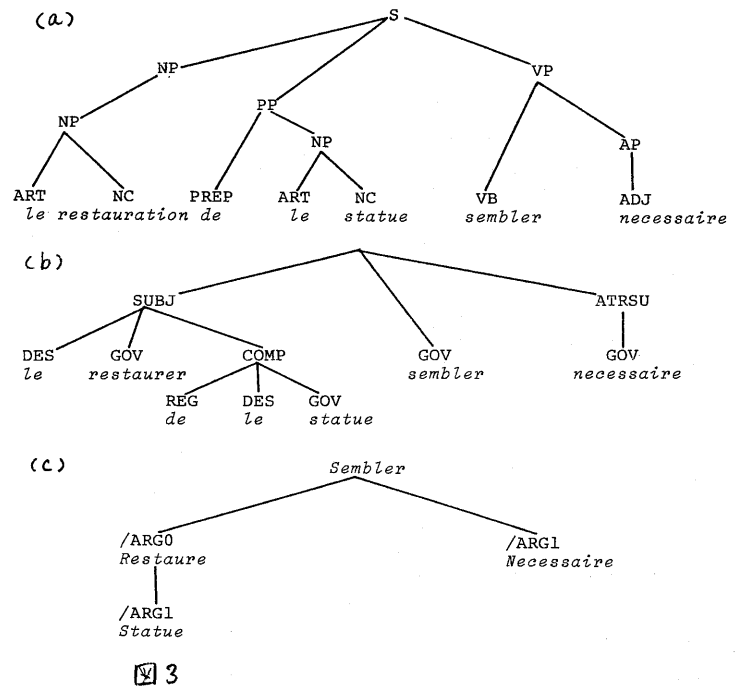


図3

図3 (a)・(b)・(c)は，図2のMATで表現された各レベルでの構造だけを抜き出して示したものである。GETAのMATの基本的考え方は，このように，異なるレベルの情報を，一つの種類の木構造にとじ込めて表現しようとするもので，他の多くのシステムが各レベルで異なった表現形を想定し，各表現形の間を変換によって結びつけていたのと対照的である。GETAがこのような立場をとった理由は，

(1) 解析過程では，あらゆるレベルでの情報を同時に参照する必要があり，したがって，あらゆるレベルの情報が1つの表現形の中に記述されている必要があること，

(2) 翻訳は，1つの言語内で表現されている，あらゆるレベルでの情報をなるべく保存した形で，もう1つの言語に移し変える作業であり，'意味'だけを移行すれば良いわけではないこと，

があげられる。もちろん，一つの種類の木構造の中に，すべてのレベルの情報を節点中の記述を使って閉じこめるという考え方には大きな制限がある。例えば，意味構造としてShankのCD構造のような，統語構造から大きく離れた構造を考えることはできないし，また，そこまで極端でなくとも，King (4) の挙げた，

John drank a bottle of wine
John break a bottle of wine

のように，統語構造と意味構造が乖離してしまうこともある。しかしながら，GETAのMATは，理想的とはいえないが，現時点の技術で実現可能な現実的な構造であるといえよう。さらに，MATは，文全体の「完全な」解釈結果が得られなくても，その時点で得られている情報を使って，なんとか出力文を生成する，というsafty screenの考え方にも有効である。

以上のことからわかるように，GETAにおける言語モデルは，節点に付加される属性－属性値対としてどのようなものを設定するか，によって決定される。現在，GETAでは，多少の言語対依存な属性を除く，主要な属性について，どのような値を持つべきかがほぼ決定されており，開発・実験中の各MTシステムをなるべく共通な言語モデルで行なおうとしている（実際，新たな属性を導入したり，また，既存の属性の値を変更することに対しては，非常に慎重である）。

5. プログラム言語 — Robra

GETAのプログラム言語には，Robra以外にATEF・TRANSF・SYGMORの各言語があるが，ここでは，解析・トランスファ・生成の各段階において最も重要な役割を果しているRobraについて述べる。Robraによるプログラムは，次の要素からできている（実際のシンタックスについては，文献(4) また，規則の例については(1)参照）。

