TR-I-0044

Record of Six Work Sessions on Concepts, Methods, and Tools from Existing Running Real-Size MT Systems

ワークセッション記録:
MTシステムの概念、手法、ツール

Christian Boitet
クリスチャン・ブワテ

October, 1988

ATR Interpreting Telephony Research Laboratories
ATR 自動翻訳電話研究所

Foreword

This is a record of six work sessions organized by Professor Christian Boitet during his stay at ATR.

Using his long research experience in the field of Machine Translation, Professor Boitet reviewed the concepts, methods, and tools of existing or previously running real-size MT systems. He discussed them with the ATR researchers in order to work out the "best set" of principles for an Automatic Interpretation System, taking into account existing dangers, illusions, and solutions.

He prepared more than a hundred transparent sheets for the work sessions, which are the main sources of this record. However, some errors might have occured in the editing process of the hand-written transparent sheets into this typed format. I would like to apologize to the author for the remaining errors.

(T. Aizawa)

Six Work Sessions on Concepts, Methods, and Tools from Existing Running Real-Size MT Systems

Christian Boitet

ATR Interpreting Telephony Research Laboratories, Osaka
and
GETA (Study Group for Machine Translation)
Joseph Fourier University & CNRS, Grenoble

TABLE OF CONTENTS

INTRODUCTION

Why refer to MT (Machine Translation) systems if we attack MI (Machine Interpretation) systems?

- Prof. Nagao's warning not to follow MT design blindly
- Written & spoken language, translation & interpetation:
  A common basis
- Research in NLP as a finite-state device
  much has been done .....
  .....and forgotten
- Usual scientific practice to
  build on accumulated knowledge .....
  .....without rediscovering the wheel
- B. Vauquois' contribution & evolution;
  - Coincidental with the history of modern MT
  - Illustrates the bychological side

Historical Perspective

Outline of Presentations

A. Through a retracing of B. Vauquois' contribution & evolution

1. The researcher & his ideas

1960~65	Applying formal language theory
1965~70	Integrating modern semantic theories
1970~75	Defining modular grammars and heuristic methods
1975~80	Inventing multilevel (M-)Structures
1980~85	"Static grammars" for specifying the string-tree correspondences
1985~	Towards a new treatment of ambiguity

2. The implementer & his methods

   Erudition, eclectism & pragmatism in linguistics
   specialized languages adapted to the tasks emergence of lingware engineering methods

B. With a rapid survey of some MT systems

1. SYSTRAN & variants
2. CETA & METAL
3. SUSY & LOGOS
4. METEO & TAUM-AVIATION
5. ARIANE-78 & -85
6. Japanese systems
7. EUROTRA ?

C. A summary of the important issues

1960~65 Applying formal language theory

Grammar in CNF

$A \rightarrow BC$

$A \rightarrow t$

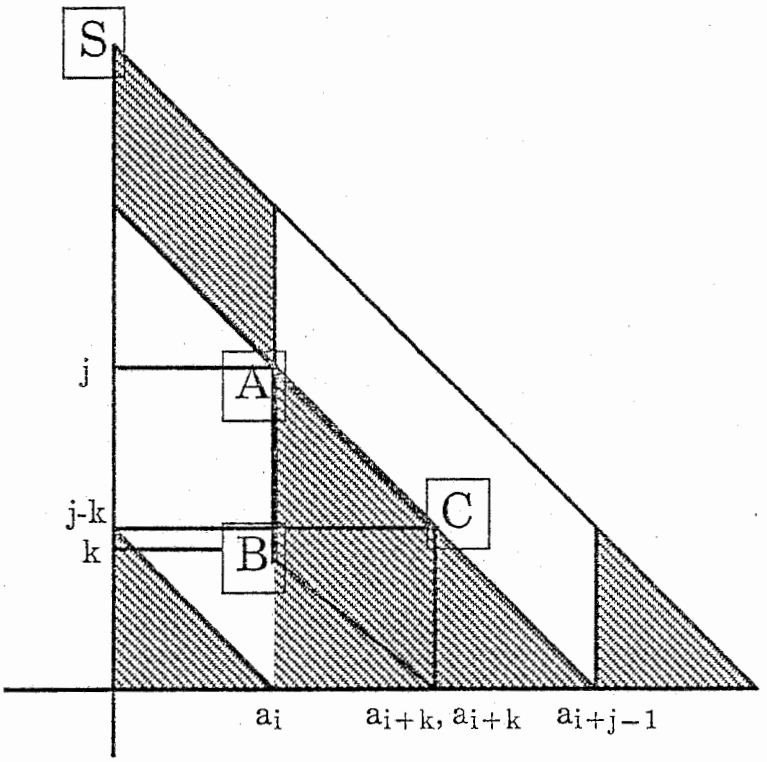
With attributes on symbols

- Normal (gender, number .....)
- "Vehicular" (for discontinuines)

And validations/saturations on rules

R_{A1}: $A \rightarrow BC$ / P(a(B), a(C))
/ a(A): = F(a(B), a(C))
/ - (R_i, R_j, ...), + (R_r, ...).

Cocke algorithm



with some improvements.

1965~70 Integrating modern semantic theories

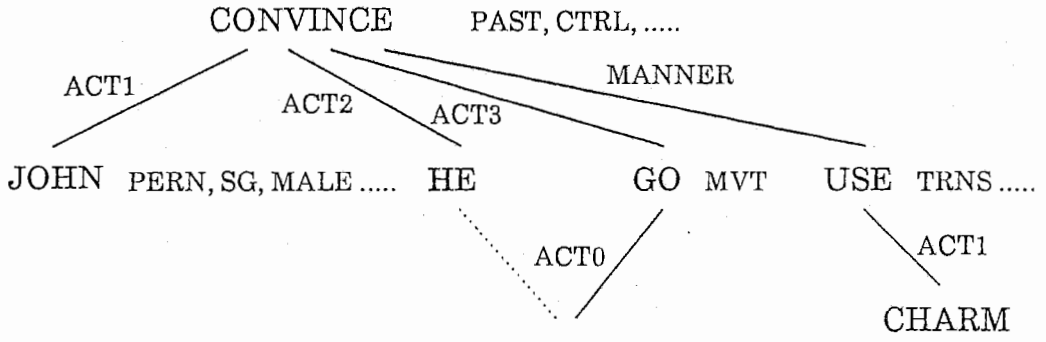
- Need for a "PIVOT",
  More universal than binary attributed phrase-structures

- Ideas from
  Tesnière : Actants
  Mel'čuk : Meaning-text
  Šaumjan : Hybrid pivot

- Necessity to
  Select one binary structure
  transform it into PIVOT form

The PIVOT

A "universal" structure (grammatical & relational symbols) with a language-dependent lexicon.



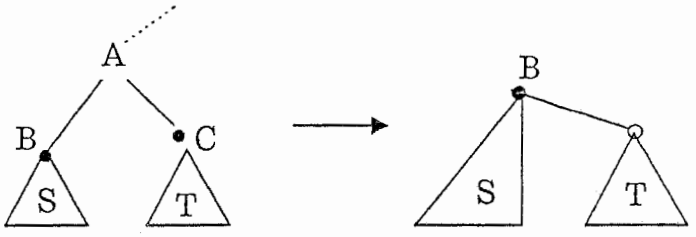
John convinced him to go using charm

Selection of initial structure and obtention of PIVOT structure

- Preference rules
  (For same covered substring)
  Not implemented yet as metalanguage

- Transformation into an initial
  Dependency structure

A → BC / - / - / B



- Then, transformations expressed in a special metalanguage

1970~75 Defining modular grammars & heuristic methods

- Intractability of big rule systems for human control
  Still applicable today

- Difficulty in expressing preferences

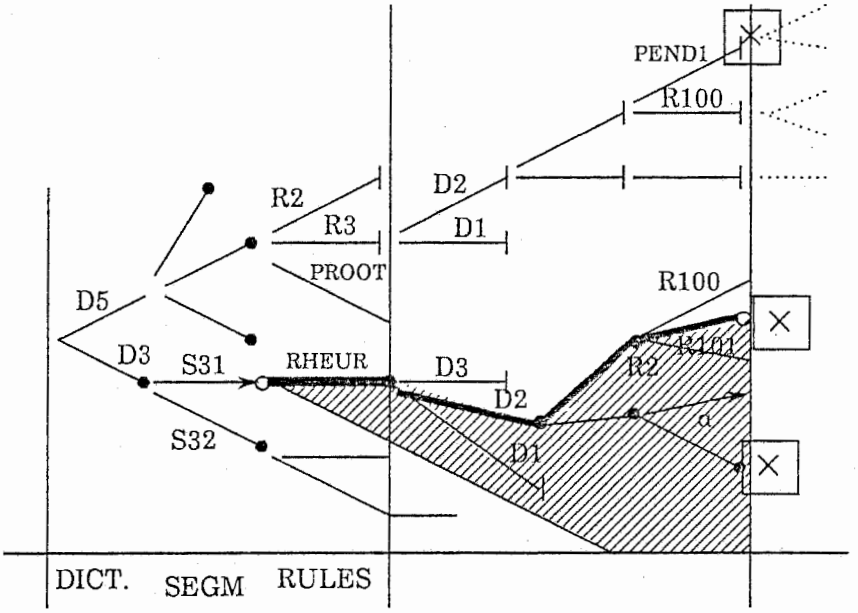
- Belief that direct methods based on heuristics would be quicker & more adequate.

⇒ Analysis in ARIANE-78

Morphology	:	Combinatorial & Heuristics (ATEF)
Syntax / Semantics	:	Procedural & Heuristics (ROBRA)

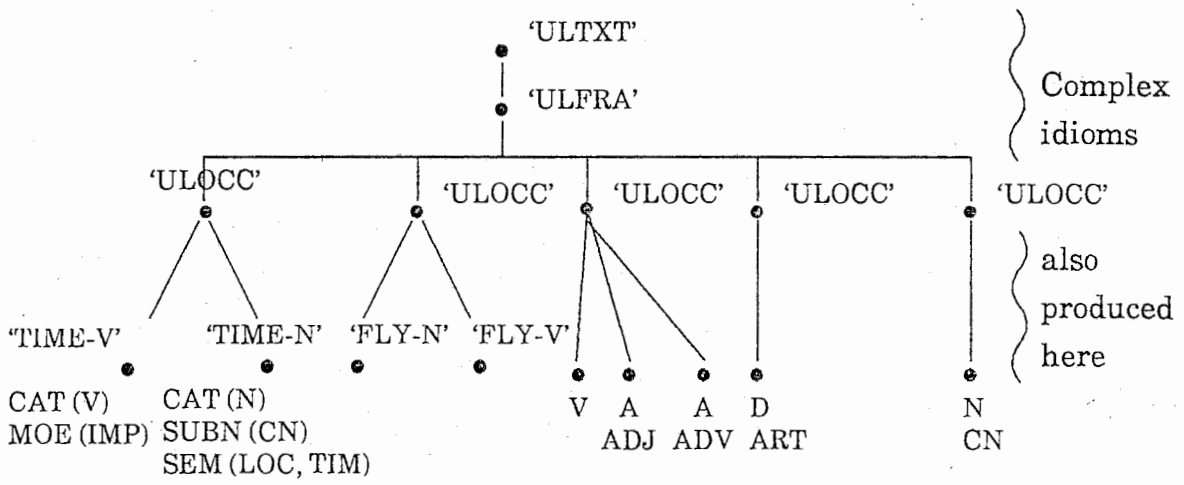
ATEF

- Computation : ND / * / Depth-first & Heuristics



- Pruning of search tree
- Possibility to suppress previous, tentative results.

- Output : Factorizing tree



Time flies like an arrow

ROBRA

Tree-transformational systems

- Control graph

- Ordered grammars on nodes
- Conditions on the arcs
- ND / 1 / Depth first
  & Context-senstive rules (on trees)
  Recursive calls
  Parallel application of rules
  Guarded iteration & recursion

  See GRADE for comparison.

1975~80 Inventing multilevel (M-)Structures

- Layers of interpretation levels

Syntactic categories	CAT	Words
	K	Clauses
Syntactic functions	SUBJ	
	OBJ1	
	OBJ2	
Slightly different	ATROBJ	
From dependency	CPAG	

Relations which relate only words, not words / groups with words / groups.

Logical argument places	ARGO
	ARG1
	ARG2
	TRL01
	TRL12

Semantic relations	INSTR	AGENT
	CAUSE	"TOY"
	ACCOMP	LOCAL
	MEASURE	CONSEQ
	QFIER	
	QFIED	

- Other levels

Actualisation	Time / Mode / Aspect
Discourse type	Quest / Indir / Cmd .....
Theme/Rheme	– Not implemented yet
Sem. features	Concr, Abstr, Docum,Presfied, Animate, Human, ...

+ Lexical units (Sem. derivational families)

Considering M-structures as implicit generators of other structures

- Classical C -
  F - Structures
  SPA -

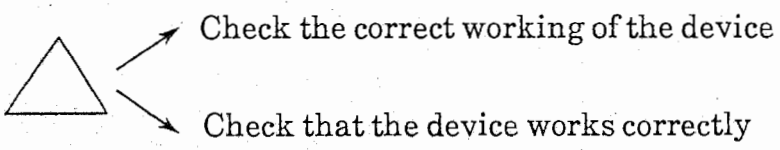
- Alternate structures:

Encoding	Ambiguities
	Doubts

& During transfer

Encoding	Advices or Orders
	To the generator

- At the beginning of generation, 1 structure encodes all paraphrases legitimate (For translation)

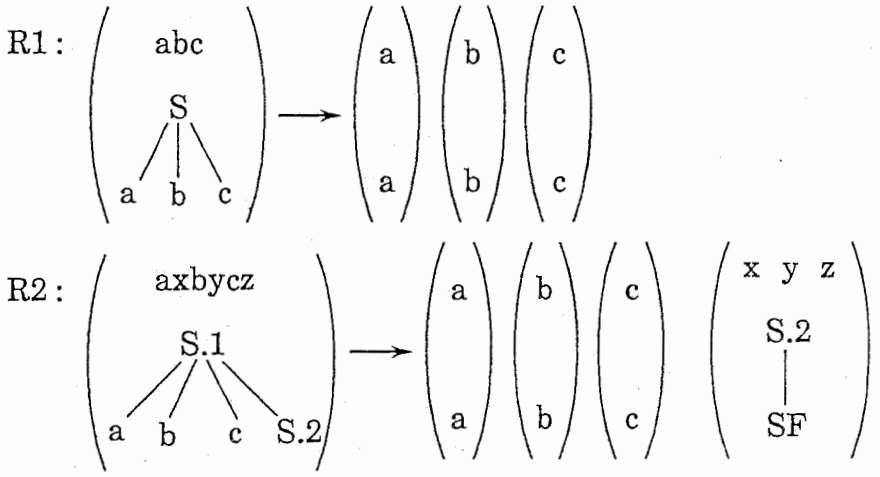


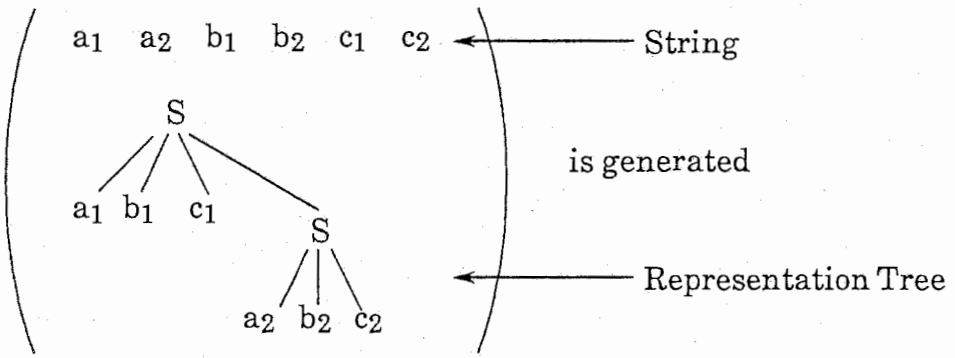
Usually, but not necessarily, lexical units are not changed during generation.

