Knowledge Representation via Verbal Description Generalization: Alternative Programming in Sampletalk Language

Andrew Gleibman
Sampletalk Research
https://sampletalk.github.io/

Knowledge Representation via Verbal Description Generalization: Alternative Programming in Sampletalk Language

Andrew Gleibman
Sampletalk Research
https://sampletalk.github.io/

Sampletalk Technology:
Alternative Logic
Alternative Syntax & Semantics
Alternative NLP

Definition of the Alternative Programming via a Metaphor:

Conventional Medicine
Chemistry
Physics
Electronics
Mechanics
…

⟺

Alternative Medicine
Natural Preparations as Healing Means;
Natural Adaptation of the Organism to the Environment

Conventional Programming
Formal Grammar Theory
DB theory
Logic
Calculus
…

⟺

Alternative Programming
Patterns of Natural Texts as Programming Means

Alignment, Structurization, Generalization and Composition of Natural Texts

Alignment, Structurization, Generalization and Composition of Natural Texts:

Aligned Reasoning

What can John see? A book
What can Mary see? The red table

Aligned Facts

John can see a book
Mary can see the red table

Question ***Answer***

Generalization

what can [A] see? [X]

Generalized Reasoning

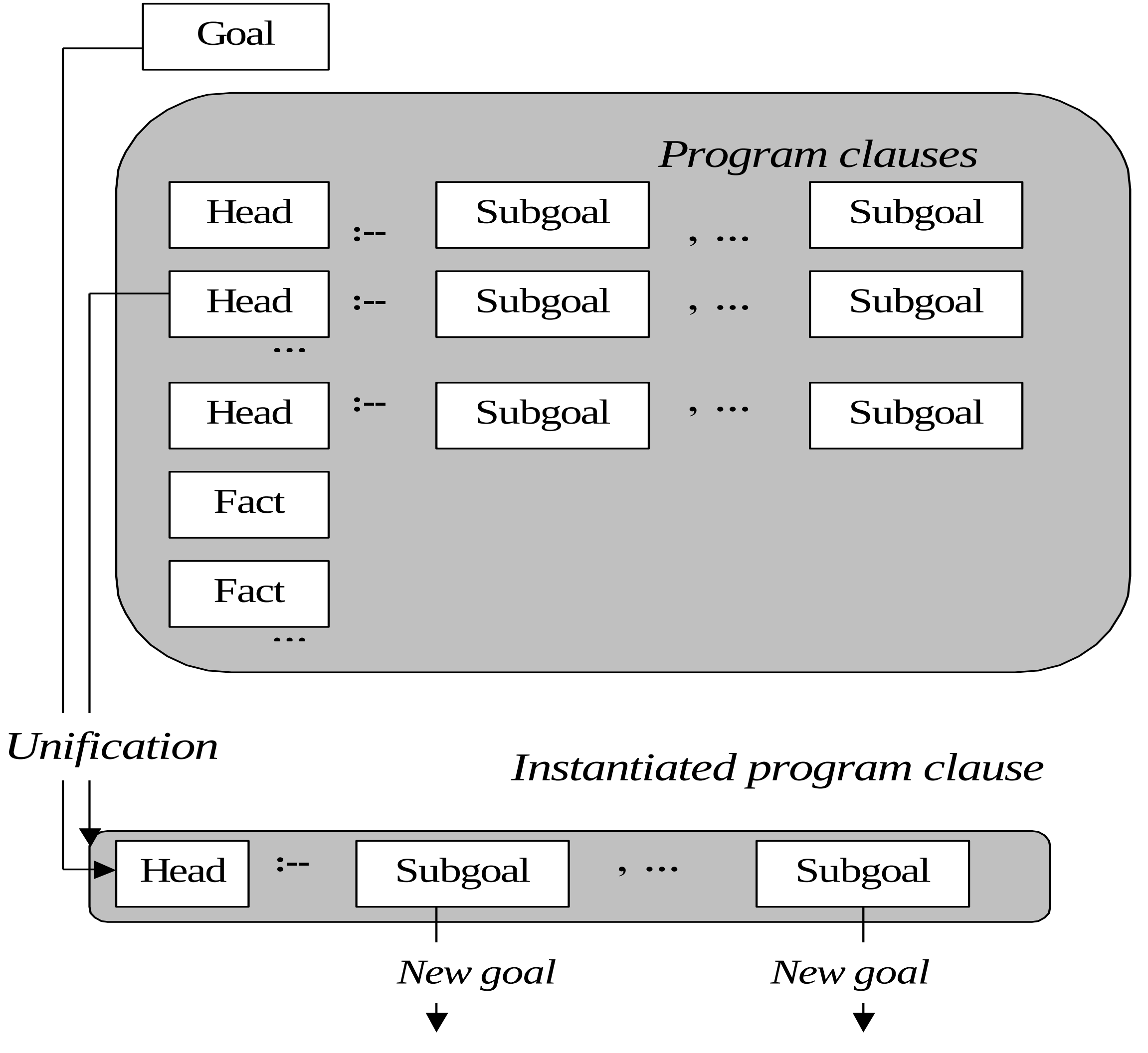
[A] can see [X]

