

Fluid Construction Grammars

Luc Steels^{1,2}, Joachim De Beule¹, Joris Van Looveren¹, Nicolas Neubauer¹

¹Vrije Universiteit Brussel – Artificial Intelligence Lab
http://arti.vub.ac.be.

²Sony CSL, Paris
http://www.csl.sony.fr

Goal:

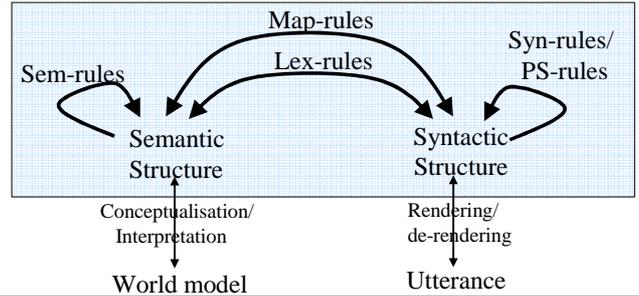
Develop a formalism for construction grammar and computational mechanisms for parsing and production
Should be usable in multi-agent simulations of language evolution.

Requirements:

- > Linguistic Requirements:
 - ✓ Representation of constructions (meaning-form mappings)
 - ✓ Representation of hierarchical categorisation rules (syntactic and semantic)
 - ✓ No formal distinction between lexicon and grammar
- > Processing Requirements:
 - ✓ No distinction between parsing and production
 - ✓ Multiple hypotheses tracked in parallel
 - ✓ Partial analysis if rules are missing or violated
- > Adaptation Requirements:
 - ✓ New categories can be added or changed at any time
 - ✓ New constructions can be added at any time
 - ✓ Rules are in competition for dominance in the population

Approach:

- >Lexicon and grammar are expressed by constraints on feature structures
- >Construction application through matching + unification over feature structures
- >All form aspects are represented in a declarative way
- >Multi-tasking to explore multiple hypotheses



Ex: ((walk ev1 true) (walk-1 ev1 obj1)
(john obj1))

<==>

“John walks”

Lex-rules: relate part of semantic structure to lexemes (words):

<pre>(def-lex walk-entry ((?unit (referent ?event) (meaning (== (walk ?event ?truth) (walk-1 ?event ?object)))))) <-> ((?unit (form (== (string ?unit "walks"))))))</pre>	<pre>(def-lex john-entry ((?unit (referent ?obj) (meaning (== (john ?obj)))))) <-> ((?unit (form (== (string ?unit "john"))))))</pre>
---	---

Syn-rules: categorise aspects of syntactic structure:

<pre>(def-syn john-noun ((?unit (form (== string ?unit "john")))) -> ((?unit (syn-cat (== noun (number singular))))))</pre>	<pre>(def-syn walk-verb ((?unit (form (== (string ?unit "walks")))) -> ((?unit (syn-cat (== verb (number singular))))))</pre>
--	--

Sem-rules: categorise aspects of semantic structure

<pre>(def-sem walk-1-agent ((?event-unit (referent ?event) (meaning (== (walk ?event ?true) (walk-1 ?event ?object)))))) --> ((?event-unit (sem-cat (== (move ?event) (agent ?event ?object))))))</pre>
--

Sem	MOVE	<Agent>
	walks	<walk-1>
Syn	Pred	Subject

Map-rules: relate part of semantic structure to syntactic

<pre>(def-map SV-construction ((?parent-unit (referent ?ev1) (sem-subunits (== ?event-unit ?agent-unit))) (?event-unit (referent ?ev1) (sem-cat (== (move ?ev1) (agent ?ev1 ?obj)))) (?agent-unit (referent ?obj)) <-> ((?parent-unit (syn-cat (== SV-sentence))) (?event-unit (syn-cat (== (subject ?parent-unit ?agent-unit)))) (?agent-unit (syn-cat (== (predicate ?parent-unit ?event-unit))))))</pre>

Ps-rules: relate constructions to phrase structure

<pre>(def-ps SV-sentence ((?top-unit (syn-cat (== SV-sentence))) (?subject-unit (syn-cat (== (subject ?top-unit ?subject-unit)))) (?predicate-unit (syn-cat (== (predicate ?top-unit ?predicate-unit)))) <-> ((?top-unit (syn-cat (== sentence) (form (== (precedes ?subject-unit ?predicate-unit)))) (?subject-unit (syn-cat (== noun (number ?number)))) (?predicate-unit (syn-cat (== verb (number ?number))))))</pre>

Syntactic structure:

<pre>((top-unit (syn-subunits (unit-1 unit-2)) (syn-cat (sentence)) (form ((precedes unit-1 unit-2))) (unit-1 (form ((string unit-1 "john")))) (unit-2 (form ((string unit-2 "walks")))))</pre>

Semantic Structure:

<pre>((top-unit (sem-subunits (unit-1 unit-2)) (unit-1 (referent ?obj2) (meaning ((john ?obj2))) (unit-2 (referent ?ev1) (meaning ((walk ?ev1 ?truth) (walk-1 ?ev1 ?obj1))))))</pre>
--

<pre>((top-unit (syn-subunits (unit-1 unit-2)) (unit-1 (form ((string unit-1 "john")) (syn-cat (noun (number singular)))) (unit-2 (form ((string unit-2 "walks")) (syn-cat (verb (number singular))))))</pre>

<pre>((top-unit ... (unit-1 (referent ?obj2) (meaning ((john ?obj2))) (unit-2 (referent ?ev1) (meaning ((walk ?ev1 ?truth) (walk-1 ?ev1 ?obj1))) (sem-cat ((move ?ev1) (agent ?ev1 ?obj1))))))</pre>
--

Syntactic Structure:

<pre>((top-unit (syn-subunits (unit-1 unit-2)) (syn-cat (sentence SV-sentence)) (form ((precedes unit-1 unit-2))) (unit-1 (syn-cat (subject top-unit unit-1)) (noun (number singular)) (form ((string unit-1 "john"))) (unit-2 (syn-cat (predicate top-unit unit-2)) (verb (number singular)) (form ((string unit-2 "walks"))))</pre>

Semantic Structure:

<pre>((top-unit (sem-subunits (unit-1 unit-2)) (unit-1 (referent ?obj1/2) (meaning ((john ?obj1/2))) (unit-2 (referent ?ev1) (sem-cat ((move ?ev1) (agent ?ev1 ?obj1/2))) (meaning ((walk ?ev1 ?truth) (walk-1 ?ev1 ?obj1/2))))))</pre>

“John walks”

<->

```
((walk ?ev1 ?truth)
(walk-1 ?ev1 ?obj1/2)
(john ?obj1/2))
```