1980~85 **Specifying string-tree correspondences by static grammars**

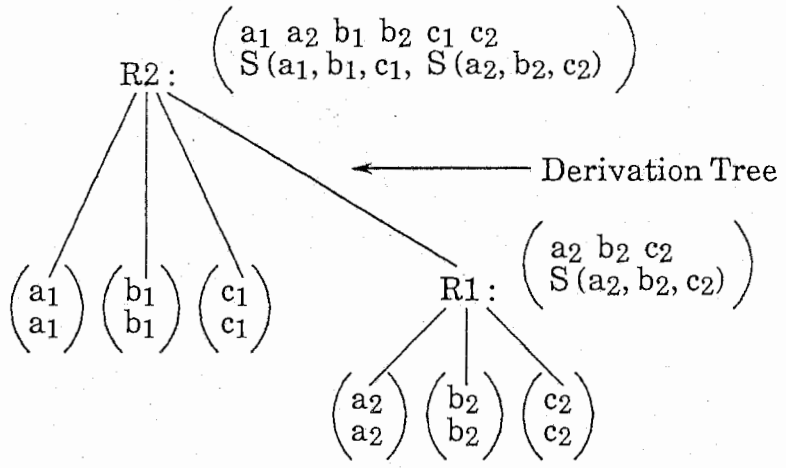
- Necessity of specification formalism, in view of the size of the (procedural) grammars, even if modular.

- Used in practice since 1983 (national project)
- A small example on a formal language (using a simplified syntax)





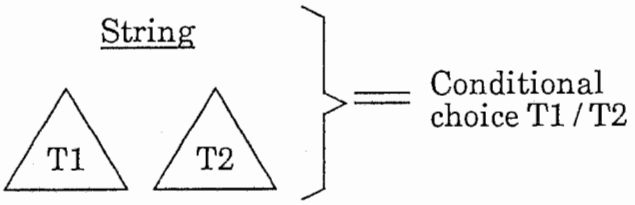
By the following derivation



See some analogy with XYZGs ··· . But in Zaharin-86, Unification -- Identification.

1985~ **Toward a new treatment of ambiguity**

- In practice, "static grammars" also contain (up to 75% !) preference rules



(In procedural analysis grammar, if T1 is chosen, T2 may be encoded.)

- Desire to treat ambiguities directly at the level of the final description of the linguistic computation
  - Open area, research theme
  - Somewhat analogous to more ancient techniques, but in the spirit of today's "expert systems".

The implementor & his methods

Erudition, eclectism, pragmatism in linguistics

- Not a blind follower of some ideology ("Mainstream" or other)
- No linguistic theory actually covers more than a fraction of the linguistic facts / phenomena.
- Good ideas may come from different sources
  Theoreticians and grammarians
- Theories & usually accepted concept are sometime inexact and must be corrected / improved.

Specialized languages for linguistic programming adapted to the various tasks

- No unique computational
  Framework is good for everything
  Very general SLLPs tend to be very inefficient for specific tasks such

as morphology

Maybe a useful consideration also in the case of MI systems.

- Building SLLPs on subrecursive formalisms, with the possibility
  - To allow for some "liberty switches" static detection
  - To evaluate the order of complexity of the associated computations.

The emergence of lingware engineering

- "Lingware" coined at GETA to reflect the parallelism with software engineering.

- Development & maintenance problems

  Some figures derived from CAT-NP quite in accordance with B. Vauquois' estimates.

- From all points of view

  Linguistics (Content: Grammar / Lexicons)

  Computer science

  Formalisms & mathematics

  Building MT systems necessitates to fix the state of the art at some point

  ⇒ Success of METAL

  Failure of eternal prototyping / EUROTRA)

Basic Architectures and Concepts

A rapid survey of real-size MT systems

SYSTRAN & Variants

Fixed flow of control

No possibility to backtrack or process several solutions in parallel

No explicit grammar rules

No .....

But

Enormous amount of work put in dictionaries

Experience on extremely large corpuses

Incredible speed

Usable for

- information-gathering in general
- translation, in very specific settings (XEROX .....)

CETA & METAL

- Comparable in computational technique
  - All-path of analysis
  - Attached transformations for analysis
  - Big dictionaries

But some differences

Preference rules	/	Weights
Hybrid PIVOT	/	Hybrid transfer (T + G mixed)
Dependency structures	/	Constituent structures
All-or-nothing	/	Fail-soft

Implementation of the SLLPs

Assembler	/	Lisp

- CETA faster (80000 ipw / 1.5 Mopw)
- METAL (1981~86) 15 years later, more usable & up-to-date
- Becase ceta model was much more theory-oriented ..... ? → Discussion !

SUSY & LOGOS

- Both ~ 1.5 generation:
  SLLP for dictionaries, not for grammars
  →Difficult to adapt the grammars
- Both usable (LOGOS commercially, SUSY exper. at IAI)
- Both relying on combinatorial approach for analysis
- Both unclear linguistically & computationally
  Structures manipulated & produced not independently defined, but in terms of the implementation.
- Both, but LOGOS even more, give great emphasis on dictionary-building facilities.

METEO & TAUM-Aviation

- METEO — Q-systems (A. Colmerauer)
  95% quality since 1983

In 1985, switched to fail-safe approach

1988 : 120 w/mn on a lap top
300,000 pages translated since 1977 (~ 100 p/day)
(or 25,000 w/day)

Sematics-based analysis
(Categories such as "Meteorological Event")

Q-systems : General "Addition"
Production system (on Q-graph)

- All rules apply on all possible occurrences; arcs used in L.H.S. are marked.
- If & when application stops, marked arcs & "loose ends" are erased; another Q-graph is output.

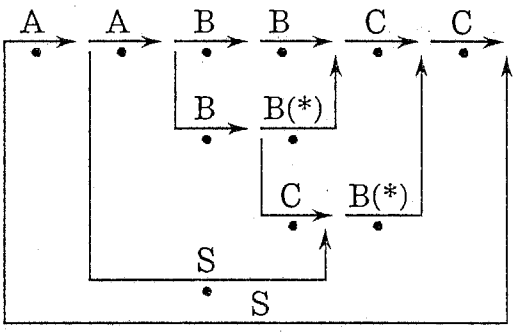
$\{A^nB^nC^n \mid n \geq 1\}$ in Q-Systems

Input: -01- A + A + A + B + B + B + C + C + C -02-

(G1) Parameter-Free

A + B + C	= = S
A + S + B(*) + C	= = S
B + B	= = B + B(*)
B(*) + C	= = C + B(*)

Q-Grammar



(G2) Counting Q-Grammar (Output = $S(1^n)$)

```
      A + B + C              = = S(I)
A + S(U*) + B(*) + C         = = S(I, U*)          ** U* is a list.
      B + B                  = = B + B(*)
      B(*) + C               = = C + B(*)
```

Also reversible
(produce $A^nB^nC^n$ from $S(1^n)$).

(G3) Structuring Q-Grammar (Output = $S(A_1, B_1, C_1, S(.....))$)

```
A* + B*     = = S(A*, B*) / A*–IN–$$ABC –AND– B*–IN–$$ABC
A* + S(U*) = = S(A*, U*) / A*–IN–A, B, C
          –REQ–     HERE, –01– S(A1, A2 ... An, B1, ... Bn, C1, ... Cn)
S(U*, A, V*, B, W*, C) = = S(U*, V*, W*, S(A, B, C)) / U*–IN–A –AND–
                                                       V*–IN–B –AND–
                                                       W*–IN–C
S(U*, A, V*, B, W*, C, S(X*)) = = S(U*, V*, W*, S(A, B, C, S(X*))) / ––.
```

Also reversible (produce $A^nB^nC^n$
from S(A, B, C, S(A, B, C, S(....., S(A, B, C))))).

TAUM-Aviation (1976/77~1981)

- Aim at large sublanguage
  (Maintenance manuals for aircraft)
- Several SLLP's

Q-systems	(Generation & 1st version of morph. an.)
REZO	(ATN-like transducers of R-graphs binary features added)
LEXTRA	(Lexical transfer + transformational rules)

- All or nothing approach
  (Acceptor-based) → High quality
  But failure (1981)

Main reason:
Dictionaries (transfer) too costly because of complex transformations

Also:
ATNs bad for large grammars
(Delicate heuristics & "PATCH-UPs")
Too detailed semantic feaures
→ increase in cost
→ decrease in adaptability

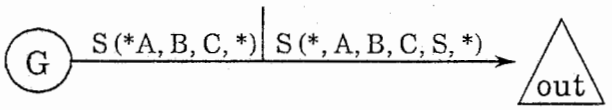
ARIANE-78 & 85

- In both, structural analysis using tree-transducer (in ROBRA)
- Pattern-matching facilitates
  heuristic programming, but decreases speed ($\sim \times 40$)
- Dictionary aspect not strong enough in Ariane-78:
  → Improvements in ARIANE-85 ("Lexical expansion" optional phases)
  → External "neutral" lexical data base + support utilities (CAT-NP)
- Some combinatorial step could be added to reduce ambiguities before structural analysis.
- However, tree-transducer gives more flexibility than attached transformations or ATN-like mechanism.

$\{A^nB^nC^n \mid n \geqq 1\}$ in ROBRA

Input : $X(A_1, ..., A_n, B_1, ..., B_n, C_1, ..., C_n)$

Control Graph :



(Slightly simplified syntax)

G (H) : R1 (G / R2, R3 / X (Y))

Search from "High" to "Low"

R2 (G / R2, R3 / X (Y))

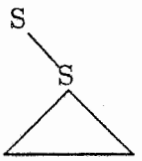
R3 ;

"Validated Nodes" during guarded recursion

R1 : X & R ($U, A, $V, B, $W, C, *) / A : 'A' ; B : 'B' ; C : 'C' ;
right $U : 'A' ; $V : 'B' ; $W : 'C' ;
X : 'X'

== X ($U, $V, $W, Y (A, B, C)) // Y : 'S' ; X : 'S'.

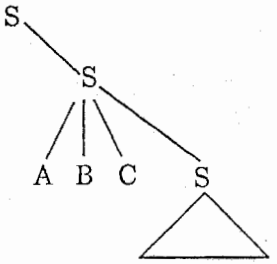
** May be added, tree is :



R2 : (LEVEL = 1) X & R ($U, A, $V, B, $W, C, Z, *) / ..... ; X : 'S' ; Z : 'S'

== X ($U, $V, $W, Y (A, B, C, Z)) // Y : 'S'.

** Succeeds at the end when tree is :



R3 : (LEVEL = 0) X (*, Y, *) / X = Y = 'S'

== Y.

Japanese Systems

- Pivot — FUJITSU , NEC
- Transfer — Others

  - Analysis

    Tree-transducers — MU
    HITACHI

    ATNs — TOSHIBA

CF-based	IBM JAPAN
	KDD(KATE)
"Unification-based"	NTT (LUTE) ?

- Dictionaries (Large)

Direct	HITACHI, KDD, ...
Neutral	MU (JICST)
Conceptual	FUJITSU, NEC

Enormous work, perhaps multiplied if pivot (→ sort of normalization across languages).

Summary of some important points in MT (of written texts)

Basic architecture & concepts
- The external characteristics for decision-makers
- Linked with intended use/evolution

Representation of the units of translation
..... And what they are (sentences, paragraphs .....)
¿ Can developers understand them ?

Specification / application of grammars & dict.
The declarative / procedural
Competence / performance Controversy

Specialized languages for linguistic programming
A classification by the cost of running
Comparable applications

Organization of the whole MT system
The importance of size
Various environments necessary (tests, debugging, prod.)
How to achieve modularity at user-lever (lingware / texts)

External characteristics seen by potential users (interested in technique)

- Basic architecture & concepts

  ① Direct
  Transfer
  Pivot

  ② Structure of grammar / dictionaries
  Explicit / Buried in codes
  Modular / Big block
  → Domain
  Typology

  Expert knowledge available ?
  Mixed with linguistic Knowledge. (TAUM-METEO)
  Separated ("coupling" 2G + KB)
  Integrated by compilation (CMU)

  ③ Treatment of ambiguities
  Combinatorial / Heuristic / ..... None
  Batch / Interactive

Representation of the Units of Translation

- In existing MT systems
  - Computer structures
  - Linguistic content / interpretation
  - Dubious / incomplete / ambiguous analysis
- What about (unmentioned) F-structures ?
  - Derivation trees vs. representation trees
  - Geometry (lists) vs. Algebra (sets)
  - Keep (attributed) trees with F-structures, if any (& why not ?)
- A sketch of the beginning of a tentative approach for a convenient RS for MIS
  - Some motivations
  - Levels of description, contents of cells
  - Open problems (& possibilities)

More Internal Characteristics

- Representation of the units of translation

① Computer structures

↓
- Strings
- Lattices / charts
- Labelled / decorated / "featured" trees
- General graphs / semantic networks
- Logical formulae

More "AI" oriented

± Encoding of ambiguity → important for MT

② Linguistic content
- Levels of interpretation (CAT/K, SF, LR, SR)
- Grammatical properties
- Actualisation, determination, quantification

Semantic features / restrictions
Discourse-related (theme, emphasis, ...)

③ Representation of ambiguity /doubt .....
± Encoding of strategic indications for later processing

What Units of Translation ?

- In all systems but GETA's, sentence by sentence
  → Quite empty talk about anaphora resolution, discourse structure, etc.

- At TAUM (~ 1972-73), some experiments with "transferring" to next .....
  → Not quite useful

- In GETA's systems, (~ 1/2 page or more), still
  → Only used for intersentential anaphora, sentence splitting / joining
  → Not enough linguistic knowledge for really more discourse-related proc.
  → Long term memory needed
  e.g. definition of acronyms at beginning

- In MIS ? One can suggest
  Current complete utterance + All past dialogue
  + Abstractions (SPK, TOPIC)

Here, we center on the representation of the current occurrence

LENGTH

HYP.		
	In dialogue,	≦ 4/5 lines, or 40/50 words
	In conversation,	≦ 15/20 lines, or 150/200 words
	In meeting,	≦ ..... Maximum allowed by the chairman !

Anyway, a decomposition in paragraph-like units seems reasonable.
Then use previous limit.