Generalized Pattern of the Facts

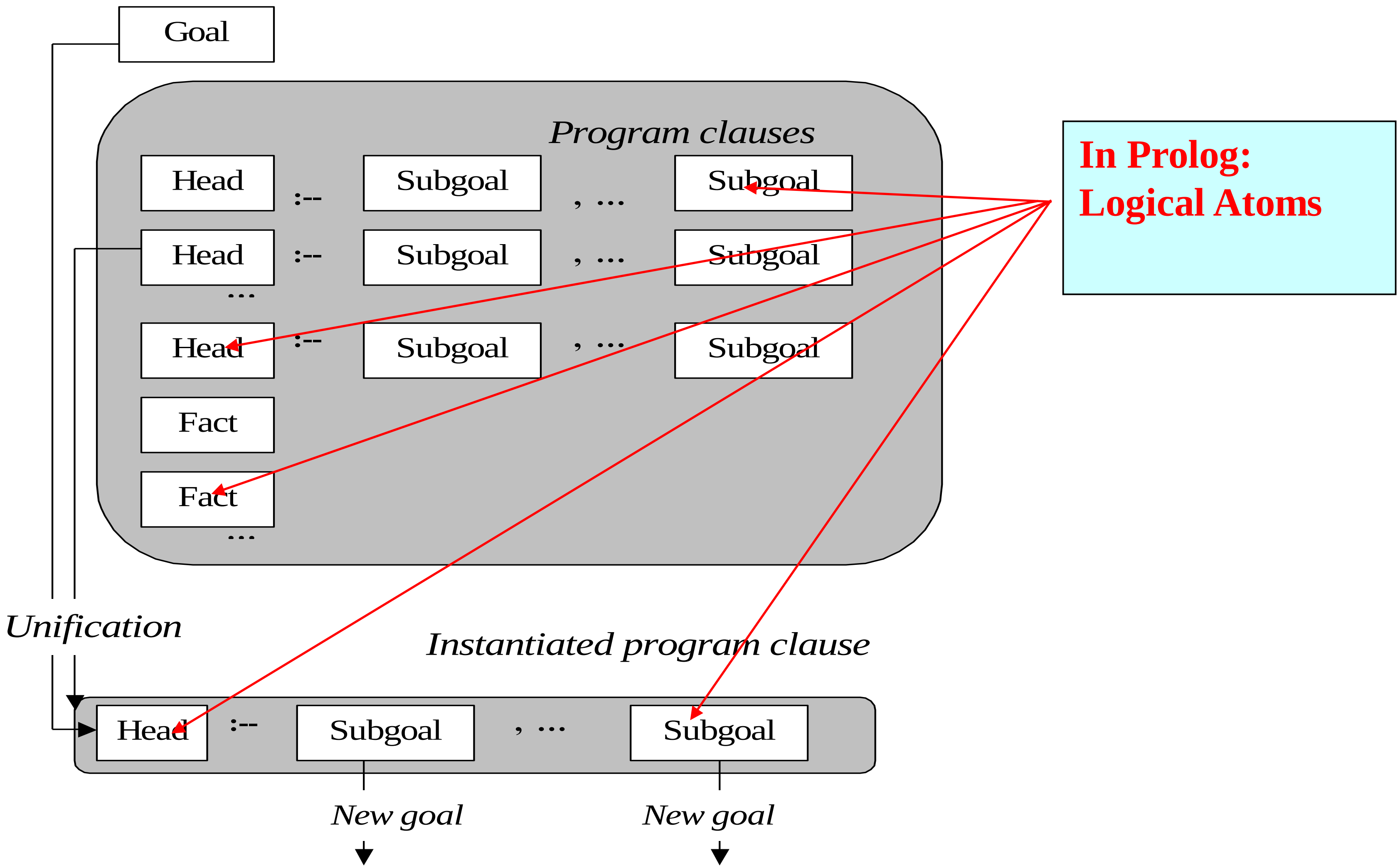
The Composed Sampletalk Clause:

what can [A] see? [X] :– [A] can see [X]..

