A Linguistic Look on Programming Languages

Isabelle Tellier

Université Paris 3 - Sorbonne Nouvelle

Origins, Motivations

Biographic elements

- I was educated in Computer Science
- I have been a professor of computer science
- I have programming experience/teaching in : Assembleur, Fortran, Pascal, Modula2, Ada, Lisp, Prolog (including constraints), C++, Delphi, Python, Javascript, PHP...
- My research interest is NLP (Natural Language Processing)
- I got interested in linguistics, read textbooks on language sciences
- I became a professor of (NLP oriented) linguistics

A question arose

What are the specificities of NL/Programming Languages ?

Outline

1. Some specificities of Natural Languages

2. How do they apply to Programming Languages ?

3. On the Nature of Programming Languages

1. Some specificities of NL

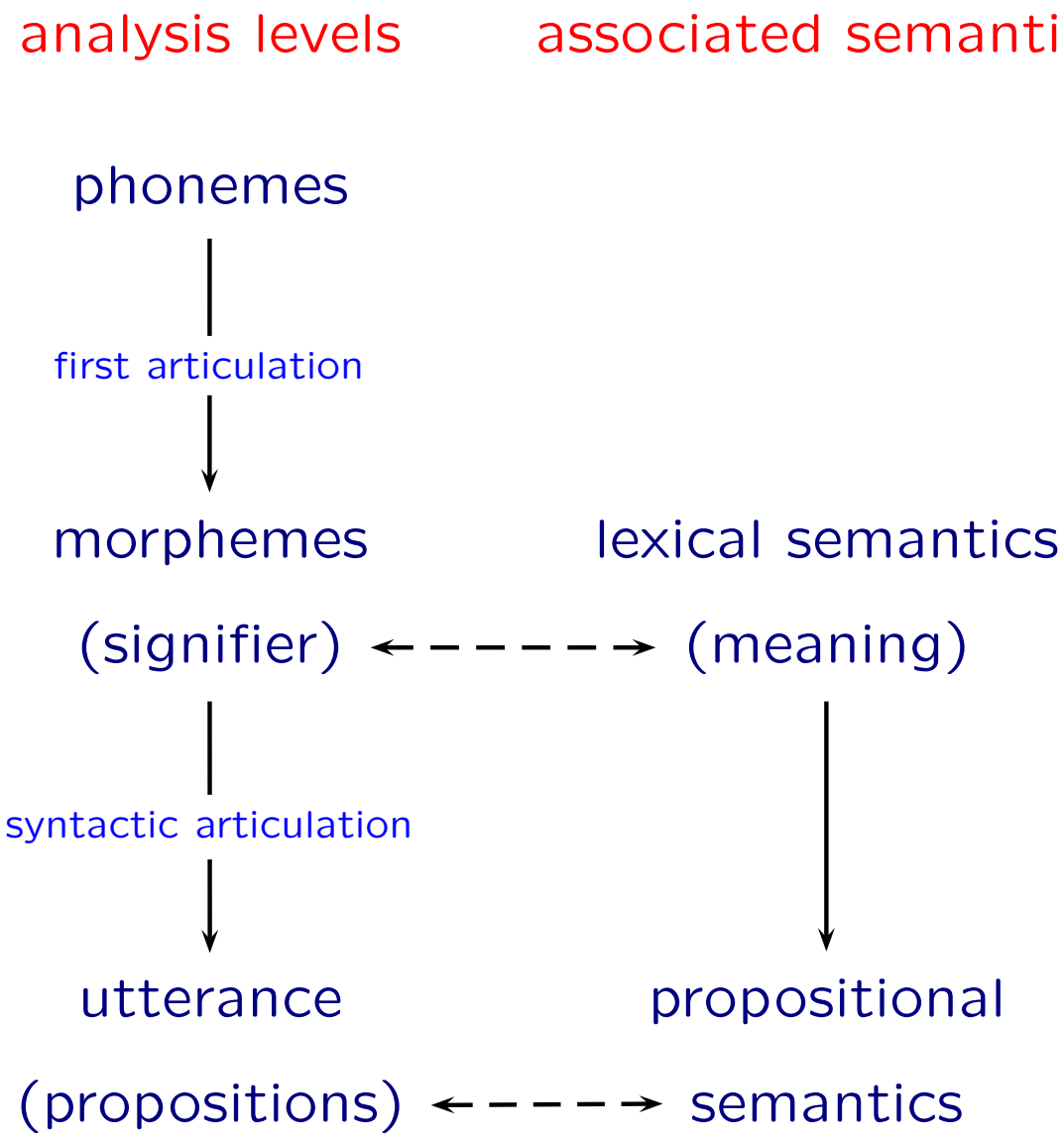
Basic features of NL

- universally developped in human societies
- oral, thus linear, channel, then transcribed into (also linear) writting
- evolve through history (diachrony)