2 KINDS

Input or output representations
"Interface Structures"

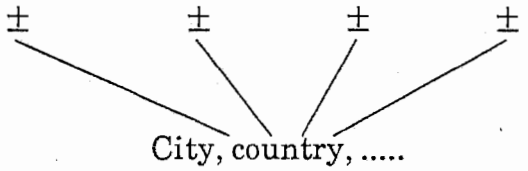
Internal representations
"Working Structures"

Elements / Constructors of Structures

Strings — of characters (not so simple)
of complex elements ("Nodes")

Binary features

+HUM	+CONCR	+ANIM	+LOC	+PRSN
−HUM	−CONCR	−ANIM	−LOC	−PRSN



Typed attributes

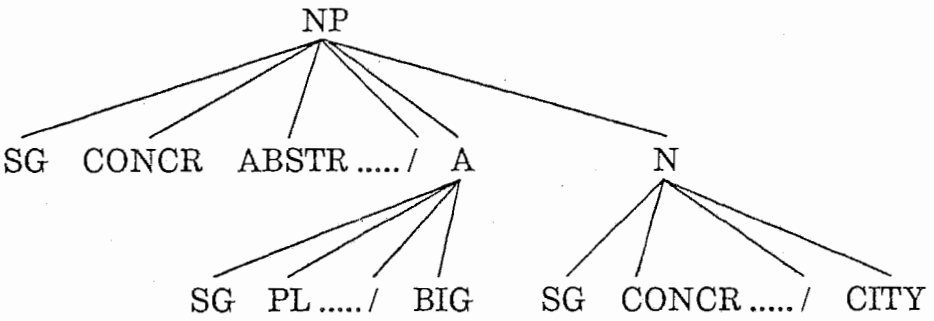
GNR : nex (MAS, FEM, NEU)
SF : exc (SUBJ, OBJ1, OBJ2, CPAG, ...)
LEV : exc (RESPECT, POLITE, NEUTRAL, FAMILIAR, INJURIOUS)
CAT : exc (N : nex (CN, PN), A : nex (ADV, ADJ), ...)

Hierarchy may be implicit (→ Prop. list)

CAT : exc (N, A, ...)
SUBN : nex (CN, PN)
SUBA : nex (ADV, ADJ)

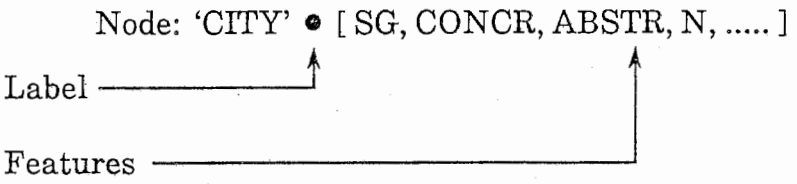
Trees

Labelled



Problem : Algebraic information mixed with the geometric structure

Attributed
(Binary features)



Decorated
(Typed attributes)

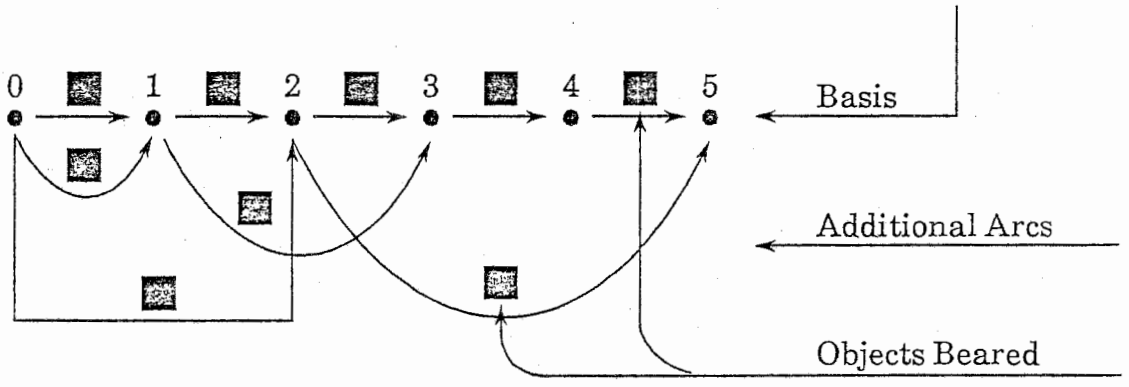
Node: 'NP' • [NB (SG), SEM (CONCR, ABSTR),
HEAD ('CITY')]

Many systems (incl. SUSY, GETA, SALAT-Heidelberg .....)
Use the label to express :

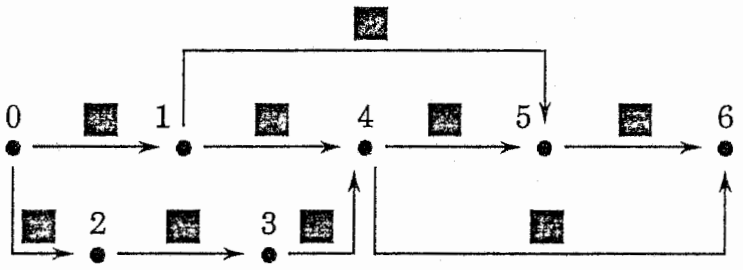
- The "Main category" on non-terminals
- The "Lexical reference" on terminals

Charts / Q-Graphs

Charts (KAY, MIND system, ~1965)
Loop-free graph with linear basis



Q-graphs (Colmerauer, Q-systems, 1970)



Objects : Labelled Trees (Q-system)
Attributed Trees (TAUM-AVIA)
Dotted Rules (CHART PARSERI)
Etc.

Other types of interface structures
Logical formulae (McCORD, LMT)
Sematic network ¿ (FUJITSU, ATLAS II) ?
Others ?

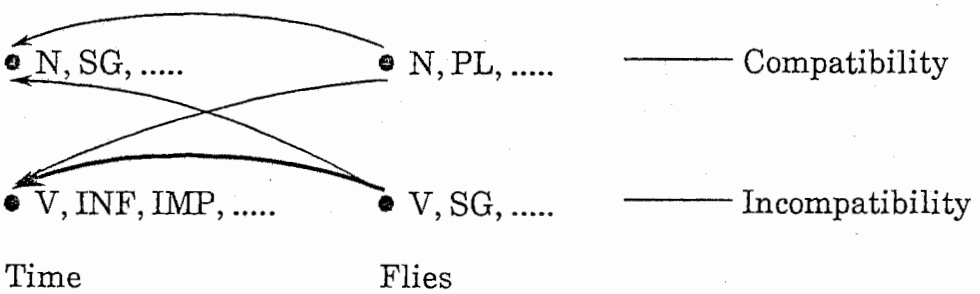
Working Structures

Same, plus

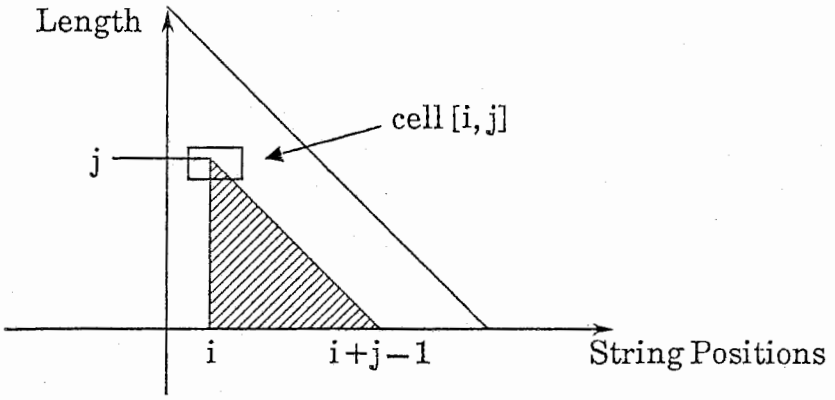
String of decorated nodes (I Pointers) SYSTRAN

Lattice of (strings of) dec. nodes GETA

(Morph, Analysis)



Matrix of	Tree Elements	Cocke	(CETA)
	Dotted Rules	CF-Variants	(METAL, ...)



Stacks / Arrays of trees	REZO	(TAUM)
	ROBRA	(GETA)
	GRADE	(MU)

☒ Interface Structure ☒ Working Structure

Structures / Steps	Strings			Labelled Trees					Charts	Q-Graphs		Other Graphs	Logical Formulae
	Simple	Decorations	Pointers	Simple	+Binary Features	Decorated	Pointers	Weights		Simple	+Binary Features		
Input Text	~ALL									TAUM T-AVIA			
Morph. Analysis										TAUM		ARIANE	LMT ?
		SYSTRAN				ARIANE MU			CETA	METEO TAUM-AVIATION			LMT
Struct. Analysis			SYSTRAN			(STACK OF) ARIANE MU			CETA METAL ~ALL CF-based	METEO	T-AVIA		LMT
Interface Source Structure			SYSTRAN (?)	METEO (1 ARC)	T-AVIA (1 ARC)	ARIANE MU METAL ...	CETA			METEO	T-AVIA	FUJITSU (?) (SEM. NET LUTE	LMT
Transfer						ARIANE MU Others ?	CETA			METEO	T-AVIA	ATLAS (II LUTE	LMT
Interface Target Structure						ARIANE MU	CETA			METEO TAUM-AVIATION		?	?
Struct. Generation						ARIANE MU	CETA	CETA		METEO T-AVIA			
						ARIANE MU		ARIANE (Num. attributes)		METEO T-AVIA			
Morph. Generation		ARIANE CETA								METEO T-AVIA			
Output Text	~ALL									METEO T-AVIA (1 part.)			

Linguistic "Content"
(rather, "INTERPRETATION")

Derivation tree / Representation tree

Grammar bound — Language bound
~ Always projective — Independently defined
Better for MT
(As for Compilers)

Lexical Information

- Lemmas
- Lexical Units — MOVE / MOVER / MOTION / MOVABLE

1 LU = 1 Derivational Family (MOR / SYN / SEM)
Several Meaning Subfamilies
(OBSERVE / OBSERVANCE)
(OBSERVE / OBSERVATION)

Lexical Functions (à la Mel'chuk)
Not (yet) used in MT [Power]
Lexical units give good (translational) paraphrasing
+ Annex INFO
- Valencies / Case Frames
- Semantic Features

.....

Grammatical Information

- Morphosyntactic Info (Gender, Number, Category ...)
- Determination
- Actualization (Tense / Mode / Aspect .....)
- Syntagmatic Category (1st "Level of Interpretation")

Relational Information

SF Syntactic Function

(More than dependency relation, because relates a word or a group of words to same.)

LR Logical Relation

(ARGO, ARG1, ARG2, ARG12, TRL10, TRL21, ...)

SR Semantic Relation

(CAUSE, INSTR, CONSEQ, MEASURE, ACCOMP, LOCAL, TOY, DEST, ORIGIN, ...)

SR ≠ LR

SR difficult to compute on arguments

For translation of arguments, LR is enough (+ SEM features).

A Key^{INSTR}	Opens	a Door^{TOY}	
One^{AGNT}	Opens	a Door^{TOY}	With a Key^{INST}
A DOOR^{2}	Opens		(With a Key^{INST})
ARGO	PRED	ARG1	

Levels of Interpretation in Interface Source Structures

MT Systems \ Levels & al.	K	SF	LR	SR	Sematic Features	Lexical Units	Traces
SYSTRAN	+	+	–	–	50↗400↘~100 used	–	+
LOGOS	+	+	–	?	~100 or more	–	+
CETA	○	○	+	+	~25/30	+	–
METEO	○	○	–	+	~30/50 (Domain-Spec.)	–	– ?
ARIANE	+	+	+	+	~40/50	+ (Variants Exist)	+
TAUM-AVIA	+	+	–	+	~100? (Domain-Spec.)	–	+ ?
METAL	+	+	–	?	~50 (?)	– ?	+
MU	+	+	–	+	~40	–	+

○ Used for analysis but not in interface structure

Dubious / Incomplete / Ambiguous Analysis

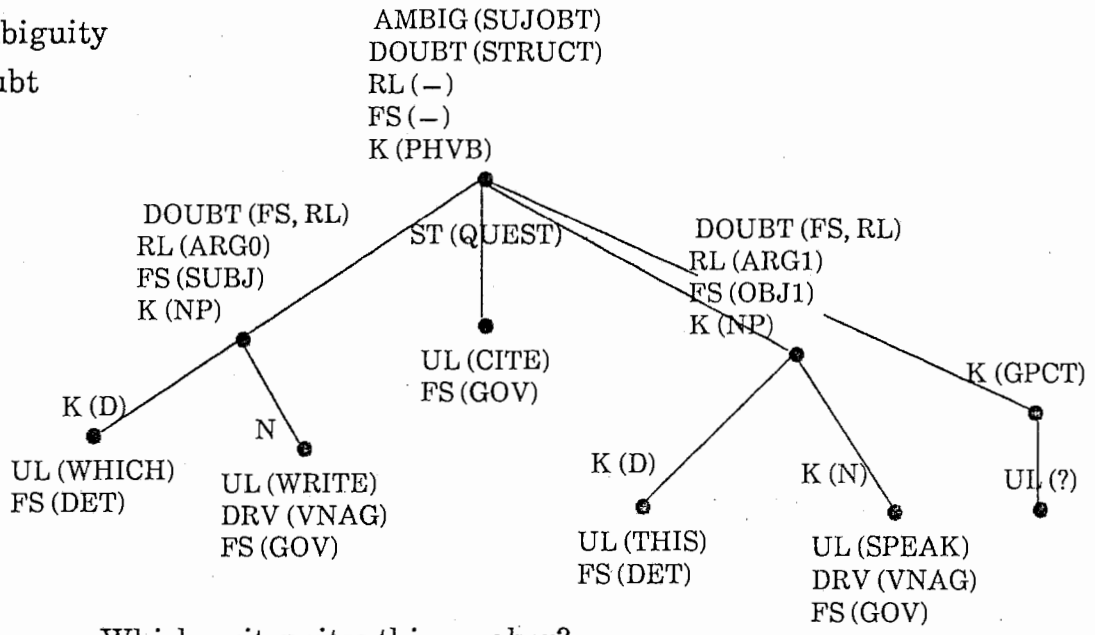
- Almost all systems choose a "Disambiguated" solution, thereby losing the ability to
  - Transfer the Doubt / Ambiguity
  - Treat these phenomena

- Pr. Vauquois' Solution

  Use Interface Structures as Objects, & Encode
  - Marks of Doubt
  - Some Ambiguities / Polysemies
  - Indications for later processing

Examples

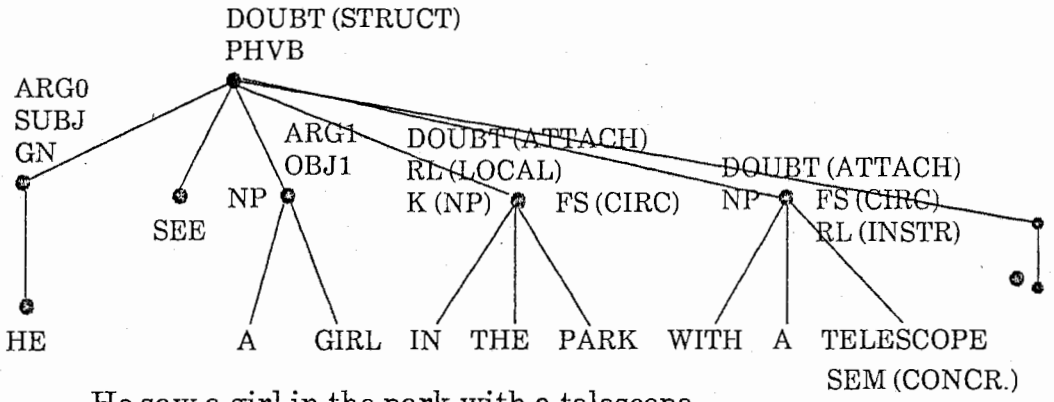
Ambiguity
Doubt



Which writer cites this speaker?