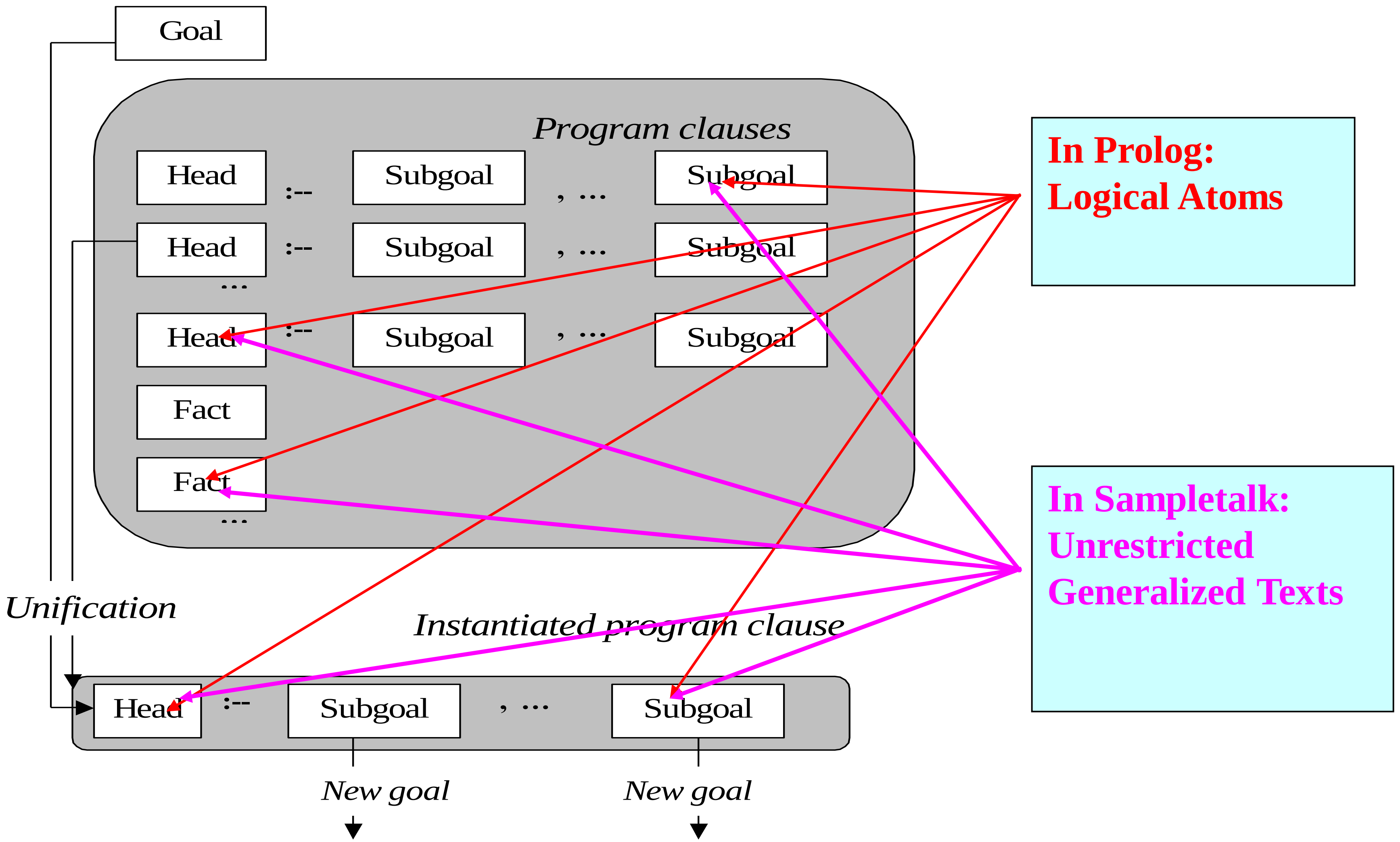
A scheme of a logic program



A scheme of a logic program



A scheme of a logic program



Sampletalk Program Example:

Logico-linguistic inference rules:

Negation

can [A] see [X]? :– [A] can see [X]..
what can [A] see? [X] :– [A] can see [X]..
what is in [A]? [X] :– [X] is in [A]..
what is visible on [A]? [X] :– [X] is visible on [A]..
what is invisible on [A]? [X] :– [X] is in [B] ,, [B] is on [A] ,, ~~[X] is visible on [A]..
what is on [A]? [X] :– [X] is on [A]..
what is on [A]? [X] :– [X] is in [B] ,, [B] is on [A]..
[A] can see [X] :– [A] is standing near [B] ,, [X] is visible on [B]..
[A] is visible on [B] :– [A] is on [B]..
[A] is visible on [B] :– [A] is in [open X] ,, [open X] is visible on [B]..
[A] is standing near [B] :– [A] has approached [B]..