(1) 木構造の節点に付与される属性とその値の宣言
(2) 節点の属性値に関するチェック，および，属性値の割当てのためのマクロ・プロシジャの定義
(3) sub-grammarとsub-grammar間の遷移関係を規定するネットワークの定義
(4) 木構造の変換規則の定義

(1)は，4で述べた属性－属性値の可能な組み合せを宣言するもので，文法作成者に彼の作ろうとしている言語モデルを明確に意識させる役割を持ち，この宣言を使うことにより，コンパイル時のエラーチェックが行なわれる。(2)は，(4)の木構造変換規則中で指定されるもので，よく使われるチェックや値の割当てを定義する（実際に使ってみた印象では，規則記述が簡潔で見易くなること，言語モデルの変換に際して，マクロ定義を変更するだけで，個々の木構造変換規則を修正する必要がなくなることなど，便利な機能である）。Robraプログラムの主体は，(3)と(4)である。(4)の木構造変換規則は，

(A) 規則適用の可否を決定する条件　(B) 適用後の構造記述

の2つの部分からなり，(A)はさらに，

(A－1) 木構造の幾何学的形状（schema tree）
(A－2) 各節点につけられた属性－属性値対についての条件
(A－3) 節点間の属性－属性値対の一致条件

に分けて記述され，(B)も，

(B-1) 変換後の木構造の幾何学的形状 (image tree)

(B-2) 各節点への属性値の割当ての部分に分かれる。

(A-2)・(A-3)・(B-2) においては，IF－then－else－の形式が使用できるなど，通常のプログラムに近い指定が許されている。入力文を表現している木構造中に，1つの規則が複数個所で適用できる場合があるが，この場合には，文法作成者のモード指定に従って，並列に複数個所での木構造変換が行なわれる (例えば，Total モードでは，すべての個所で，Cut モードでは，木構造中で Frontier ―図4参照,― を成している節点集合に対して etc)。使用した経験では，(A-1) の幾何学的形状の指定や並列実行モードの指定等にかなり豊かな機能が用意されており，また，変換後の木構造に対して，再び同じ規則を適用させるといった recursive モードもあり，思いついた規則は大方表現できる。汎用の Tree Transducer としての機能は十分あるといってよい。しかしながら，意味処理として比較的素朴なものと考えられる意味マーカ間の階層構造が表現できないことや，節点につけられた属性－属性値間の相互関係がうまく記述できないために，規則記述が非常に冗長になり，規則の意図が読みとり難くなるという欠点がある (AI研究で使われた，階層構造にもとづく property inheritance の機能，あるいは，一連の属性－属性値対をフレーム構造としてまとめ上げておく機能等は，Robra の記述をより理解し易くするのに役に立つと思われる)。