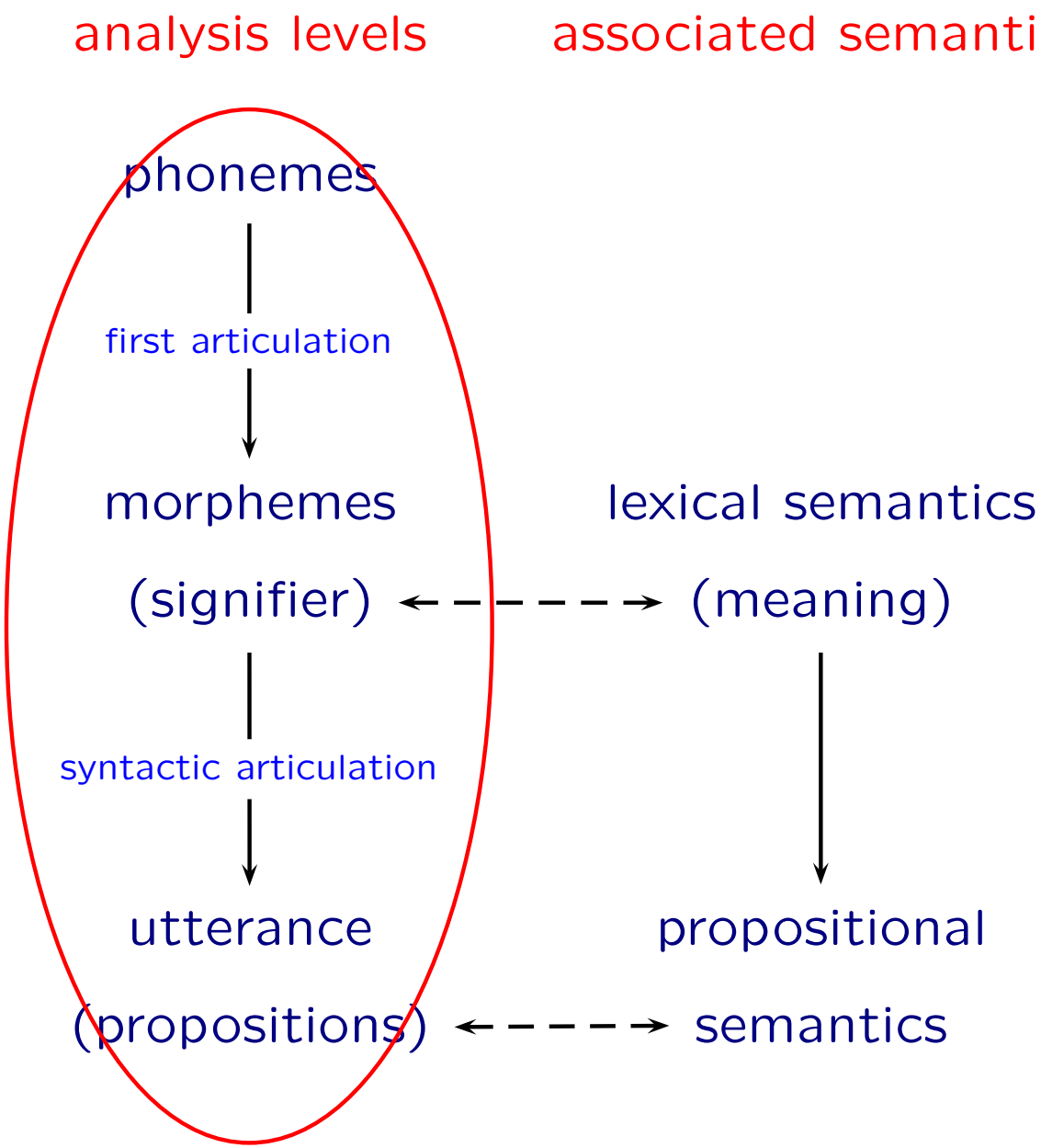
The "double articulation" characterization (Martinet)

- Natural languages are all composed of 3 levels of discrete units
  - phonemes : equivalence classes of meaningless distinctive sounds
  - morphemes : smallest meaningful units
  - propositions : syntactically combined morphemes expressing potentially true/false utterances
- connected by two levels of combination rules
- associated semantics are potentially ambiguous at each level