Facts:

[book] is in [open box]..
[notebook] is in [closed box]..
[open box] is on [red table]..
[closed box] is on [red table]..
[john] has approached [red table]..

Sampletalk Program Example:

Logico-linguistic inference rules:

Negation

can [A] see [X]? :– [A] can see [X]..
what can [A] see? [X] :– [A] can see [X]..
what is in [A]? [X] :– [X] is in [A]..
what is visible on [A]? [X] :– [X] is visible on [A]..
what is invisible on [A]? [X] :– [X] is in [B] ,, [B] is on [A] ,, ~~[X] is visible on [A]..
what is on [A]? [X] :– [X] is on [A]..
what is on [A]? [X] :– [X] is in [B] ,, [B] is on [A]..
[A] can see [X] :– [A] is standing near [B] ,, [X] is visible on [B]..
[A] is visible on [B] :– [A] is on [B]..
[A] is visible on [B] :– [A] is in [open X] ,, [open X] is visible on [B]..
[A] is standing near [B] :– [A] has approached [B]..

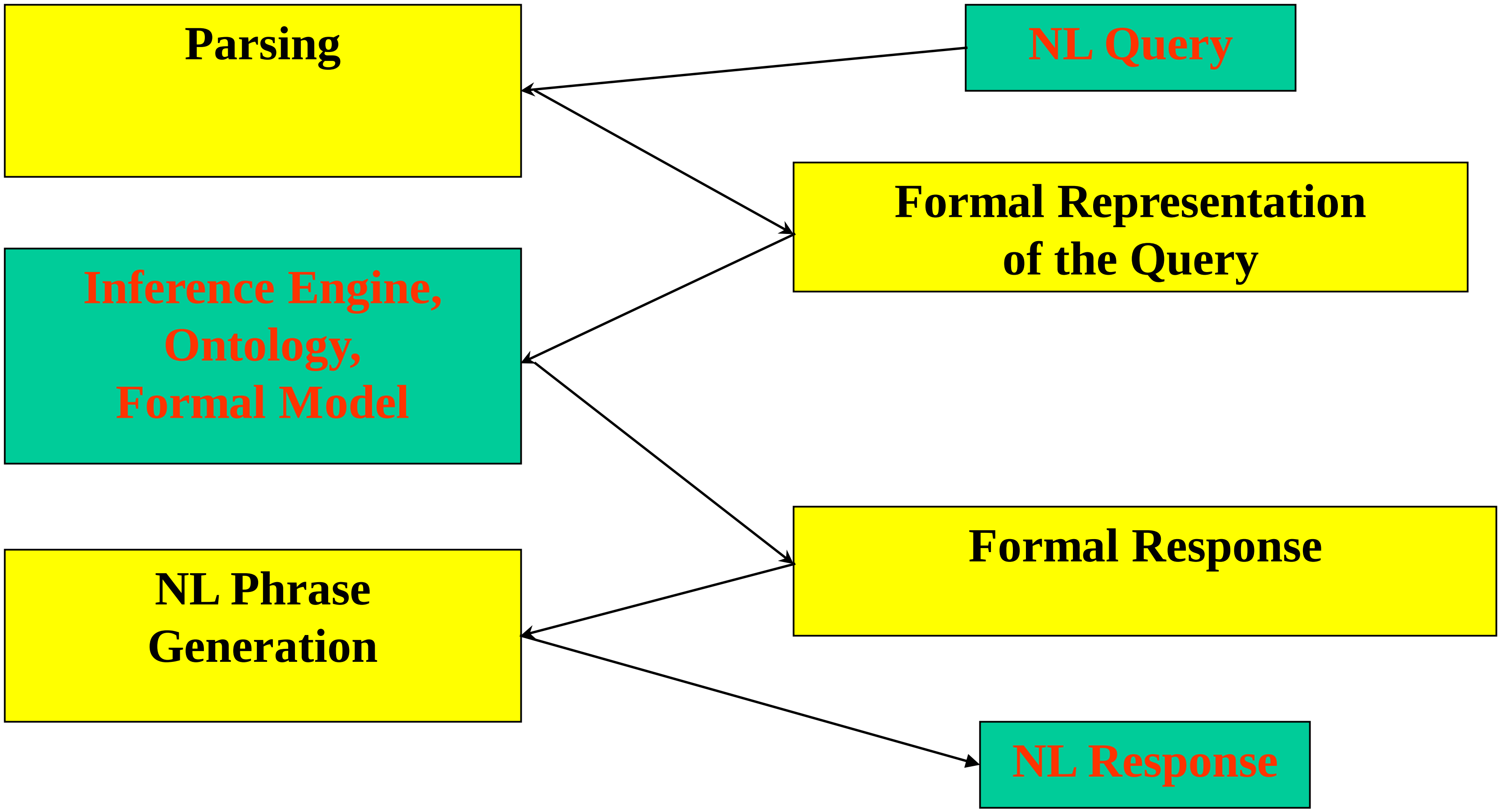
!