It is best to cook it in the oven.

RS (LOC)
DOUBT (RS) INSTR ?



He saw a girl in the park with a telescope.

For each type of problem, each application defines the default chosen, and the encoding of alternatives.

What about still unmentioned Feature-Structures (complex DAGs) ?

- 2 Extremes

① Simply labelled representation trees

② Complex F-Structures on binary derivation trees
- 1 → Clumsy handling of set operations or predicates
- 2 → Clumsy handling of sequences
  (Order, Repetitions, ...)

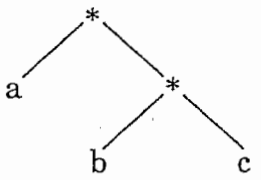
- Decorated Trees more "Balanced"
  - Each node has bounded-size Info
  - Natural encoding of set / sequence related Infos

- But Complex F-Structures more explicit
  (Actually, too much for intensive computation:
  Reentrancy impedes "Divide & Conquer")

A Simple Example

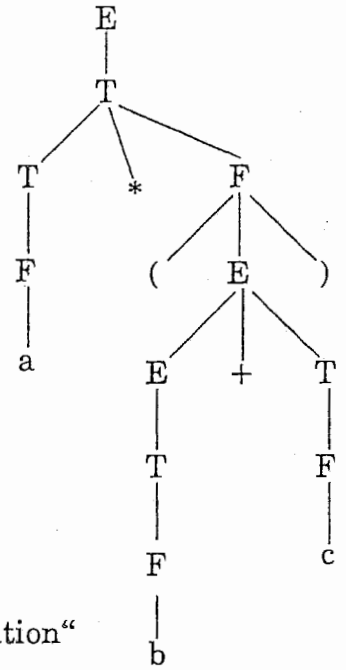
E → E + T / + (α1, α3)
E → T / α1
T → T * F / * (α1, α3)
T → F / α1
F → (E) / α2
F → id / id

5 Nodes



a * (b + c)

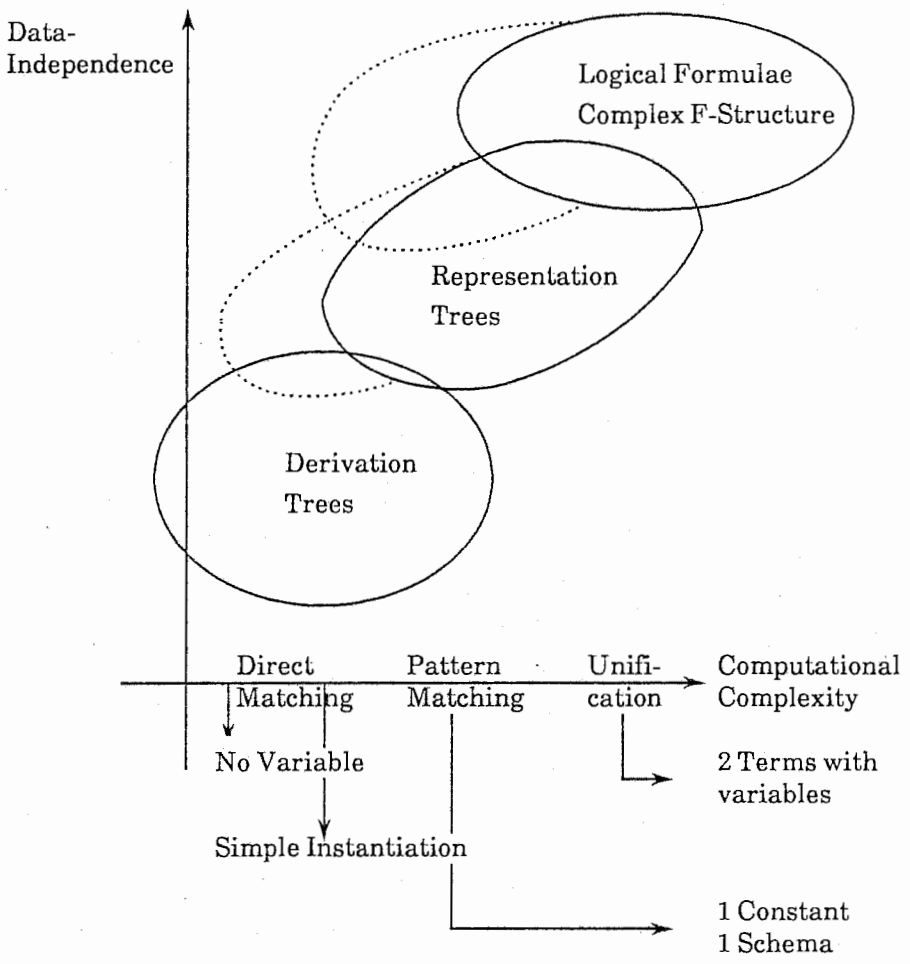
18 Nodes



Linguistic representations should be also be
- Economical
- Conspicuous

However, there are
2 more conflicting goals :
- Be Data-Bound
  "Don't loose anything"
  Here --- How many brackets ?
- Be Goal-Bound
  "Fit for Sematic Processing / Integration"

A "Rule of Thumb" for using this or that Structure



Of course, some simple Representation Trees / F-Structures may be obtained from Derivation Trees by local attached actions.

But interesting Representation Trees (e.g. $a^n b^n c^n$) usually can not be obtained without a more Complex Device.

A Sketch of the Beginning of a Tentative Approach .....
To Define a Convenient Interface Representation Structure in a MI System

Some Motivations

- The main problem of MI is to link speech & NL techniques.
- Symbolic & heuristic approaches have failed in speech processing, while stochastic methods, more "Brute-Force" are (now) doing better.

- The integration through a "Blackboard" seems a good idea, if programmed heuristics (Agenda, ...) are replaced by more parallel, weight-manipulating methods.

- The next problem of MI is basic speed.
  → Impossibility of using combinatorial approaches with heavy pattern-matching or unification.

Some Ideas

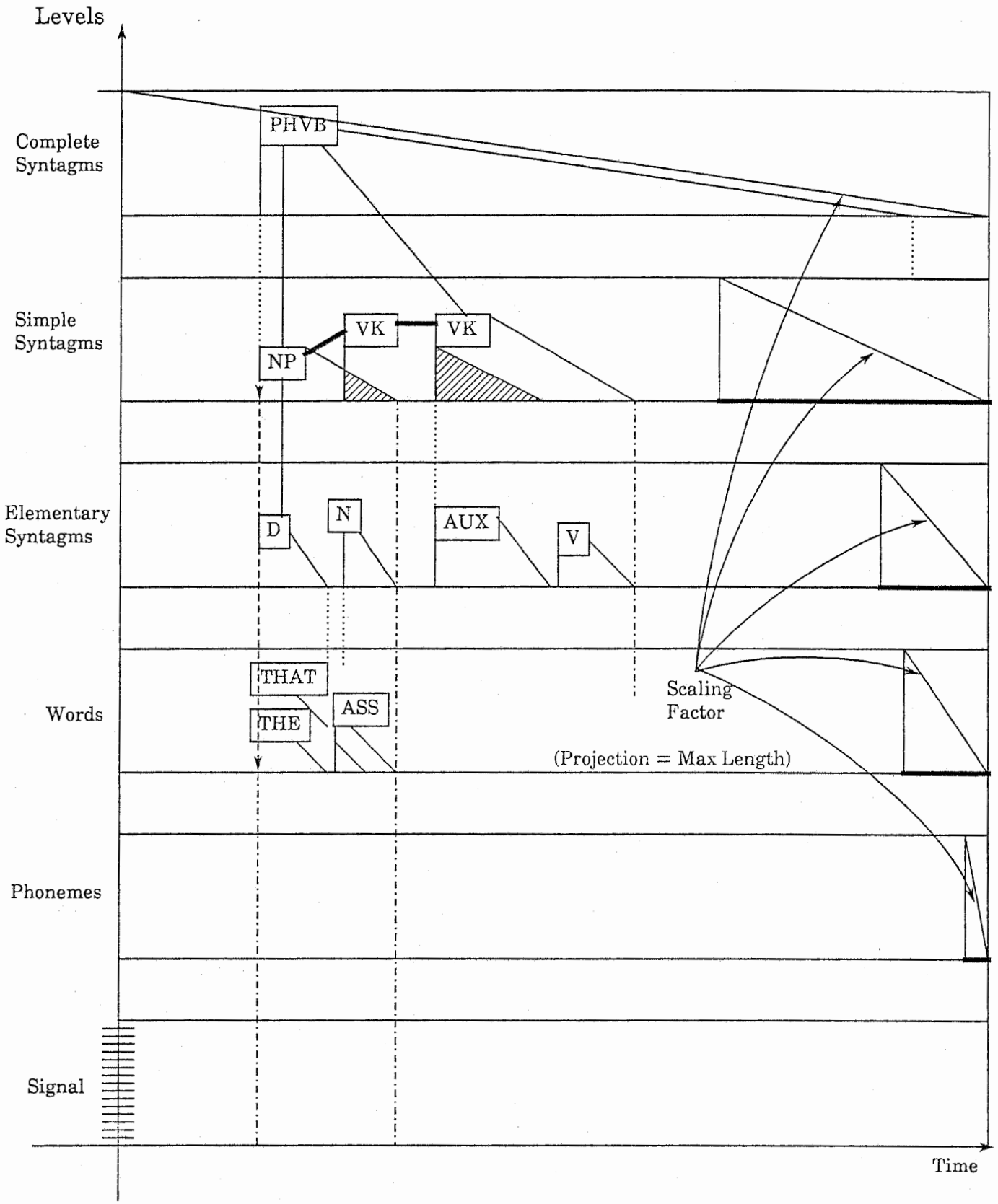
- Use a working (factorizing) structure all the way through analysis (Blackboard-Like)

- Work by levels, in the way of speech techniques.

- Construct simple structures first, and more elaborate ones on / from the simple ones which get high grades.

- Use vertical activation / lateral inhibition.

A Possible Macroscopic View



Content of a Cell

- Main Label & Connexions (?) may be built in the data structure.
  (Not the words, however)
- Simple Attribute Structure
  (Or direct link to microfeatures)
- Node of Representation Tree
- With a Complex Decoration
- With the correspondences to input
  (SNODE, STREE)
- & ..... with Complex F-Structure, if needed & computable in real time.

Specification and Application of Grammars and Dictionaries

Outline of Presentations

Grammars & Dictionaries

Static / Dynamic

General / Specific (to MT) but Neutral

/ Specialized (System, Phase)

& their writers

Main Computational Methods in NU

Combinatorial / Heuristic

Pure / Impure

How to reconcile the 2 main trends ?

- Impossibility of deriving automatically an efficient program from a functional or relational specification.
- Techniques from COMP SC : Invariants
- Techniques from AI : Specific Heuristics

A Suggestion for MI Systems

From Data-Bound to Goal-Bound

&

From Combinatorial to Heuristic

Grammars & Dictionaries

- Static / Dynamic

  Declarative + Procedural

- General / .....
  - Typically hand-made, dictionaries rarely computerized, grammars never
  - Not usable directly for MT or MI

- But indispensable resources

..... / Specific to MT but neutral

- Ex : MU-Project, CALLIOPE, LOGOS
  LEXDB + Automatic Generation of "Running" Dict.

- For Grammars : GETA's Static Grammars
  All other MT Projects have only Dynamic Grammars.

..... / Specialized (System, Phase)

- Exact format required
  eg. MORPH. ANAL. : Access by MORPH, ...

- Use of predefined set of codes
  (Templates, ...)

Necessity to Separate Grammars & Diecionaries

- Not the same writers, different competences & creation techniques

- Dictionary / Domain
  Grammar / Typology

An MT or MI System is <u>always</u> specialized.

..... Success of Expert Systems,
Failure of General Systems .....

- Computational Side (Time) :

- Dictionary-related phase

n : length

◯ (ng) ◯ (ng log d)

g : grammar
d : dictionary

- Grammar-related phase

$G = g + d$

◯ (nG) ◯ ($n^{\{2,3\}}G^2$) ◯ ($n^6 2^G$) ◯ ($n^3 3^{GG}$)

RARE ! EARLEY ..... HGPSG GPSG .....

Contrary to beliefs of theorists, a lexical approach does not guarantee small grammars.

Also

Actual dictionaries proposed by XYZGs proponents are

- Very small
- Very poor (quite poorer than usual MT dictionaries)

Actual grammars proposed by same are also

- Very small
- Too "clean" (no interest for usual phenomena)

  See Tsujii's, Tomita's Comments.

- It is quite possible to get a grammar larger than 200 p (10000 lines).

Main Computational Methods

Combinatorial Methods

- Pure : Produce all solutions
  CF-Grammars in the sixties
  AUGM. CF-G in the sixty-fives
  (& now - METAL, ...)
  U-Grammars in the eighties

  ("Packing" is the new word for "Factorizing", used-of course-since 1960)

- Impure : Deliver 1 or some solutions
  - Filters old idea
    Mel'tchuk & Kulagina (1956)
    PIAF (1970)
    ..... All "Sequential" approaches
  - Preferences ~ 1967-73
    CETA System (For Syntax)
    WILKS (Preference Semantics)
    GETA's Systems (Integrated in AS)
  - Weights
    METAL, Critique

All solutions are computed first, then some choice is effected.

Heuristic Methods

Try not to compute all possible solutions

- Pure : Search Strategies
  (Without storing of partial results)
  Backtrack Methods (PROLOG, ATN, ROBRA)
  Idea : The first found is the best.

- Impure : Associated to combinatorial base
  ATEF (ND-Finite State + Heuristic "Functions")
  Agenda + Blackboard idea
  (Not yet used in MT, but ..... in MI ?)

How to reconcile the 2 Main Trends ?

- In MT, the 2 types of techniques have achieved success.

- In speech processing (recognition), the Data-Bound, Combinatorial Methods (using weights & not preferences) seem to WIN, the "More Intelligent" to fail.

- Quite the contrapy when it comes to "Deep" understanding, or knowledge-based processes.

- To this day, it is impossible to derive automatically an efficient program from a functional (relational) specification

  - No practical results of program synthesis research

  - No actual use of applicative programming
    ..... Also, incredibly difficult to debug even toy examples

  - Theoretical limitations, eg:
    $I\Psi = \{x \mid \psi_x \mid \Psi\}$ not recursively enumerable

  → Computational linguists or linguistic-oriented computer scientists are not likely to discover this GRAAL

→ Some procedural component (performance related ?) has to be ..... programmed. Just how ?