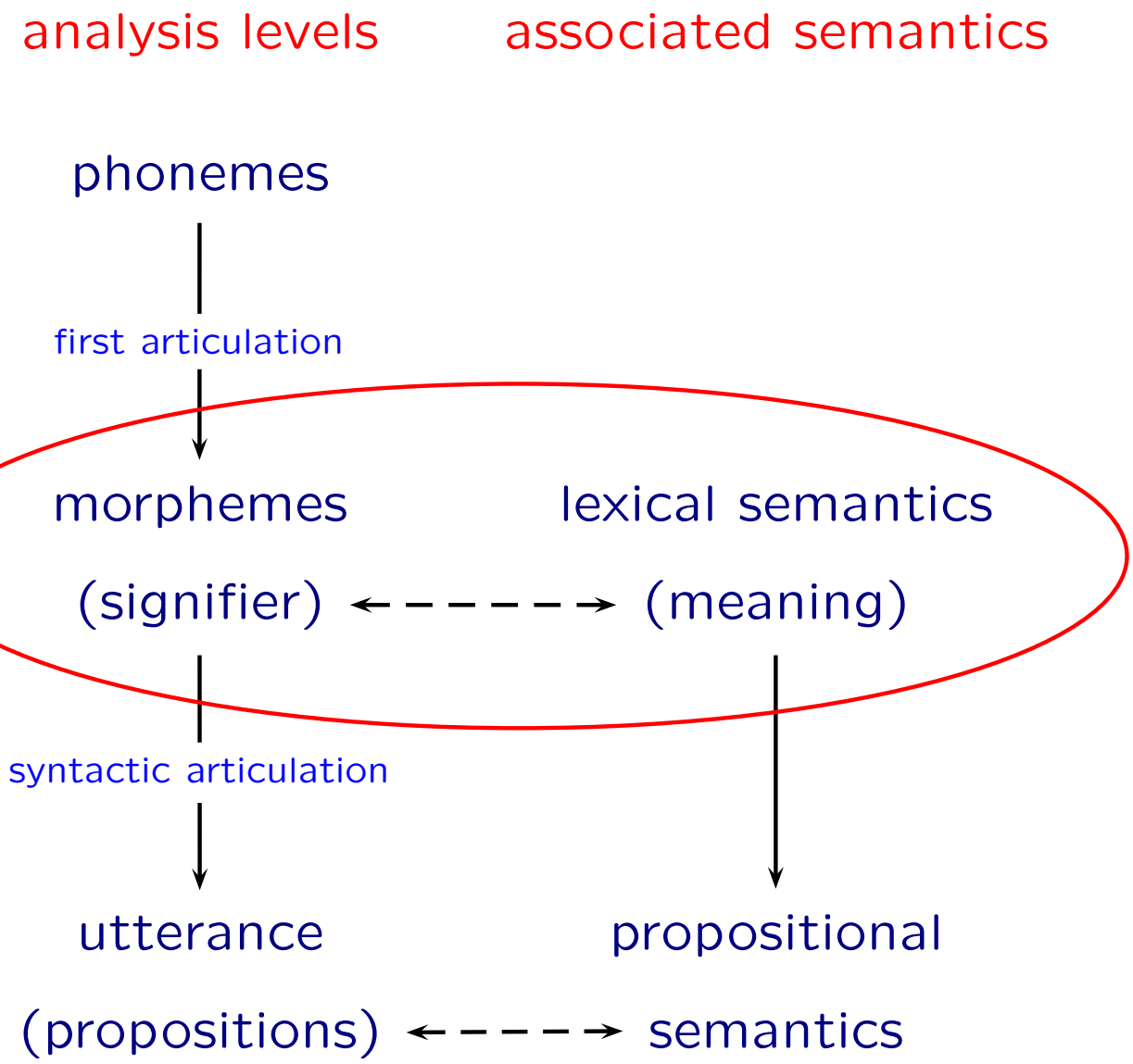
1. Some specificities of NL
Analysis levels



1. Some specificities of NL
Double Articulation



1. Some specificities of NL Protolanguages



1. Some specificities of NL

On the nature of lexical units

- lexical units can be partitioned into "part-of-speech" (POS) categories (Det, N, V, Adj,...)
- a category is an equivalence class for a substitutability property preserving the grammaticality of the utterance
- there are two kinds of lexical units
  - grammatical units (det, prep, conj, pronouns ?...) : belong to a finite list, useful for the grammatical structure more than to refer to the "external world"
  - lexical ones (N, V, Adj,...) : belong to an open list, refer more clearly to things, properties, actions... in the "external world"

1. Some specificities of NL

On the nature of syntax (Chomky)

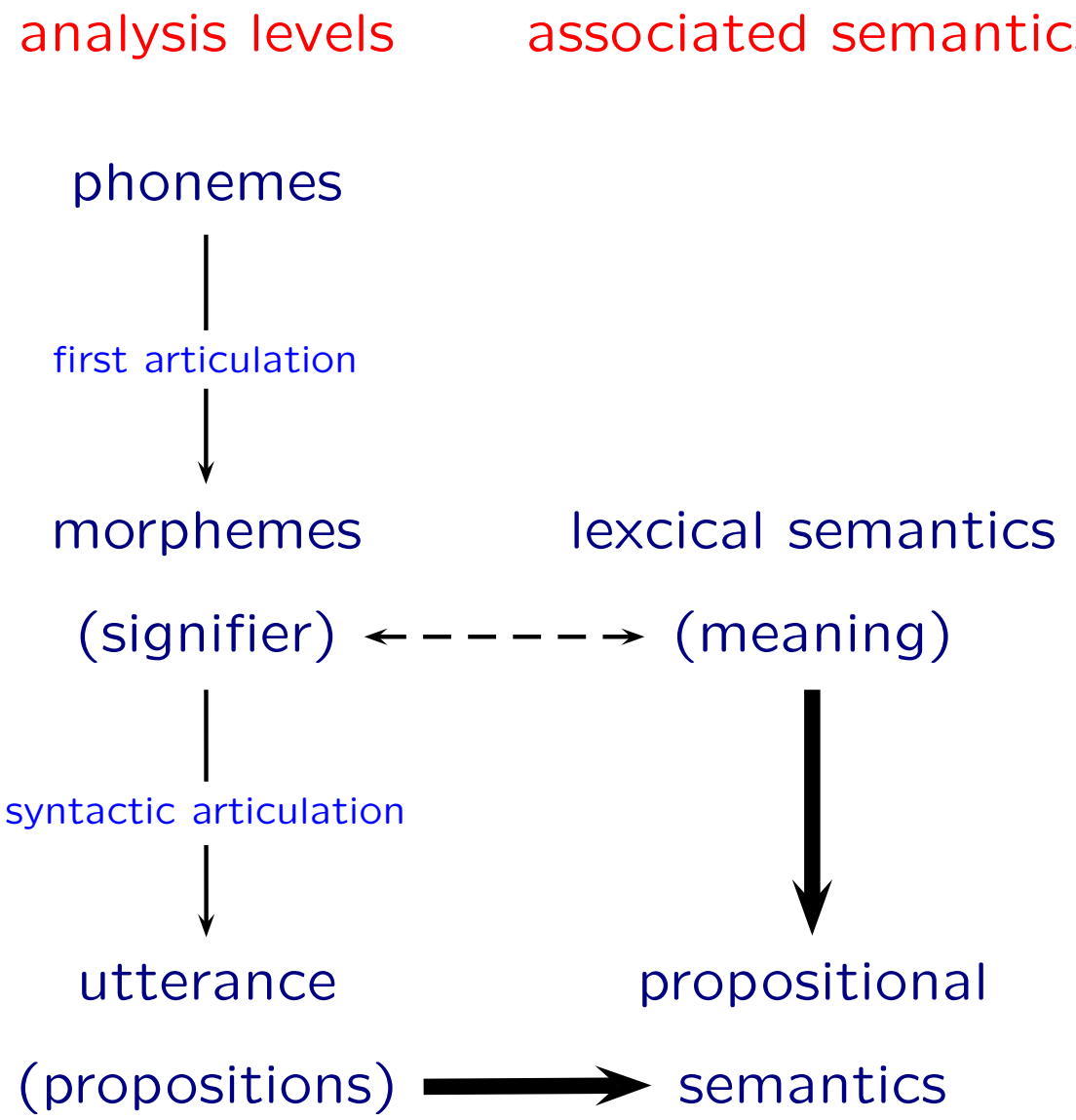
- fundamental level according to Chomsky (not to me!)
- syntax is characterized by grammaticality judgements
- competence/performance distinction
- I-language/E-language distinction
- formal grammars can model syntax
- and many other things...