Facts:

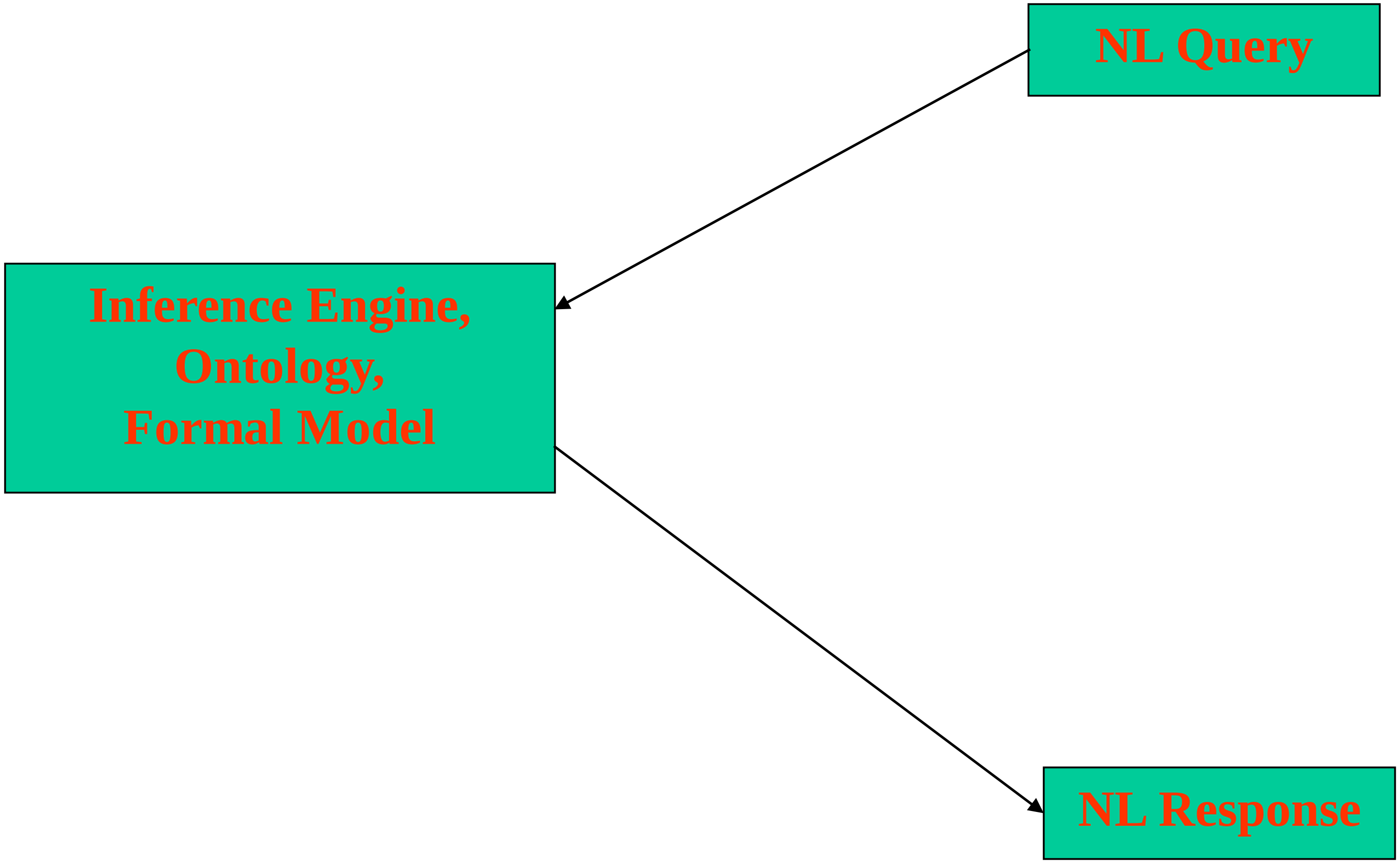
[book] is in [open box]..
[notebook] is in [closed box]..
[open box] is on [red table]..
[closed box] is on [red table]..
[john] has approached [red table]..