Techniques from Classical Computer Science

- Use the (static) grammars as specification
- Construct (EX NIHILO) some program (in ROBRA, GRADE, ANT, ...)
- Enrich the program with
  - Normal comments explaining the relation to the specification
  - Special comments, of formal nature, the invariants, at well-chosen points
- Enrich the programming language (SLLP) with run-time procedures for checking the truth value of the invariants (always, periodically .....) and take appropriate action

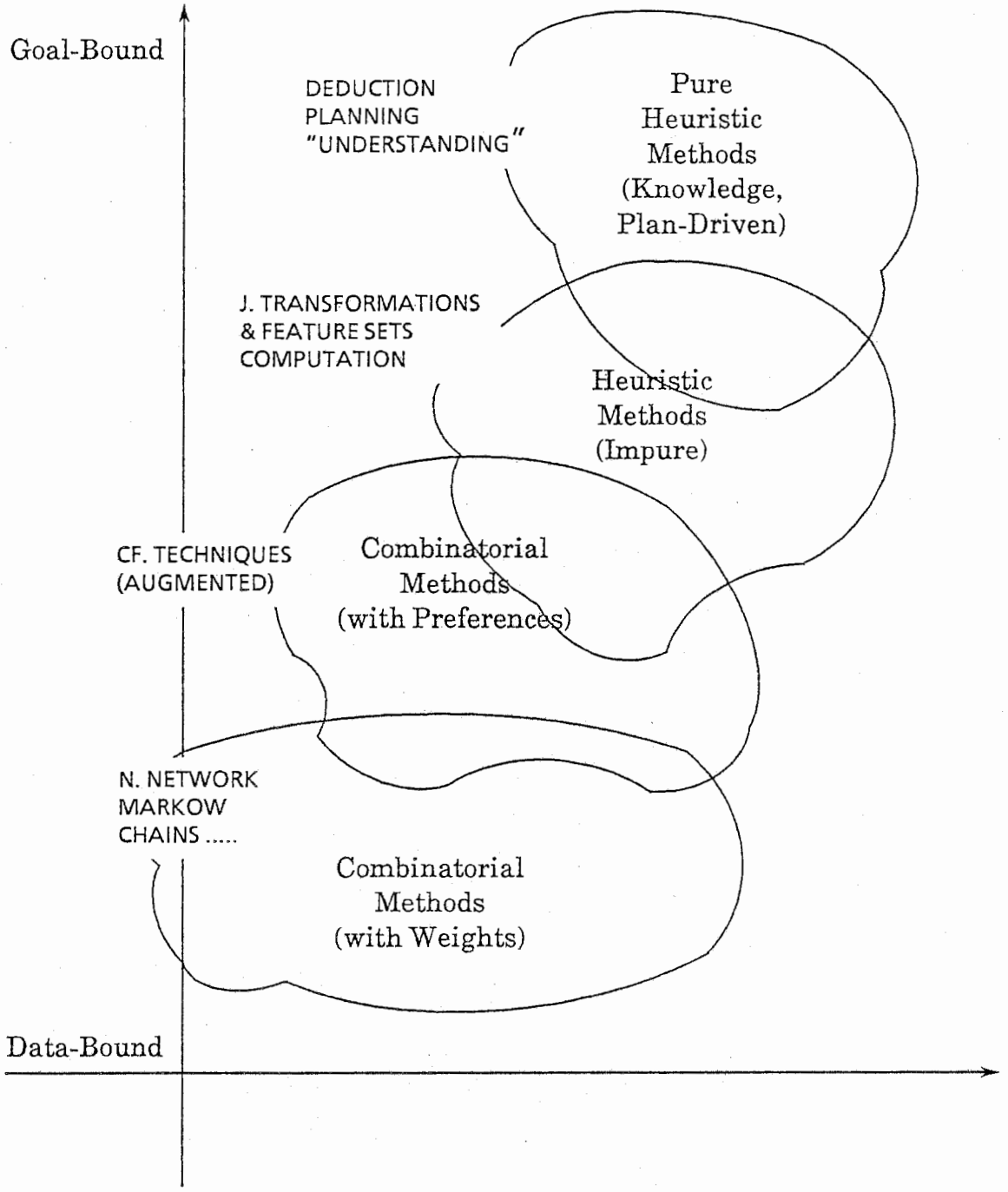
Techniques from AI

- Declarative Component
- Meta-rule Component

  How to solve local problems, or heuristics
  - Preferences, weights, interaction with user or knowledge base
- General Strategy

  At the highest level, global handling of non-determinism
  - Depth / Breadth / Best First
  - Development of n solutions in parallel
  - Freezing / Activating .....

A Suggestion for MI



Specialized Languages for Linguistic Programming

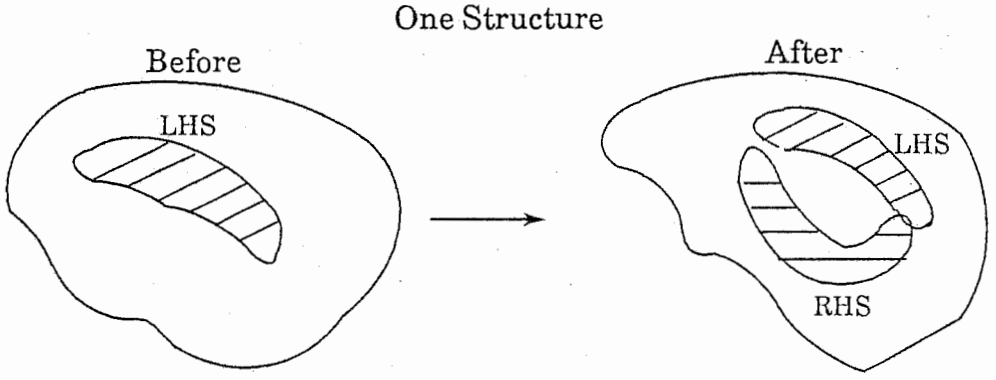
Outline of Presentations

1. 3 Possible Criteria for Classifying SLLPs
   - Types of Production Systems
   - Rule Mechanisms (Direct, P-M, Unification)
   - Control & Modular Organization
2. SLLPs of the Direct Type
3. Pattern-Matching Based SLLPs
4. Unification-Based SLLPs
5. Discussion on the place of these 3 types of SLLPs in machine interpretation.

3 Possible Criteria for Classifying SLLPs

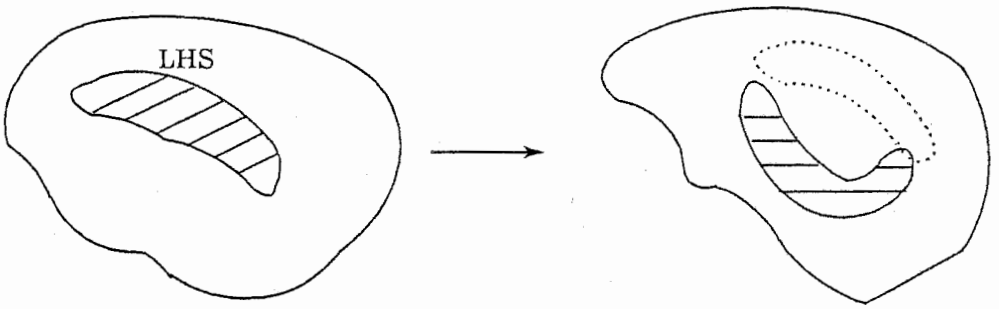
- **Types of Production Systems**

Addition - Usually followed by cleaning-up



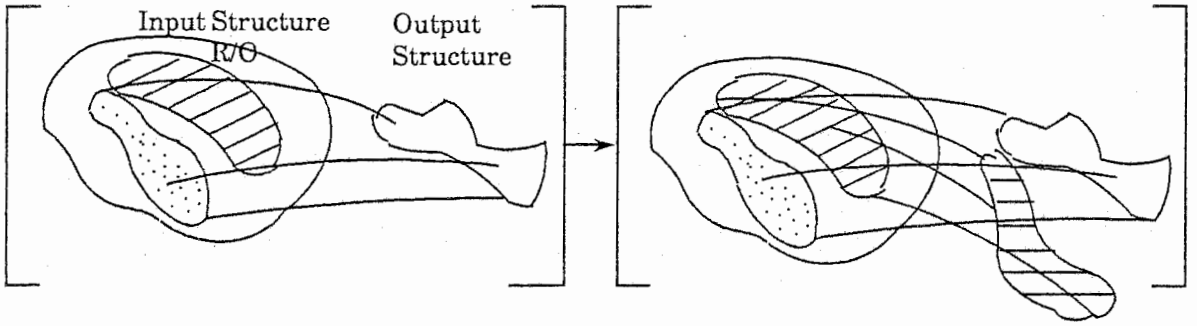
ex : Q-Systems, Chart Parsers

Substitution



ex: Editors (Substitute / <s1> / <s2> /)
Transformational Systems
TRANSFO, ROBRA, GRADE

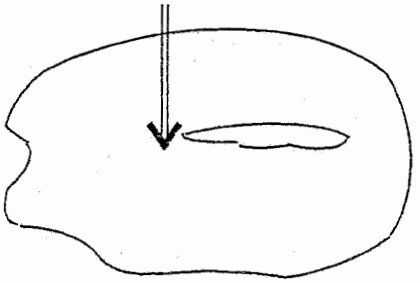
Transduction



- **Rule Mechanisms**

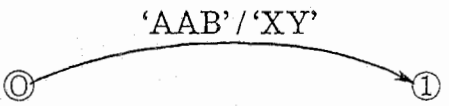
Direct: No global search in the working structure(s)

① Anchor



② Main element in the left-hand side is constant.

ex: Finite-state transducer (ATEF)

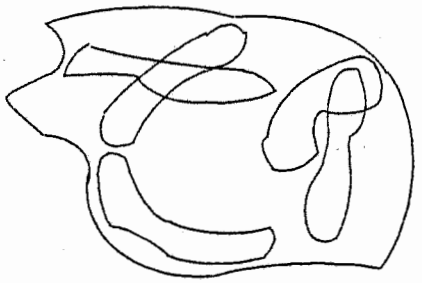


If variables appear, they are merely instantiaded

also : Context-Free Parsers, ATNs

Pattern-Matching : LHS with variables global search

But structure has no variables

¿  ?

ex : Q-Systems
ROBRA, GRADE

Unification : Variables on both side

ex : Logic Grammars (MG, DCG, XG / LMT .....)
"Unification-Based" (F-Structure Based) Grammars

- **Control & Modular Organization**

  - Subgrammars

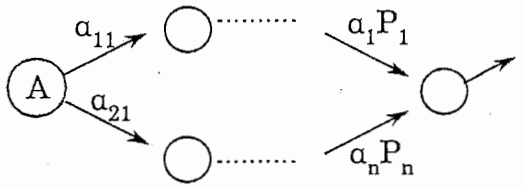
    - Programmed | Grammars cf. Salomaa — r : $A \rightarrow \alpha /G1 / G2$

      Matrix — $[r^1, r^3, r^{10}]$

    - Q-Systems — Simple block of rules

    - ATNs, REZO — In reality, no subgrammars a (sub)network corresponds to all rules with same L.H.S.

$A \rightarrow \alpha_1 \mid \alpha_2 \mid ..... \mid \alpha_n$



    - ATEF, ROBRA, GRADE — Real subgrammar organisation

- Control Graph / Control Language

ROBRA, GRADE — on Strings in NLP — cf. Salomaa Validations / Saturations (CETA)

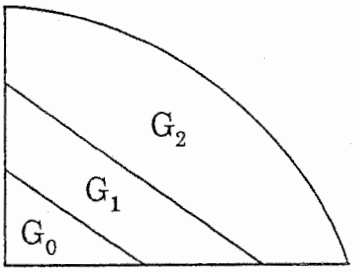
Q-Systems :

Sequence (MTL)

Trial & Error (Prague)

- Levels of Rules

METAL (with CFG)



For Machine Interpretation,

The Main Criterion is Time Complexity
(~ Real-Time Constraint)

Hence Study by the Rule-Mechanism Classification

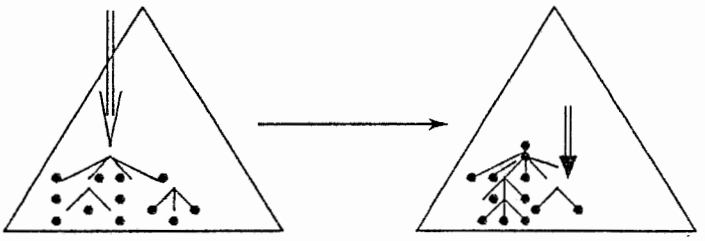
SLLIPs of the Direct Type
(No global search for a pattern, essentially data-driven)

- Regular (L-R) String Transducers, even if non deterministic.
  ex. ATEF (ND) Morphological Analysis
  SYGMOR (D) ——— Generation

- All CF-based formalisms, equipped with reasonable algorithms, if the computation does not entail matching / unification on unbounded structures

METAL, PNLP (IBM YTH)
ATN, REZO

- Tree-Transformational Systems, if in "Anchored" mode
  ex. TTEDIT (CETA)

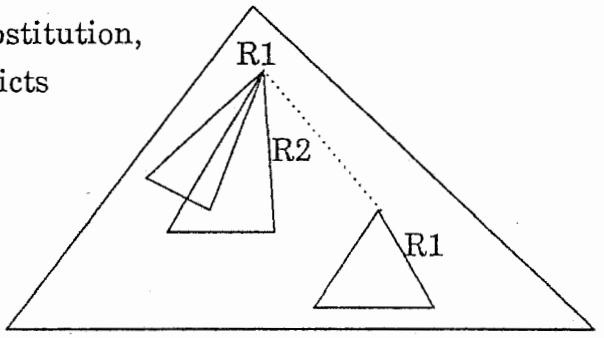


Pattern-Matching Based SLLPs

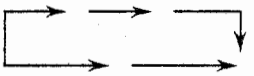
Global search for a pattern over an entire "Object Structure"

- Tree-Transformational Systems
  ROBRA, GRADE, TELESI (Chanché)

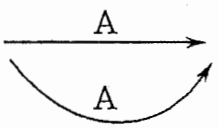
  → Because they use substitution, they must solve conflicts



- Q-Systems
  Not like chart parsers because :
  - May create new nodes



  - Don't test for equality



These systems are extremely powerful, & usable not only in analysis, but in transfer & generation

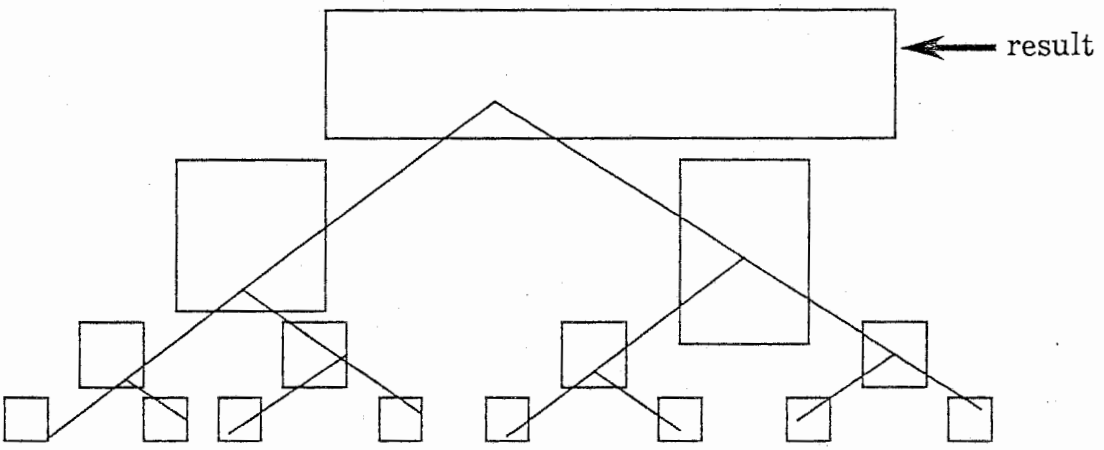
Cost : ~ 50 times cost of direct SLLPs
(If programmed comparably and if linguistic data comparable)

Unification-Based SLLP's

All those where unification of potentially unbounded structures is the main operation.