1. Some specificities of NL

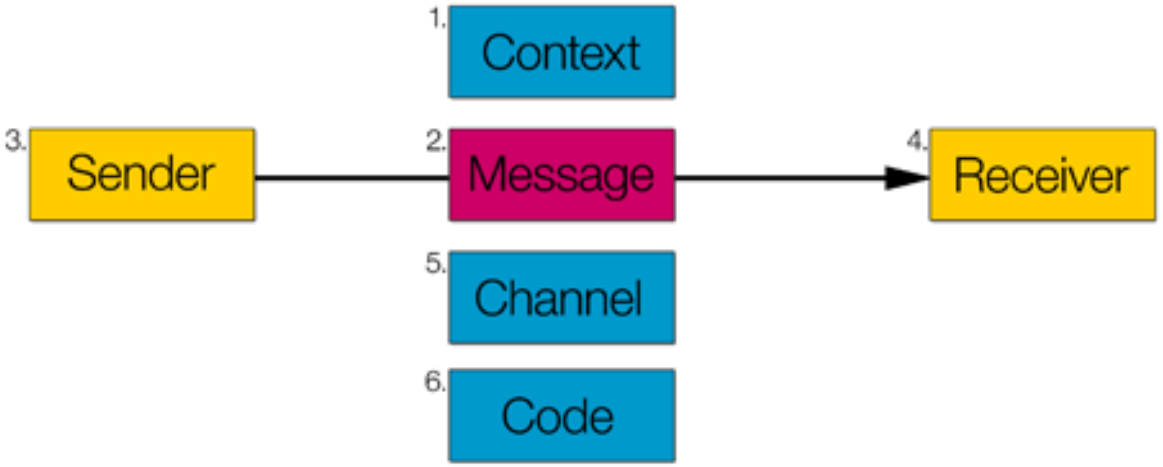
On the nature of semantics

- can be conceived :
  - compositional : the propositional meaning only depends on the lexical meanings and on syntax
  - distributional : "you shall know a word by the company it keeps" (Wittgenstein)
  - can be characterized :
    - at the lexical level : by a combination of primitive semantic units (componential analysis, Schank...) or by a position in a network (de Saussure, semantic networks)
    - at both levels : by a formal language (at least first order logic)

1. Some specificities of NL
The Principle of Compositionality



Functions of NL (Jacobson)



1. referential : the sender expresses contextual information
2. esthetic/poetic : the focus is on the message itself
3. emotive : the sender expresses his/her own feelings
4. conative : the sender wants the receiver to produce an action
5. phatic : the message is intended to check that the channel works
6. metalingual : the message describes its own code

Outline

1. Some specificities of Natural Languages

2. How do they apply to Programming Languages ?

3. On the Nature of Programming Languages

2. How do they apply to PL ?

Basic features of PL

- Turing-equivalent
- compiled or interpreted

Families of PL

- imperative : Fortran, Pascal, Python...
  and the usual "algorithmic pseudo-languages" used in education
- functional : Lisp, CaML
- object-oriented : C++, Java...
- declarative and constraint-oriented : Prolog
- event-oriented : Delphi, Javascript