Exploiting the Natural Syntax & Semantics of the Original NL Phrases
rather than
Inventing an Artificial Syntax and Making an Artificial Semantic Model

Example Inputs:	*The Outputs:*
1) can [john] see [book]?	**1) can [john] see [book]?** *Yes.*
2) can [john] see [notebook]?	**2) can [john] see [notebook]?** *No.*
3) can [john] see [open box]?	**3) can [john] see [open box]?** *Yes.*
4) what can [john] see? [X]	**4) what can [john] see? [open box].** *Yes;* **what can [john] see? [closed box].** *Yes ;* **what can [john] see? [book].** *Yes*.
5) what is visible on [red table]? [X]	**5) what is visible on [red table]? [open box].** *Yes;* **what is visible on [red table]? [closed box].** *Yes;* **what is visible on [red table]? [book].** *Yes*.
6) what is invisible on [red table]? [X]	**6) what is invisible on [red table]? [notebook].** *Yes*.
7) what is on [red table]? [X]	**7) what is on [red table]? [open box].** *Yes;* **what is on [red table]? [closed box].** *Yes;* **what is on [red table]? [book].** *Yes;* **what is on [red table]? [notebook].** *Yes*.
8) what is in [open box]? [X]	**8) what is in [open box]? [book].** *Yes*.
9) what is in [closed box]? [X]	**9) what is in [closed box]? [notebook].** *Yes*.



Common FOL+NLP approach: The need to design predicates, grammars, a semantic model, and implement a parser, a formal reasoning engine and a NL phrase generator



The alternative NLP approach: No predicates, no grammars, no artificial semantics, no parsers. Everything is defined via immediate interaction of patterns of the existing NL phrases.

Advantages of Alternative Programming -- Logico-Linguistic Inference:

Logico-linguistic inference rules:

can [A] see [X]? :– [A] can see [X]..
what can [A] see? [X] :– [A] can see [X]..
what is in [A]? [X] :– [X] is in [A]..
what is visible on [A]? [X] :– [X] is visible on [A]..
what is invisible on [A]? [X] :– [X] is in [B] ,, [B] is on [A] ,, ~~[X] is visible on [A]..
what is on [A]? [X] :– [X] is on [A]..
what is on [A]? [X] :– [X] is in [B] ,, [B] is on [A]..
[A] can see [X] :– [A] is standing near [B] ,, [X] is visible on [B]..
[A] is visible on [B] :– [A] is on [B]..
[A] is visible on [B] :– [A] is in [open X] ,, [open X] is visible on [B]..
[A] is standing near [B] :– [A] has approached [B]..

!

The alternative NLP approach: No predicates, no grammars, no artificial semantics, no parsers. Everything is defined via immediate interaction of patterns of the existing NL phrases.

Advantages of Alternative Programming -- Logico-Linguistic Inference:

Ontology:

Set of NL phrase patterns with model usage of terms

Current Knowledge:

Representation of facts in the form of verbal descriptions

Query: **NL phrase pattern**

Inference and Linguistic Knowledge:

Representation of inferences in the form of generalized verbal descriptions

Response: **NL phrase**

!

The alternative NLP approach: No predicates, no grammars, no artificial semantics, no parsers. Everything is defined via immediate interaction of patterns of the existing NL phrases.

Advantages of Alternative Programming -- Generating Verbal Explanation of the Sampletalk Reasoning:

can [john] see [book]?	***Goal***

Explanation:

1 [open box] is on [red table],	***DB Fact***
2 [open box] is visible on [red table],	***From 1***
3 [book] is in [open box],	***DB Fact***
4 [book] is visible on [red table],	***From 2,3***
5 [john] has approached [red table],	***DB Fact***
6 [john] is standing near [red table],	***From 5***
7 [john] can see [book].	***From 6,4***
can [john] see [book]? ***Yes***	***From 7***

Advantages of Alternative Programming -- Inheritance of the Syntax and Semantics from the Original NL Phrases:

Predicate Notation

hasChild (anna, jacopo)

is_in (book, open_box)
is_in (notebook, closed_box)
is_on (open_box, red_table)
is_on (closed_box, red_table)
has_approached (john, red_table)

can_see (X,Y)
is_visible_on (X,Y)
is_standing_near (X,Y)

Sampletalk Notation

[jacopo] is child of [anna]
[anna] has child [jacopo]
[anna] is parent of [jacopo]

[book] is in [open box]..
[notebook] is in [closed box]..
[open box] is on [red table]..
[closed box] is on [red table]..
[john] has approached [red table]..

[A] can see [B]
[A] is visible on [B]
[A] is standing near [B]

Algorithmic properties of Sampletalk language – Turing completeness: Any Markov algorithm can be represented as a Sampletalk theory:

THEOREM:

Let M be any Markov algorithm written in a base alphabet U and a local alphabet V, where $(U \cup V) \subset L$ and L is the alphabet of terminal symbols applied in text theories.