- CF-Based

  ex. D-PATR, as used in ATR



  ex. DCG, depending on how they are used !

  Not GPSG,
  - Where the F-structures are bounded, with no reentrancy
  - & where the result is really a complete attributed tree.

  & apparently also not JPSG (Gunji)

- MG (Metamorphosis Grammars based on CSG)

- STCG (or "Static Grammars") - Zaharin

  Here, identification rather than unification

  (Variable on 1 side may not be substituted)

Cost of U-Based SLLPs ?

- In the previous sense:

  Not yet any actual measurement, on complete systems

  As opposed to direct & PM-based

  Preliminary experiments → Staggering cost

Theoretical complexity
(Book by Barton, Berwick & Ristad)
NP-complete or more
But, in practice,

Although some formalisms claim to be theories, thus exposing themselves to this reproach,
All reduce to tools, hence SLLPs
→ eg. D-PATR from many U-formalisms

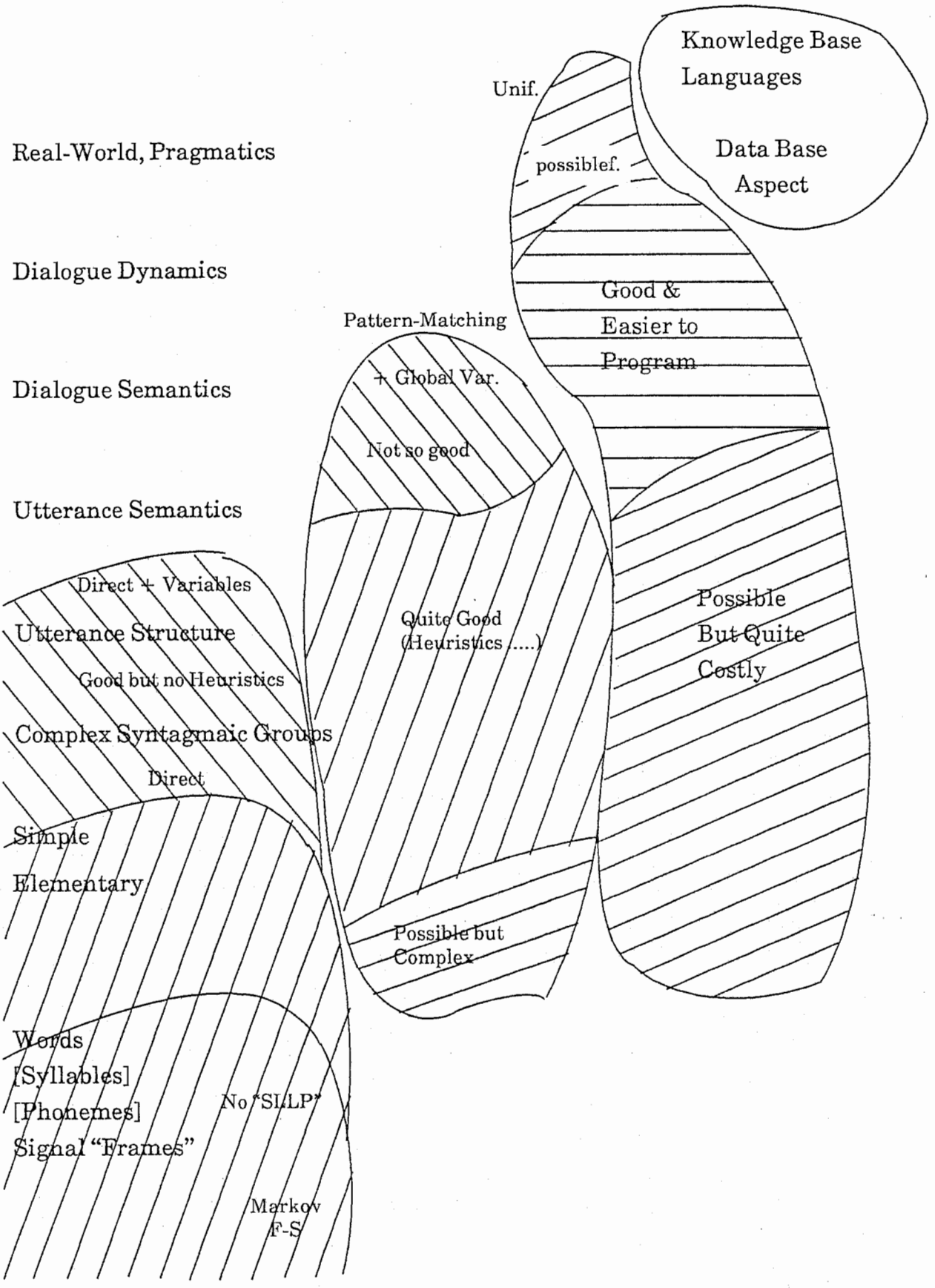
The real problems are :
- The cost of the used unification (Real / Pseudo)
- The added cost if types and other operations are allowed, which is necessary for a SLLP to be useful (numbers, strings, ... intersection, difference .....)

Discussion on the place of these 3 types of SLLP in MI

- Recall the considered levels & scope

Real-world, pragmatics	Whole situation, not only one dialogue	∞
Dialogue dynamics (in planning / understanding)	One dialogue	<2000
Dialogue semantics (speech acts, participants)	Dialogue "Window" or total dialogue	<100 ?
Utterance semantics	Utterance + Window	
Utterance (abstract) structure (complex construction)	Utterance (1-n Sentences)	<200
Complex syntagmatic groups	Sentence at most	<80 ?
Simple syntagmatic groups	No recursive embedding	<15 ?

Elementary syntagmatic groups	Kernels eg. John Smith Finite State	<5 ?
Words [Syllables] [Pphonemes]	1w = 8f = 400ms	
Signal "Frames"	1f = 50ms	


Knowledge Base Languages
Data Base Aspect
Unif.
possiblef.
Real-World, Pragmatics
Dialogue Dynamics
Dialogue Semantics
Utterance Semantics
Pattern-Matching
+ Global Var.
Not so good
Good &
Easier to
Program
Direct + Variables
Utterance Structure
Good but no Heuristics
Complex Syntagmaic Groups
Direct
Simple
Elementary
Quite Good
(Heuristics .....)
Possible
But Quite
Costly
Possible but
Complex
Words
[Syllables]
[Phonemes]
Signal "Frames"
No "SLLP"
Markov
F-S

How could one "Mix" these different types ?

Idea : Unify the SYNTAX (of the SLLPs)
Equip with several "Rule Engines"
Unify the Data Structures

Data Structures

There should be

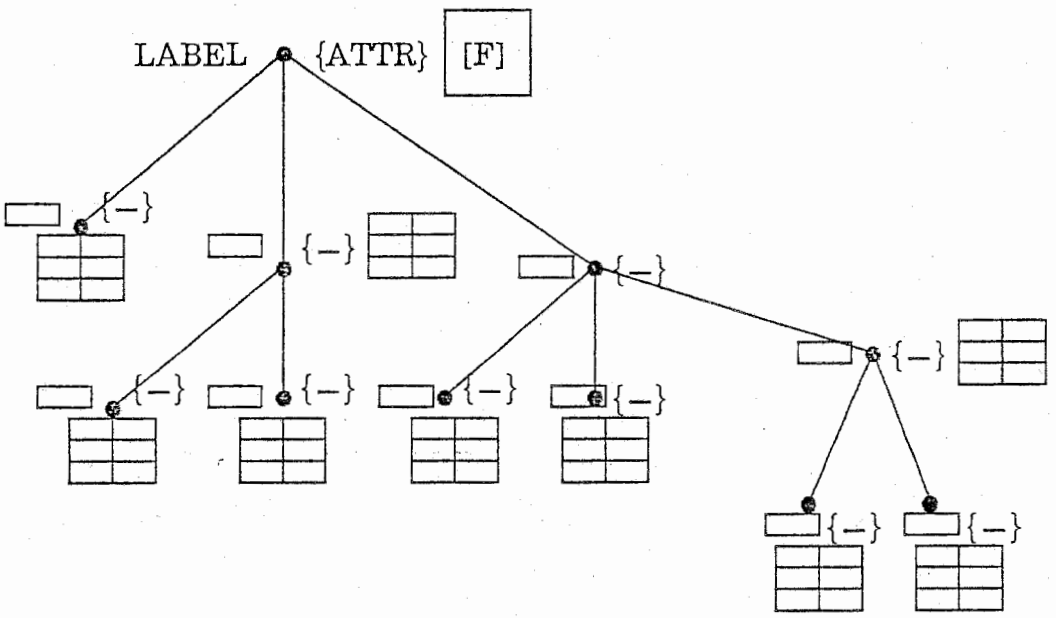
Nodes with Label, {Attributes}, [F-Structure]

Trees (ordered)

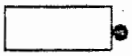
(Loopfree) Graphs

Charts ?
Q-Graphs ?
Lattices ?