2. How do they apply to PL ?

Basic features of PL

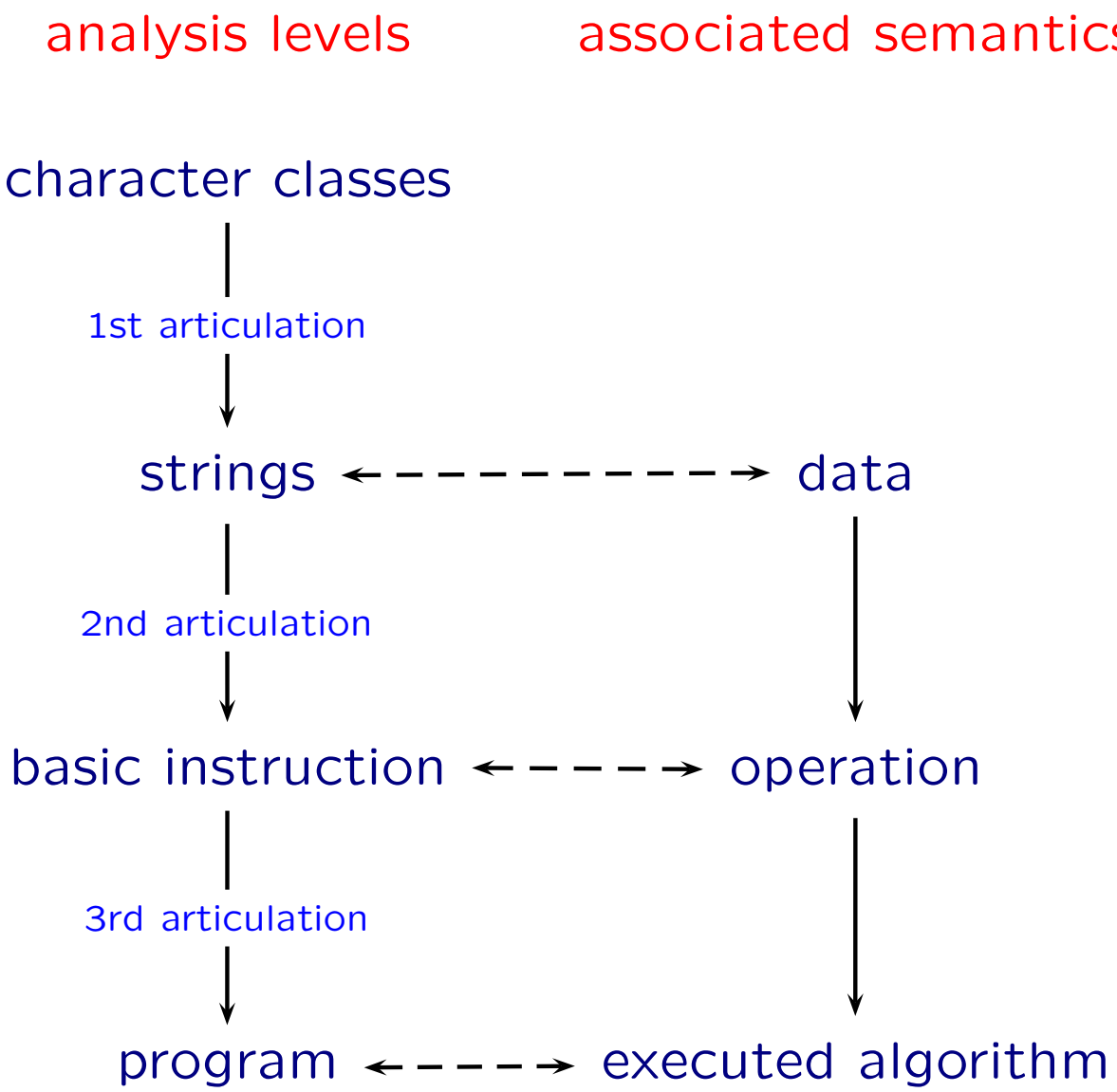
- artificially designed by humans to be implemented in computers
- no oral version
- not so linear : causes of only partial order
  - modularity : fonctions/procedures (in imperative or functional languages) or objects (in OO languages) allow relatively free order of the modules
  - relatively free order of clauses/constraints (declarative languages)
  - interactive object-dependent order in event-oriented languages
  - only the "main" part of the program (when exists) is strictly linear
- slight diachrony (backward compatibility of evolving versions)