There exists a text theory P_M, which, given a goal $[\xi t\xi] \Rightarrow [W]$ (where t is a string, ξ and $\Rightarrow$ are terminal symbols not belonging to alphabets U and V, W is a variable), does the following:

Transforms the goal into a text $[\xi t\xi] \Rightarrow [\xi t'\xi]$ and stops if M transforms string t into a string t′ and stops;

Produces a text $[\xi t\xi] \Rightarrow$ [[fail]] and stops if no rule of M is applicable to some derivation of t, produced by M;

Never stops if string t leads M to an infinite application of the Markov rules.

Algorithmic properties of Sampletalk Language -- Alternative Version of the Church-Turing Thesis

THESIS 1.

Every effective computation can be carried out using generalized patterns of some data examples and a limited set of rules for combining such patterns and the input data. !

THESIS 2 *(no previous generalization needed)***:**
Every effective computation can be carried out using a set of pairs of similar data examples and a limited set of rules for combining such pairs and the input data. !

Philosophical aspects of the Sampletalk Technology

Discussion

Alternative Programming:
Prolog → Sampletalk

Sampletalk as the generalization of Prolog

Alternative Logic:
First Order Predicate Calculus → First Order Text Calculus

Alternative Logic as the generalization of Predicate Logic.
Unification of the generalized texts (axioms) defines the inference

Artificial Semantics vs. Alternative Semantics

Application issues of the two ways to
build a semantic model
should be analyzed

Discussion (Continued)

Pair-wise Alignment and Generalization of a Corpus of Sentences:

a book is in an open box
a notebook is in a closed box
cat is in the cage
a fox is in zoo
john is in his red car
this large table is red

a [X] is in [D] [Y] box
[X] is in [D] [Z]
a [X] is in [Z]
[X] is in [Z]
[X] is [A]

A building material for constructing Sampletalk theories for NLP

Discussion (Continued)

Machine-Learning Setting –
Extracting Sampletalk Theories from Examples:

A set of positive and negative text examples:

$\mathbf{E_+ = \{(s_i, t_i): i = 1, 2, \ldots, n\}}$ *(positive examples)*
$\mathbf{E_- = \{(s'_j, t'_j): j = 1, 2, \ldots, m\}}$ *(negative examples)*

Objective function for learning a Sampletalk theory **P***:*

$$f(Q) = (\ |\{(s_i, t_i) \in E_+ : Q(s_i) = t_i\}| - |\{(s'_j, t'_j) \in E_- : Q(s'_j) = t'_j\}|\ )\ /\ |Q|,$$

where Q is a Sampletalk theory;

$$P = \text{argmax}\,(f\,(Q))$$

Discussion (Continued)

Machine-Learning Setting –
Extracting Sampletalk Theories from Examples:

A set of positive and negative text examples:

[[a + b]] ⇒ [+ a b]
[[x + y]] ⇒ [+ x y]
¬[[u + v]] ⇒ [u + v]
[[a]] ⇒ [a]
[[x]] ⇒ [x]
[[a + [b + c]]] ⇒ [+ a + b c]
[[u – v]] ⇒ [– u v]

[[e + [f + g]]] ⇒ [+ e + f g]
[[e + [f – g]]] ⇒ [+ e – f g]
[[e / [f – g]]] ⇒ [/ e – f g]
[[[a + b] + c]] ⇒ [+ + a b c]
[[[a1 / b2] + c3]] ⇒ [+ / a1 b2 c3]
[[a * b]] ⇒ [* a b]
[[[a + b] * [c – d]]] ⇒ [* + a b – c d]

Theory of the above set of examples:

[[*A Z B*]] ⇒ [*Z C D*] :-- [[*A*]] ⇒ [*C*] ,, [[*B*]] ⇒ [*D*]..
[[*A*]] ⇒ [*A*]..

Discussion (Continued)

Suggested Research:

1) Experiments with automatic Sampletalk theory generation for specific data (e.g., geographical, biological) and applications;

2) Alignment of Non-Text Data: FOPC → FOTC → FOIC → FOOC

First Order Predicate Calculus

First Order Text Calculus

First Order Image Calculus

First Order Object Calculus

Additional Sampletalk Theory Example:
A Theory of Morphological Analysis

Goal:

result of morphological analysis of word [q u e r i e s]: this is X.

The Outputs:

result of morphological analysis of word [q u e r i e s]: this is 2-sg form of verb "q u e r y";
result of morphological analysis of word [q u e r i e s]: this is plural form of noun "q u e r y".

The Theory:

result of morphological analysis of word [X i e d]: this is past form of verb "X y" :–
    word “X y” has canonical morphological tag “verb”..
result of morphological analysis of word [X e d]: this is past form of verb "X e" :–
    word “X e” has canonical morphological tag “verb”..
result of morphological analysis of word [X e d]: this is past form of verb "X" :–
    word “X” has canonical morphological tag “verb”..
result of morphological analysis of word [X i e s]: this is 2-sg form of verb "X y" :–
    word “X y” has canonical morphological tag “verb”..
result of morphological analysis of word [X e s]: this is 2-sg form of verb "X e" :–
    word “X e” has canonical morphological tag “verb”..