On the Representation Trees



1) First construct only

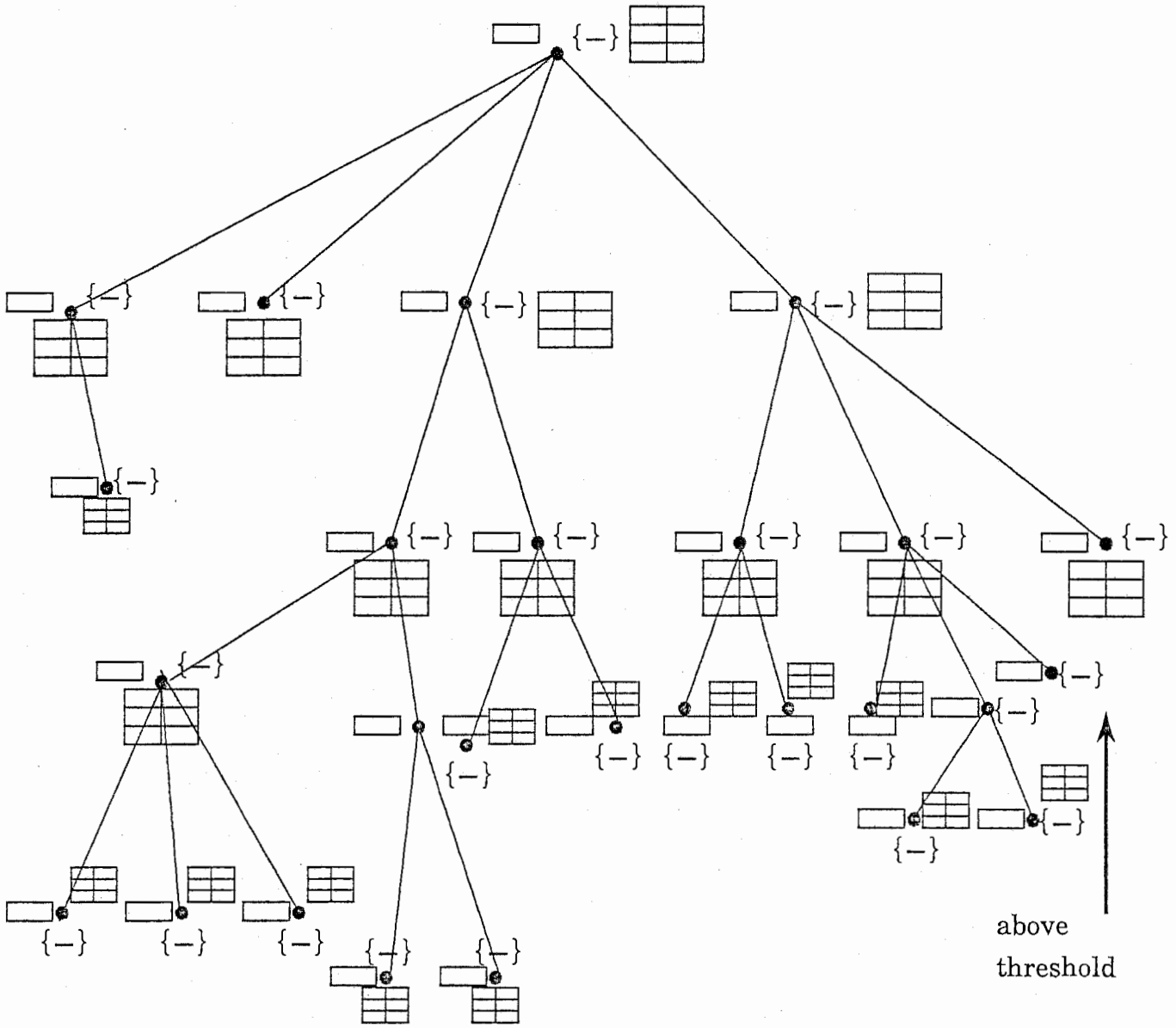


Skeleton

eg. by attached transformations

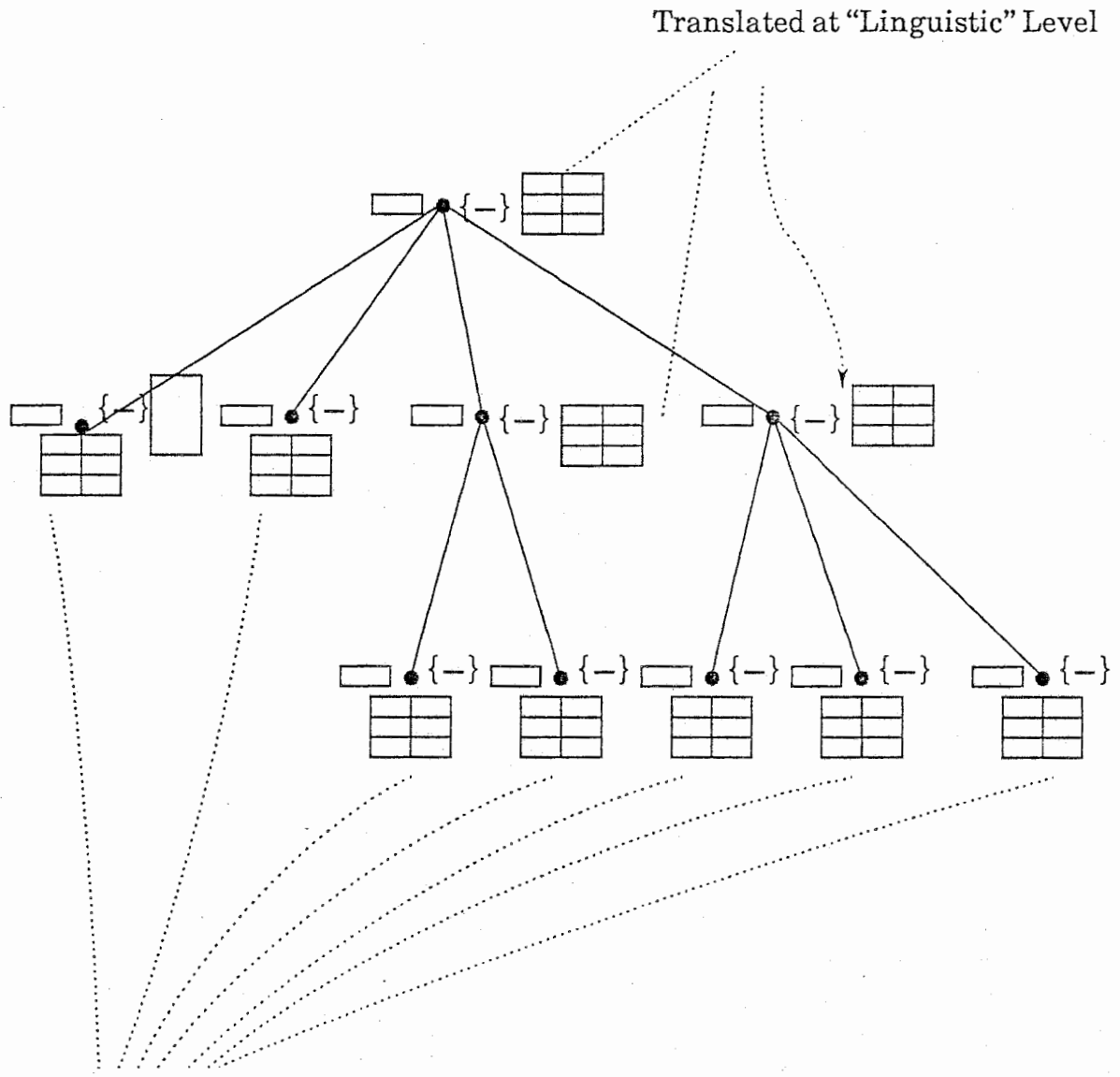
2) Then make the selected representation(s) more explicit by computing the F-STRUCT

3) Suppose weights have been used in the construction.
We get a natural way to translate the "Important" parts (of TSUJII) at a higher level than the rest.



Result of 1 + 2
3 : PRVNE

Note : "Important" does not mean "domain-related" any more, just dependent on ..... How good we judge our analysis.

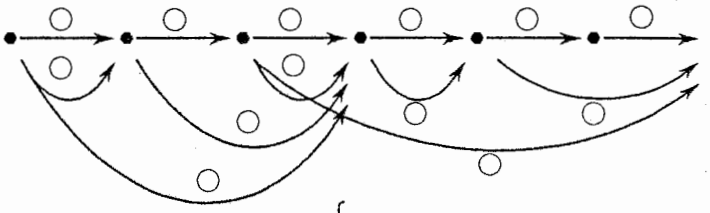


Note : Translation at linguistic level does not have to be as bad (word for word) as in TSUJII's sketch !

Lattices rather than Q-Graph or Charts ?

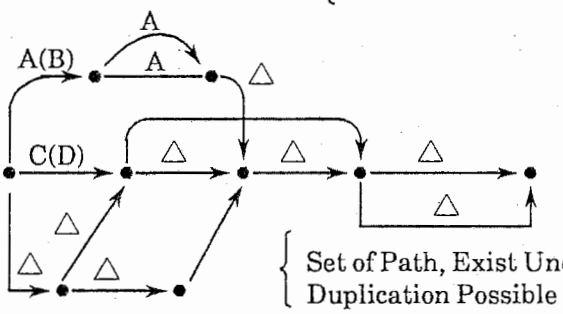
RECALL

CHART



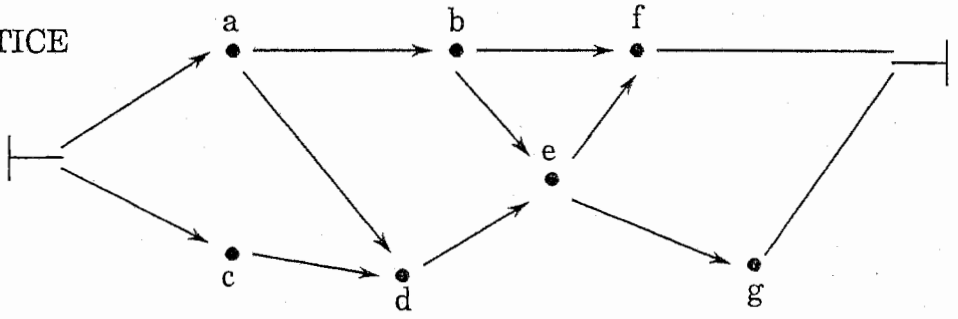
1 Sequence of Nodes
No Duplication (usually)

Q-GRAPH



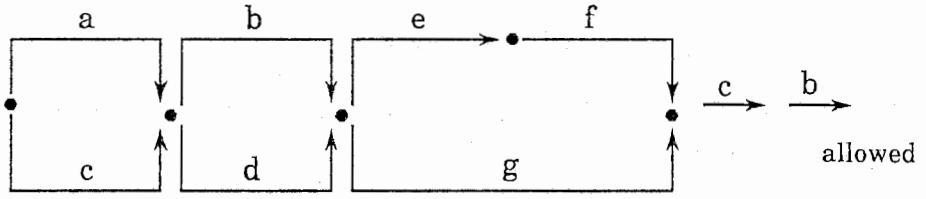
Set of Path, Exist Uncomparable Nodes
Duplication Possible

LATTICE



Lattices produce less "Parasites" when duplication is avoided

eg.



Lattices seem more used in speech processing.

Organization of the Whole MT / MI System

Outline of Presentations

I. The Importance of Size

- Quantum Leaps in Methods
- Size & Time Complexities
- Mockups, Prototypes, Operational Systems

II. Environments

- Preparation : Lingware Versions, Tests
  Test Corpuses
- Experimentation : Compiled Packages, Measures
  Validation Corpuses
- Debugging : Variety of Levels, Tools
- Production : End-User Interface
  Translation Request, Revision .....

III. Some Useful Concepts for Modularity

- Lingware : Steps, Phases, Components, Versions, ...
- Texts : Corpuses, Transcriptions, Translations, ...

The Importance of Size

- Quantum Leaps in Methods

  What works on small, closed problems is usually not extendable.

  e.g. - Morphological analysis may be unnecessary if you work on 20 sentences.
  - Trigram methods may work for 1000 words (10^9 ELTS), unlikely for 100,000 (10^{15} ELTS).

  Usual steps in MT development

		Man × Year	Dict. Size
20 sentences	(1 page, 250 w.)	1-3 × 1	200 w.
200 sentences	(10 pages, 2500 w.)	3-4 × 3	2000 w.
2000 sentences	(100 pages, 25000 w.)	~10 × 3	20000 w.
20000 sentences	(1000 pages, 250000 w.)	~20 × 3	~100000 w.

  Problems in grammar & dictionary management may force to abandon methods successful until a certain point.

  e.g. - CF-based grammars begin to be unmanageable with a few hundred rules.
  - Direct dictionary coding OK for <10-15000 w, by then a lexical data base management becomes necessary.
  - Although example of complex transfer dictionary

- Size & Time Complexities

  SLLPSs should be concise enough

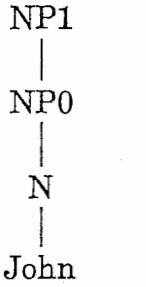
  Remember Charniak's description of "Paint" (half done !) on 9 pages of "Planner"

  The same goes for the descriptions of units of translation

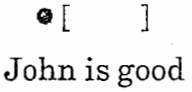
- For this reason, trees are preferred over graphs (note that any graph can be covered by trees). In any case locality is important.

- Linguistic trees should not have too many or too few nodes.

eg.



is QUITE BAD !



John is good

Also !

Compilation & execution time crucial
The best formalism is useless if it takes an hour to compile a small grammar and a day to translate 10 words (eg. EUROTRA).

- Mockups, Prototypes, Operational Systems

Mockup : Only the heart (1, 2, 3 SLLPSs)
No environment
Usually 1 language-pair & 1 version

Limit : Something like METEO during 1st year of development at TAUM.

Prototypes
Small
- Debugging environment
- DB management of lingware
- Almost nothing for the texts
- Nothing for communication / operationa

e.g. - D-PATR Only 1 text per grammar
Non sharable
(But D-PATR is far from a prototype environment for MT - no transfer, generation)
- LUTE ?

Large
- DBMS adequate for several versions, large dictionaries
- And lots of functions on texts
- But no integration in Documentation System Network

e.g. ARIAVE (-78, -85) cf. COLING-82, JACL 86/1
MU

Operational Systems

End-User interfaces
- Bilingual editor
- Command panels
- (Easy Dict. System)

e.g. SYSTRAN-Cee (+ TERMEX / MINITEL .....)
LOGOS

ATLAS (Fujitsu), PIVOT (NEC), HICAT (Hitachi),
AS-TRANSAC (Toshiba)

Environments

- The environment ..... of the environment
  - Dialogue language
  - Dialogue style (abbreviated, long, detailed)
  - Source language code
  - Target language code
  - Corpus name
  - Trace parameters for each phase

    ...

- Preparation
  - Editor settings for various types of
    - Linguistic data (free / fixed format, ...)
    - Texts (input transcription)
  - Automatic compilation (partial / total)
  - Lists (with all possible sorting orders)
  - Cross-references
  - Synthetic views across files

    (e.g. Produce all information for a set of lexical units)

    small tests

- Experimentation
  - Facility for producing compiled packages (memory images), quickly loaded / executed
  - Possibility to trace on one or several validation corpuses, ± selection
  - Storing of intermediate results to
    - Save future computation
    - Modify them (tree editor, frame editor) to produce normalized test inputs for subsequent phases.

- Debugging

  The facilities of the implementation language are not adequate

  → Various views of the unit of translation input/output of a phase

  → Various trace / step-by-step parameters

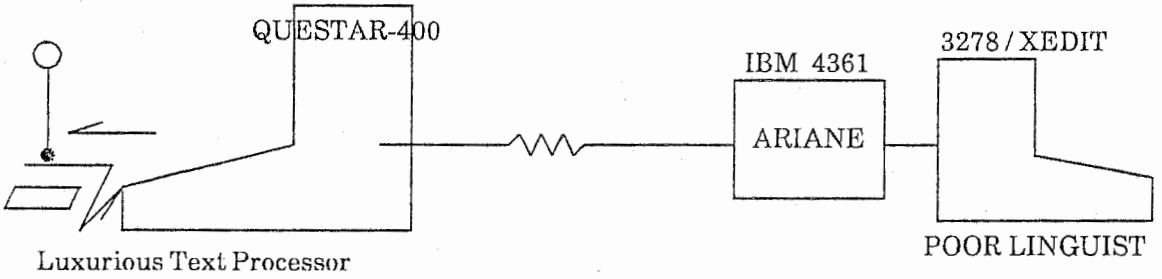
  → If the units of translation are large, it is also important to easily isolate the erroneous part, in order to deduce the size of the structures produced.

- Production

  Important to be able to start the translation of several brochures (say, 1 corpus per brochure & 1 text per section / abstract) in less than 1 minute, when rushing out at the end of the day .....

  → In a LAB, the production environment should offer the basic functions with which the operational system(s) will be built.

  e.g. ARIANE as background for CALLIOPE



Modularity, ma jolie .....
One (ARIANE) Space

- Texts
- Linguistic Applications
- Linguistic Modules (memory images)
- (Lisp) Programs → You may prefer C or PROLOG !
- Natural On-Line Dictionaries

- System Files (e.g. Internal ↔ External Names)
  (e.g. Values of environment variables)

- Lexical Data-Base, from which MT Dictionaries are constructed
  (e.g. MORPH. ANALYSIS, GENER)

- Transcriptors (in LT)
  e.g. 'Déjà là, DUPONT ?' → '*DE!1JA!2 LA!2, ** DUPONT'

- Personal files of individual user.

- Lingware

Step
- Analysis
- Transfer
- Generation

one could add
- Speech recognition
- Speech synthesis
- Understanding

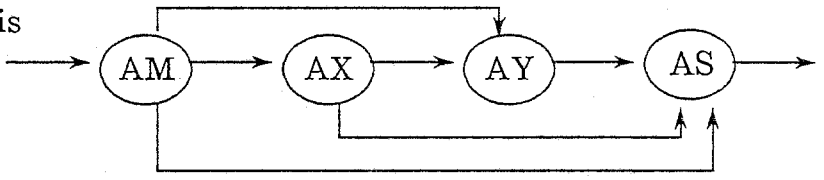
Phase — Autonomous part of a step, written in 1 SLLP
e.g. Morphological Analysis AM
Structural Analysis AS

Component — Element of a phase
e.g. Grammar G1, G2, ...
Dictionary D1, D2, ...

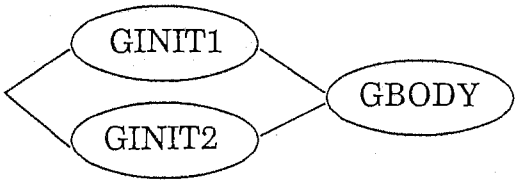
Selection of components gives possibly several realisations (G1 + D2)

Matrix — Graph of components. PATH = realisation or of phases

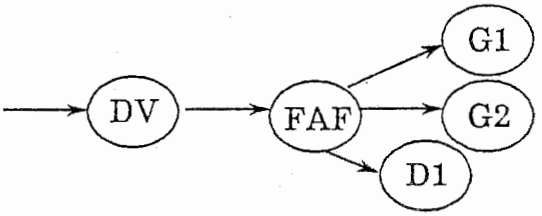
e.g. Analysis



Structural Analysis



Precedence Graphs
for Compilation of Components



- Textware

Corpus — Texts of same language, typology, domain, transcription (imagine ASCII/JIS)

Text — Not only 1 file, but complex file set :

Source Text

Brute — As inputted, in which format

Revised — Often, we have to preprocess slightly (spelling mistakes, OCR errors .....)

May be in more than 1 format,

e.g. ISO∅25 (French Keyboard) & SCRIPT
JIS & JTEX
CARIX & TETHYS (implementation 30%)

Descriptor — Gives logical structure (sections, paragraphs, sentences) used to segment in units of translation

Formats (HORS-TEXTE) — Figures, Formulas, ...