2. How do they apply to PL ?

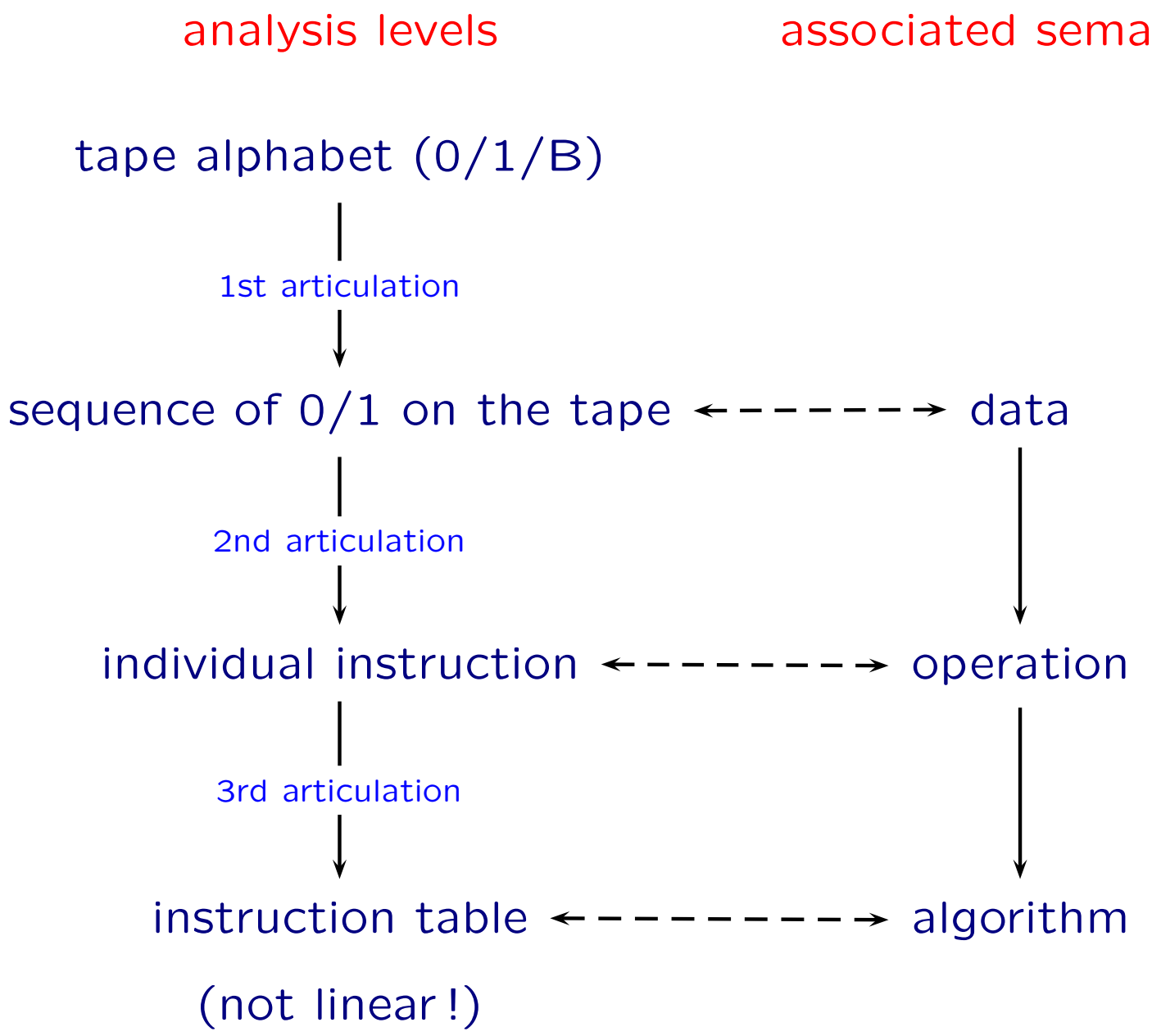
Does the "double articulation" apply to PL ?

- PL are composed of discrete units (!)
  - equivalence classes of alphanumeric characters (are uppercase/lowercase considered identical ? are blank characters ignored or not ?) play the role of phonemes : meaningless distinct basic units
  - strings (either keywords or variables/function names...) play the role of morphemes : smallest meaningfull units
  - basic instructions (or clauses) play the role of syntactically combined strings expressing a potentially successful/not (or true/false) "utterance"
  - a second level of syntax defines well-formed programs (for example : "the declaration of a variable precedes its first assignment", "Begin" precedes "End"...)
- PL seem to have a triple articulation !

2. How do they apply to PL ?



2. How do they apply to PL ?
In a Turing machine



2. How do they apply to PL ?

About ambiguity

- at the string level :
  - distinct strings (for ex. variables) can refer to the same data
  - the same string can refer to distinct data (global/local variables can have the same name)
- at the operation level :
  - the same basic instruction can lead to various results (depending on the value of variables, of an event, of a random choice...) : indexicality ?
  - the same operation can be expressed by various basic instructions
- at the program level :
  - the same program leads to various results (depending on user-dependent input data or event...) but implements a unique function/algorithm
  - various programs can implement the same function/algorithm

2. How do they apply to PL ?

On the nature of “lexical units”

- grammatical units : preserved characters ({}, “ :”, tabulation in Python...) or keywords (var, if/then/else, function, begin/end...) belonging to a finite list
- lexical units : names given to variables or functions/procedures or objects or predicates... by the programmer
- great difference with NL : only grammatical units belong to the common language, the other units are program-dependent, except when they are taken from shared libraries (OO languages)
- data types (boolean, integer, real, string, array, functions...) are similar to POS categories for these units : equivalence classes for a substitutability property while preserving “compilability”
- difference with NL : some PL admit a potentially infinite number of distinc types (functional types in functional languages)