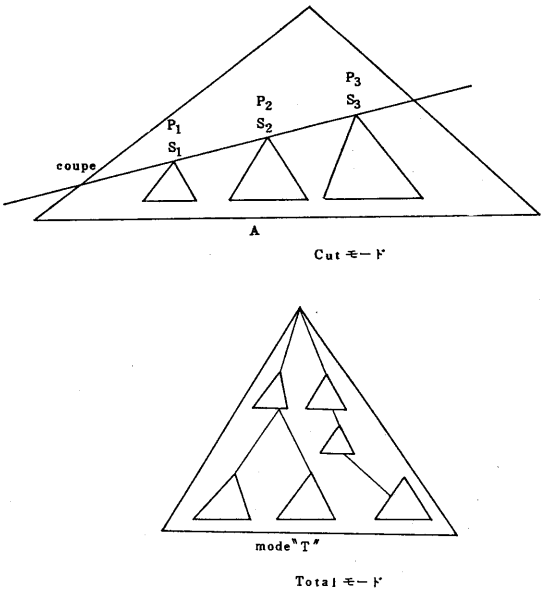


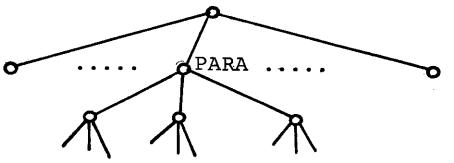
図4

(3) の sub-grammar と sub-grammar 間の network を定義する機能は，通常の拡張 CFG や DCG 等のシステムと Robra の著しい相異を示すものである。すなわち，Robra においては，文法作成者が予め定義した network に従って，処理は順次的に進められる。しかも，sub-grammar は，必ず1つの入力木構造をとり，1つの出力木構造を次の sub-grammar に渡すことになっているので，Robra での処理は，文法作成者の定義した network によって細かくコントロールされ，決定的に (deterministic に) 進行することになる。文献(1)で紹介されたように，Robra は，sub-grammar 単位での後戻り制御機構を持ち，これによって，非決定的な (non-deterministic な) 処理機構をサポートしていると主張しているが，

- 1つの sub-grammar 中には，通常複数個の木構造変換規則が定義されること
- 1つの規則が，木構造中の複数個の個所で同時に並列的に適用されることがあるため，sub-grammar 単位の後戻りはほとんど役に立たず，むしろ後戻りのために弊害が生じることの方が多い (ATN では，1つの遷移規則ごとに後戻りし，かつ，規則は一ヶ所にしかかからないので，この種の弊害は生じない)。

したがって，Robra の後戻り機構は，特殊な例外を除いて，ほとんど使われてい

ないといって良い。Robraによる処理は，処理手順が文法記述者によって厳密にコントロールされた，決定的処理になっている。もちろん，よほどの工夫をしない限り，文解析を決定的手順で行なうことはむつかしく，Robraを使ったシステムでは，曖昧さが生じた個所では，図5のPARAのように，木構造中に曖昧さを明示する節点（Tactic Nodeと称する）を設けているが，この節点は明らかに通常の言語構造を示す節点と性質が異なっており，このような節点の存在は，以後の木構造変換規則の記述を非常に複雑なものにしてしまう。GETAのグループでは，曖昧な文に対して，解析結果が組み合せ的に爆発し，相互の関係がよく判らない解析結果が一度に多数個出力されることを極度に警戒しており（MTシステムとして実用にならないとして），このために，非決定的な処理機構をシステム側に持たせることを拒否しているが，研究者によって議論のわかれるところであろう。



(注)節点PARAより下の3つの節点は，同一部分に対する，異なった3つの解釈を示す．

図5

6. おわりに

GETAでの研究活動の様子をかなり私見を混じえて紹介した。全体的な印象として，彼等は独自な構想でMTにとり組んでいること，ソフトウェア：言語モデル・処理ストラテジーともに，よくきが行き届いており，（外部から見ると多くの欠点を持っているにしても）一応システム全体として統一がとれており，簡単な翻訳実験ならば，1人の研究者が数ヶ月かかれば開発できるだけの道具立てができていること等，注目すべき点も多い。しかしながら，一方で，彼等の枠組内でできること，できないことがはっきりしてきたこと，したがって，GETAの枠組においては，新らたなアイデアを考案するよりも，その枠組内で実用規模の大規模システムを作成すべき時期が到来していることを痛感した。MTの研究としては，彼等の到達した枠組を根本的に検討し直して，次世代のMTシステムを構想するか，あるいは，彼等の枠組（もちろん，多少の改良は行うにしても）内で，大規模・実用システムを前進に踏み切るかの2つの方向があろう。いずれにしても，GETAのシステムは，その内部を詳細に検討する価値のあるシステムである。

〔参考文献〕

(1) 風田 直之：「ヨーロッパにおける自然言語処理の現状 ― グルノーブル大学における機械翻訳を中心として ―」，自然言語処理30-2，情処

(2) J. J. Robinson：'DIAGRAM：A Grammar for Dialogues,' Vol.25, No.1, C. ACM, 1982

(3) G. Gazdar：'Unbounded Dependencies and Coordinate Structure', Vol.12, No.2, Linguistic Inquiry, 1981

(4) M. King：'Design Characteristics of a Machine Translation System', IJCAI, 1981