Continued

result of morphological analysis of word [X s]: this is 2-sg form of verb "X" :–
word “X” has canonical morphological tag “verb”..
result of morphological analysis of word [X i e n]: this is past-p form of verb "X y" :–
word “X y” has canonical morphological tag “verb”..
result of morphological analysis of word [X e n]: this is past-p form of verb "X e" :–
word “X e” has canonical morphological tag “verb”..
result of morphological analysis of word [X e n]: this is past-p form of verb "X" :–
word “X” has canonical morphological tag “verb”..
result of morphological analysis of word [X y i n g]: this is ing-form of verb "X y" :–
word “X y” has canonical morphological tag “verb”..
result of morphological analysis of word [X i n g]: this is ing-form of verb "X e" :–
word “X e” has canonical morphological tag “verb”..
result of morphological analysis of word [X i n g]: this is ing-form of verb "X" :–
word “X” has canonical morphological tag “verb”..
result of morphological analysis of word [X]: this is canonical verb form "X" :–
word “X” has canonical morphological tag “verb”..
result of morphological analysis of word [X i e s]: this is plural form of noun "X y" :–
word “X y” has canonical morphological tag “singular noun”..
result of morphological analysis of word [X e s]: this is plural form of noun "X e" :–
word “X e” has canonical morphological tag “singular noun”..

Continued

result of morphological analysis of word [X s]: this is plural form of noun "X" :–
word “X” has canonical morphological tag “singular noun”..
result of morphological analysis of word [X]: this is canonical singular form of noun "X" :–
word “X” has canonical morphological tag “singular noun”..
result of morphological analysis of word [X]: this is adverb "X" :–
word “X” has canonical morphological tag “adverb”..
result of morphological analysis of word [X]: this is adjective "X" :–
word “X” has canonical morphological tag “adjective”..
…

word “b a c k”	has canonical morphological tag	“singular noun”..
word “b a c k”	has canonical morphological tag	“adverb”..
word “b a c k”	has canonical morphological tag	“verb”..
word “b a k e”	has canonical morphological tag	“singular noun”..
word “b a k e”	has canonical morphological tag	“verb”..
word “b a k e r”	has canonical morphological tag	“singular noun”..
word “b a k e r y”	has canonical morphological tag	“singular noun”..
word “q u e r y”	has canonical morphological tag	“singular noun”..
word “q u e r y”	has canonical morphological tag	“verb”..

…

Additional Sampletalk Theory Example --
A Theory for Logic Formula Transformation:

The goal (variable W stands for the transformation result):

(∀ x0)[a(x0,y)] ∨ (∀ x0)[b(x0,t)] --> W..

The theory:

(Q X)[F] ∨ (Q X)[H] --> (Q X)(Q Z)([F] ∨ [G]) :–
X is variable,,
Z is variable,,
not(F contains Z),,
(Z/X)[H]=[G]..
x0 is variable..
X10 is variable :– X0 is variable..
AXB contains X..
(Y/X)[AXM]=[AYN] :- (Y/X)[M]=[N]..
(Y/X)[A]=[A]..

The result:

(∀ x0)[a(x0,y)] ∨ (∀ x0)[b(x0,t)] --> (∀ x0)(∀ x10)([a(x0,y)] ∨ [b(x10,t)]).

Additional Sampletalk Theory Example --
Another Theory for Logic Formula Transformation:

The goal (variable W stands for the transformation result):

according to the rule R, the result of shifting quantifiers
in the formula (∀ x0)[a(x0,y)] ∨ (∀ x0)[b(x0,t)] is formula W..

The program:

according to the rule 2a from chapter 5, the result of shifting quantifiers
in the formula (Q X)[F] ∨ (Q X)[H] is formula (Q X)(Q Z)([F] ∨ [G]) :-
X is notation for variable,,
Z is notation for variable,,
not(word F contains word Z),,
the result of replacing X by Z in formula H is G..
x0 is notation for variable..
X10 is notation for variable :- X0 is notation for variable..
word AXB contains word X..
the result of replacing X by Y in formula AXM is AYN :-
the result of replacing X by Y in formula M is N..
the result of replacing X by Y in formula A is A..

The result:

according to the rule 2a from chapter 5, the result of shifting quantifiers in the formula
(∀ x0)[a(x0,y)] ∨ (∀ x0)[b(x0,t)] is formula (∀ x0)(∀ x10)([a(x0,y)] ∨ [b(x10,t)]).