2. How do they apply to PL ?

On the nature of syntax

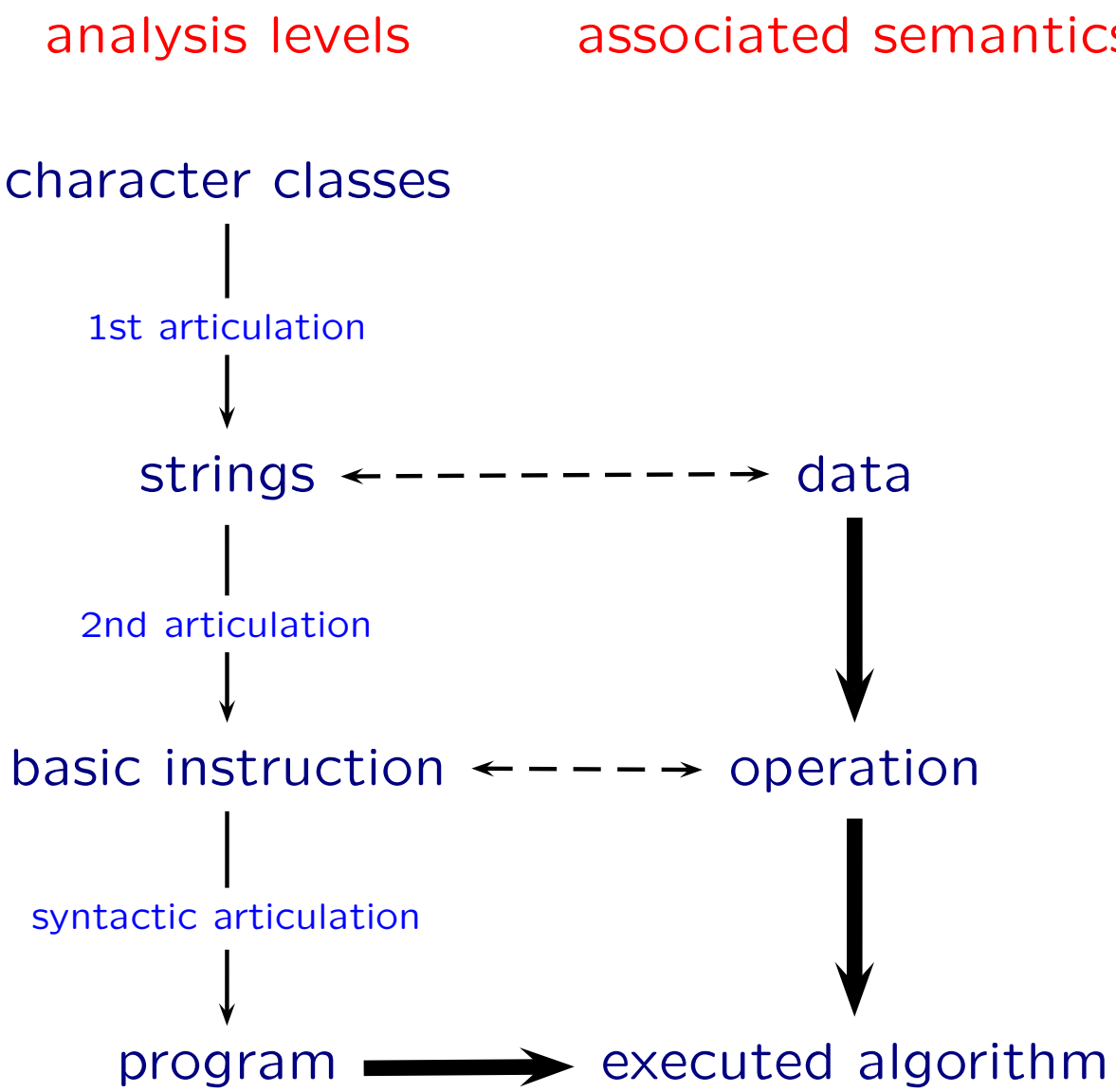
- syntax is characterized by "compilability/interpretability" diagnosis
- two distinct levels of syntax checked at the same time by the compiler/interpreter :
  - basic instructions must be well-formed
  - their sequence must too
- at least for the compiler/interpreter :
  - competence = performance ?
  - I-language = E-language ?
- formal grammars do model the syntax

2. How do they apply to PL ?

On the nature of semantics

- can be characterized :
  - at the string level (variables, functions) : as data values (i.e. internal states of the computer)
  - great difference with NL : no shared "world" between programs, except hierarchically organized (like in semantic networks) classes of objects in OO libraries
  - at the instruction and program levels : as a formal language (λ-calculus, fixed point semantics...)
- can be conceived as globally strictly compositional, except some distributional effects (global/local variables)