Brute / Revised

Translation — (Brute / Revised)
1 for each A+T+G realization

Intermediate results — 1 for each variant of each phase

Tools on Texts

List of Words Concordances	On 1 or several	Texts Corpuses
Unknown Words (± Context)	----------	

Merging / Splitting

Printing	± Translations ± Revisions	Problem of synchronization usually, by segments
Editing	----------	

Morphosyntactic Search

e.g. Look for Art Adj Adj Noun

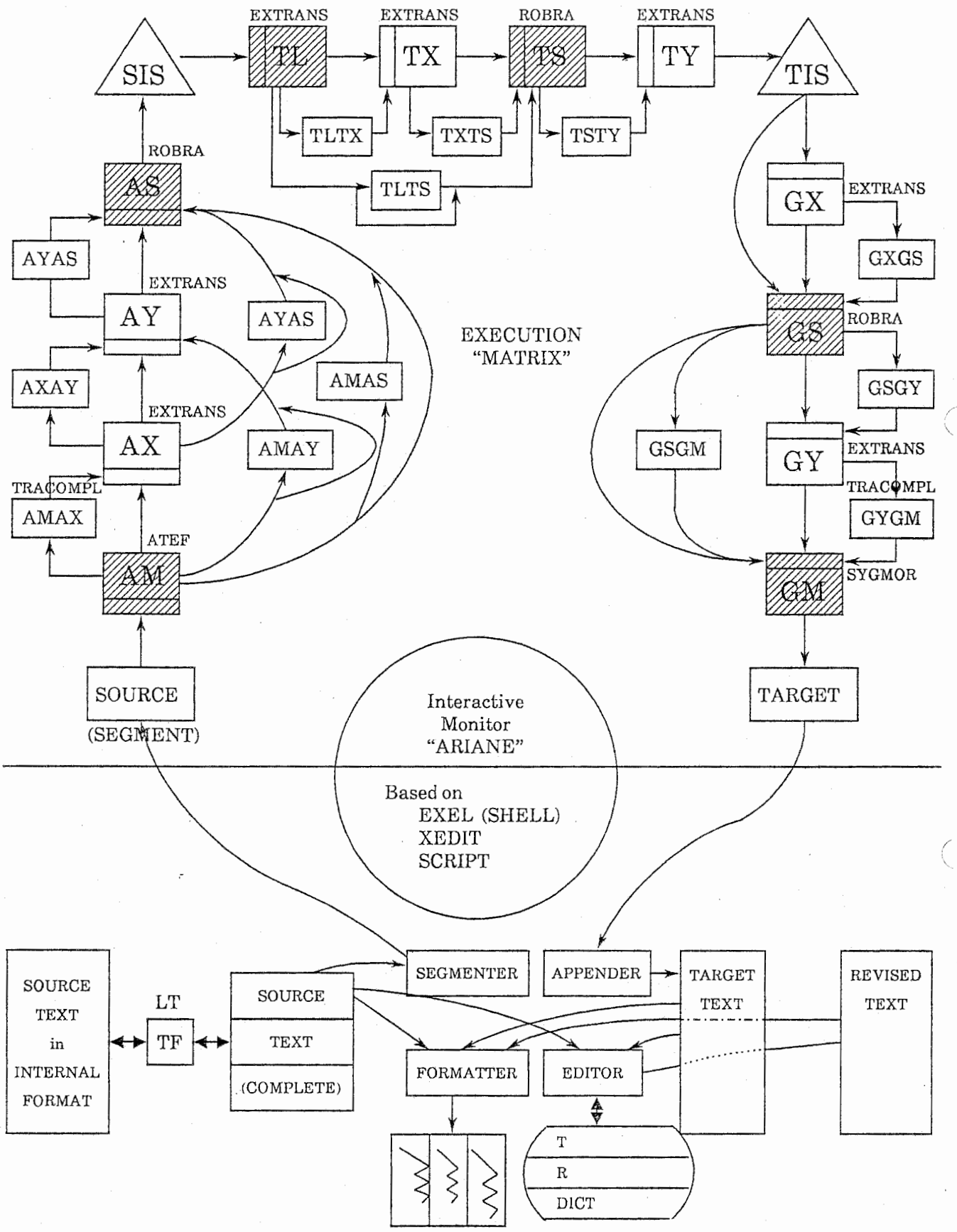
→ Stochastic techniques have been used (Spirit System) successfully

See also Lancaster group.

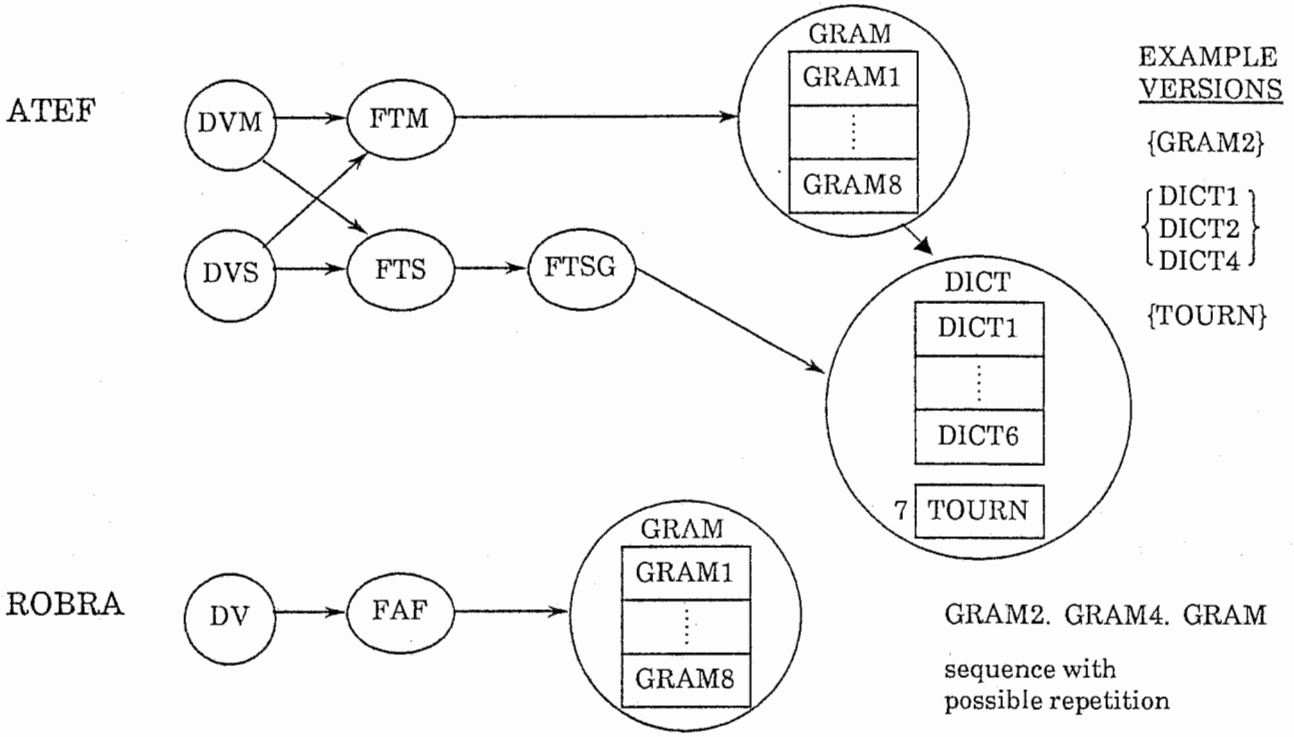
→ More powerful techniques for "Structural Search" should be reveloped.

See DEREDEC (Montreal) as first example.

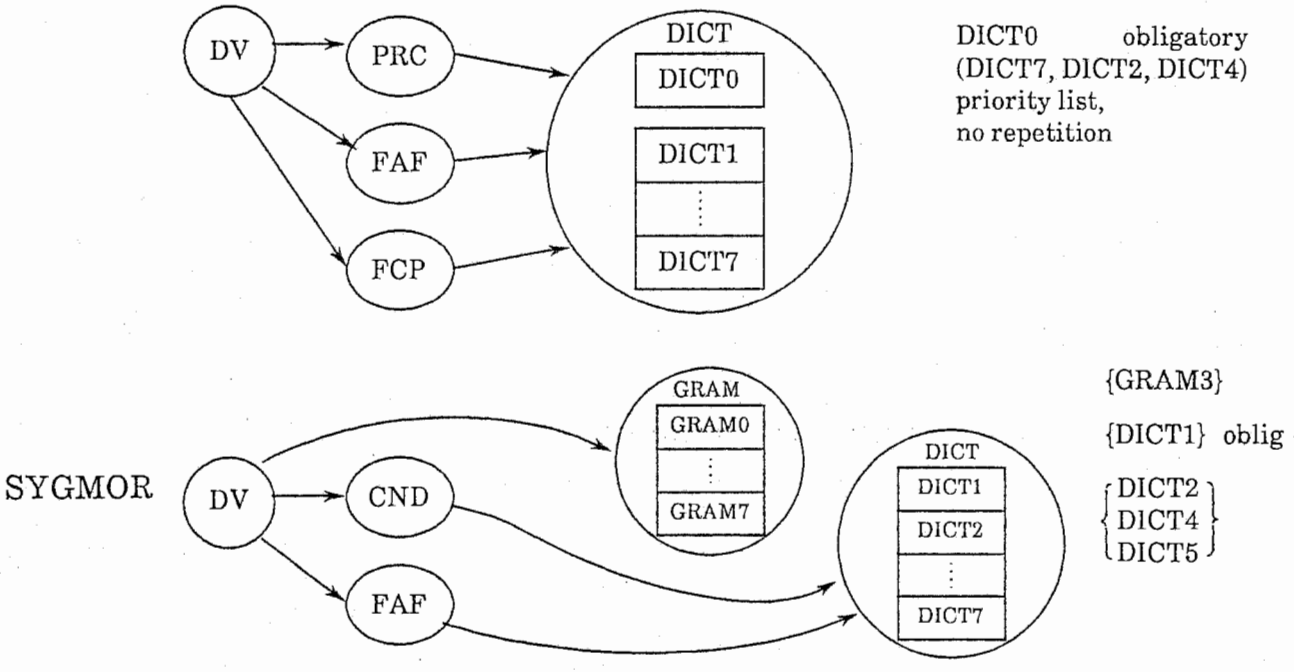
- The Example of ARIANE-85



- Component "Matrices"



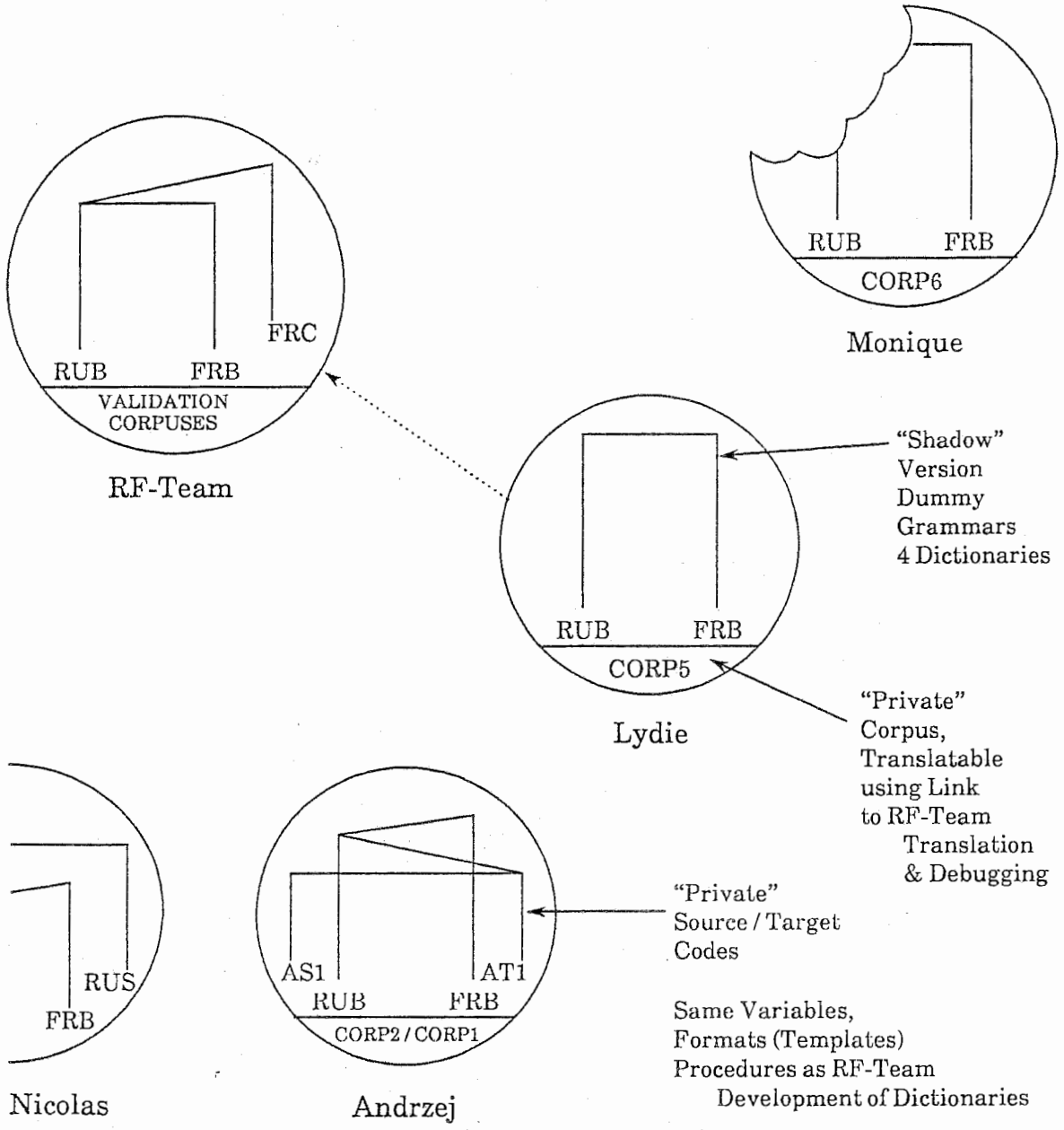
EXTRANS (for EXPANS / TRANSF)



TRACOMPL (for Transformation & Complement of DECORATIONS)

always 1 fik only

- Shared Development
  (From Russing-French Experience)



& dont't forget to Backup !!!

& Levels In user space (DUPLG / DESTRUL COMMANDS)
Use Backup ("Fixed") user space (TRGLGVM)
And tapes ..... (TRGLGVM COMMAND)

What else ?

Many things not in ARIANE, ideas could come from various sources :

- Integrate everything under the editor / window system, as in lute, instead of calling the editor on speicific files.

  Integrate indexing workbench, with
  - ± Lexical Data Base
  - \+ Indexing Aids
  - – Access to "Natural" Dictionaries (MINITEL .....)

  Reconsider the organization, to distribute 1 user space over a (physical or logical) network, with several users in parallel.

  For machine interpretation

  Add several forms of texts

  FRAMES
  PHONEME ┐
  SYLLABLE │ Lattice or Grid
  WORD (phon. ┘
  or orth.)

  Tools for stochastic classification
  Manual tagging of samples
  Construction of HHM ("Learning") etc.

Finally, the structure of a whole MT System is like that of a software factory.

- A useful partition of sub-environments is by users :

End-User	should come FIRST !
Linguist	Developers
Lexicographer	
+ Phonologist	
+ Cogniticist (!)	
Computer Specialist	Special functions, tools, ...

- Perhaps the execution can also be distributed on several machines (cf. HEARSAY), the environment should then take it into account from the start.