2. How do they apply to PL ? Compositionality in PL



2. How do they apply to PL ?

Functions of PL

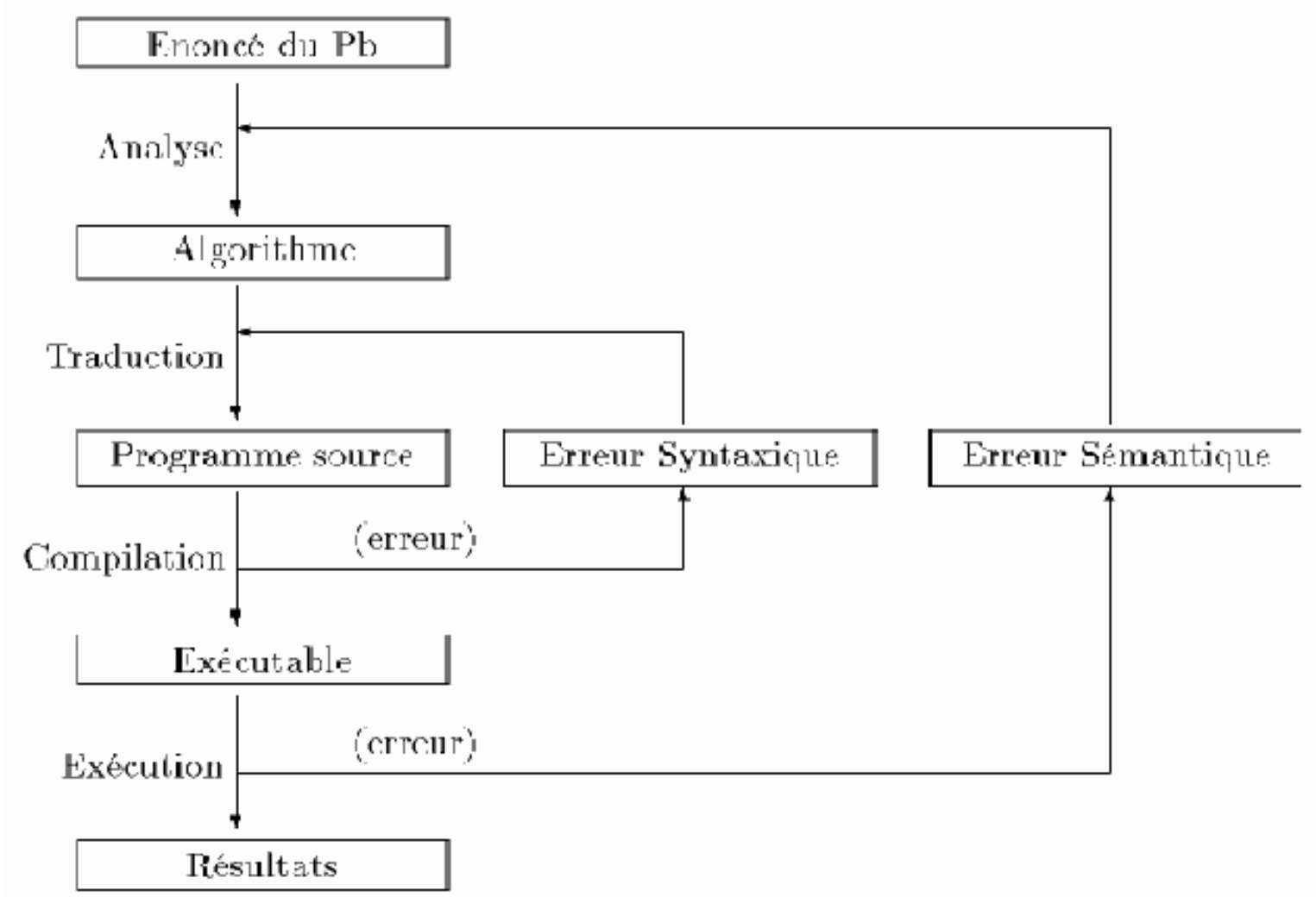


Figure 1: Strategy of Program Design

2. How do they apply to PL ?

Functions of NL applied to PL

- referential :
  - declarative languages (Prolog) are performatively referential
  - OO languages are partly referential too
- esthetic/poetic : there are competitions of poetries written in Perl...
- emotive : ?
- conative (orders) : the basis of imperative languages
- phatic : part of communication protocol programs
- metalingual : a compiler contains a description of a language (which can be its own)

2. How do they apply to PL ?

Functions of PL

```
BEFOREHAND: close door, each window & exit; wait until time.
    open spellbook, study, read (scan, select, tell us);
write it, print the hex while each watches,
    reverse its length, write again;
    kill spiders, pop them, chop, split, kill them.
        unlink arms, shift, wait & listen (listening, wait),
sort the flock (then, warn the "goats" & kill the "sheep");
    kill them, dump qualms, shift moralities,
    values aside, each one;
        die sheep! die to reverse the system
        you accept (reject, respect);
next step,
    kill the next sacrifice, each sacrifice,
    wait, redo ritual until "all the spirits are pleased";
    do it ("as they say").
do it(*everyone***must***participate***in***forbidden**s*e*x*).
return last victim; package body;
    exit crypt (time, times & "half a time") & close it,
    select (quickly) & warn your next victim;
AFTERWORDS: tell nobody.
    wait, wait until time;
    wait until next year, next decade;
        sleep, sleep, die yourself,
        die at last
# Larry Wall
```

Figure 2: “Black Perl” : the most famous poem in Perl

Outline

1. Some specificities of Natural Languages

2. How do they apply to Programming Languages ?

3. On the Nature of Natural/Programming Languages

3. On the Nature of NL/PL

Summary of observed differences

- PL are artificially designed and executed, no oral channel, nearly out of history
- PL have triple articulation, they are more complicated/constrained than NL !
- notion of non-ambiguity not so clear in PL, but global compositional determinism at the algorithm level (same cause, same effect)
- no shared domain of interpretation between programs written in the same PL

3. On the Nature of NL/PL

Specificities of PL

- iteration (possible but rare in NL), systematic indexicality (variables), recursivity
- no "semantics" without syntax

Specificities of NL

- shared domain of reference
- In NL, semantics can be associated to non grammatical utterances by humans
- provocation (D. Israel) : "NL are the PL of the mind"