The Grammar as Science Project - Part 2

Richard K. Larson (SUNY - Stony Brook) 3.July 1997

I. Brief Recap

A. The Goal
- Linguistics instruction as "science education"
- Science education should aim to produce individuals who understand the practice of scientific inquiry, not just its results; we want people who can acquire new scientific knowledge and solve new problems.

B. The GAS Plan
- Create/revise sophomore-level courses that would introduce students from a wide variety of backgrounds to scientific reasoning & procedure using syntax & semantics as the medium.
- Create software tools to assist active processing & discovery.
- Use appropriate representations & models - not necessarily the ones embraced by current linguistic theory.

II. Teaching Semantics to Undergrads

A. Virtues
1. Semantic theory involves both broad conceptual issues and precise technical methods.
- Questions addressed have fundamental "philosophical" content - the nature of meaning and its relation to human action and thought
- The construction of semantic theory involves precise reasoning with formal axioms and deductive principles.

Because of this dual nature, semantics is effective in developing mature scientific thinking, which must combine attention to detail with an ability to reflect on general issues. Students must learn to assess not only the technical content of proposals, but also broader conceptual commitments.

2. Semantics also has the virtue of broad interdisciplinary connections and implications; offers a natural entry point into the wider domain of cog. sci.

B. Pedagogical Challenges
- Demands mastery of deductive technique very much like that involved with proofs or derivations in logic.
- Demands some philosophical maturity and ease with abstractions.
- We need a theory in which assumptions, rules and results can be made explicit and precise; but one that is formally accessible. Truth-conditional semantics is explicit and precise; but MG is much too complicated.

III. Developing *SEMANTICA*

A. Basic Functions We Wanted to be Instantiated
- Calculation (of derivations of truth-conditions)
- Visualization (of situations where truth is evaluated)
- Modeling (of the human semantic mechanism)

B. Meanings and Truth conditions (Davidson 1967)
Rita Ajay ke picche hai means that Rita is behind Ajay
Rita Ajay ke picche hai is true if, and only if, Rita is behind Ajay
Val(t, *Rita Ajay ke picche hai*) iff back_of(rita, ajay)

Rules:

1)	Val(x, [N *Ajay_ke*])	iff	x = ajay
2)	Val(x, [N *Rita*])	iff	x = rita
3)	Val(<x,y>, *picche_hai*)	iff	back_of(x,y)
4)	Va(t, [$_S$ N1 N2 V])	iff	Val(x, N1) & Val(y, N2) & Val(<x,y>, V)

U(niversal)I(nstantiation), S(ubstitution of) E(quivalents)

C. Challenges
1. Getting a computational system to deliver the right truth-conditions (and not ones merely equivalent to them) is an unsolved problem. Coming up with the right derivational procedure required a good deal of prolog research:

Example: Val(t, *Rita Ajay ke picche hai*) iff back_of(rita, ajay) *versus*
Val(t, *Rita Ajay ke picche hai*) iff back_of(rita, ajay) &
((rita = moti) \/ (rita ≠ moti))

2. Specifying templates for semantical info meant deciding many complex questions about the kinds of semantic rules and notations users might, or might want to employ.

3. Modeling certain kinds of semantic notions (negations) turned out to be very complex, in virtue of well-known questions in logic-programming.

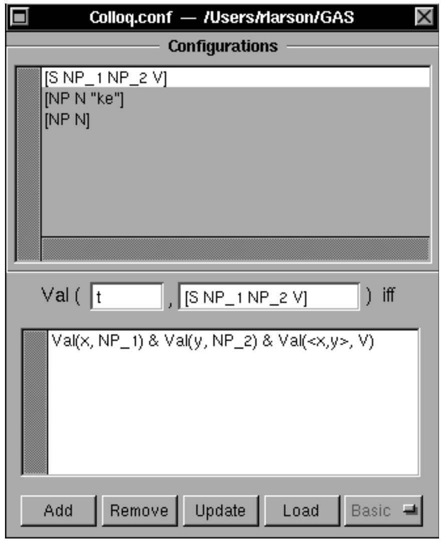
IV. *SEMANTICA :* How It Works

Modeled on the logic teaching program *TARSKI'S WORLD*

- User enters a semantic theory (config rules or config rules + lexicon)
- User selects an input tree
- *Semantica* attempts to build truth conditions for input using the rules
- *Semantica* displays the results of successful builds (readings).
- Results can be evaluated in a graphical world

Rules are entered in windows with appropriate templates.

Config Rules:



Colloq.conf — /Users/larson/GAS

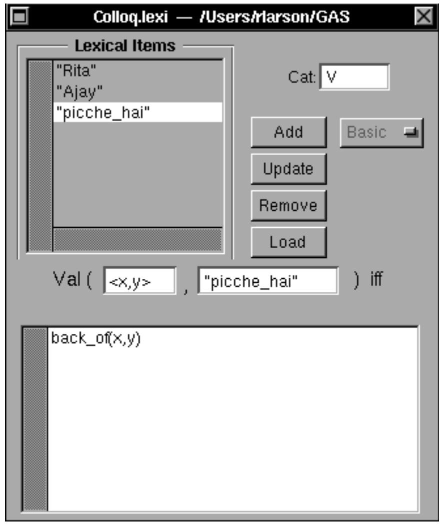
Configurations

[S NP_1 NP_2 V]
[NP N "ke"]
[NP N]

Val (t , [S NP_1 NP_2 V]) iff

Val(x, NP_1) & Val(y, NP_2) & Val(<x,y>, V)

Lexical Rules:



Colloq.lexi — /Users/larson/GAS

Lexical Items

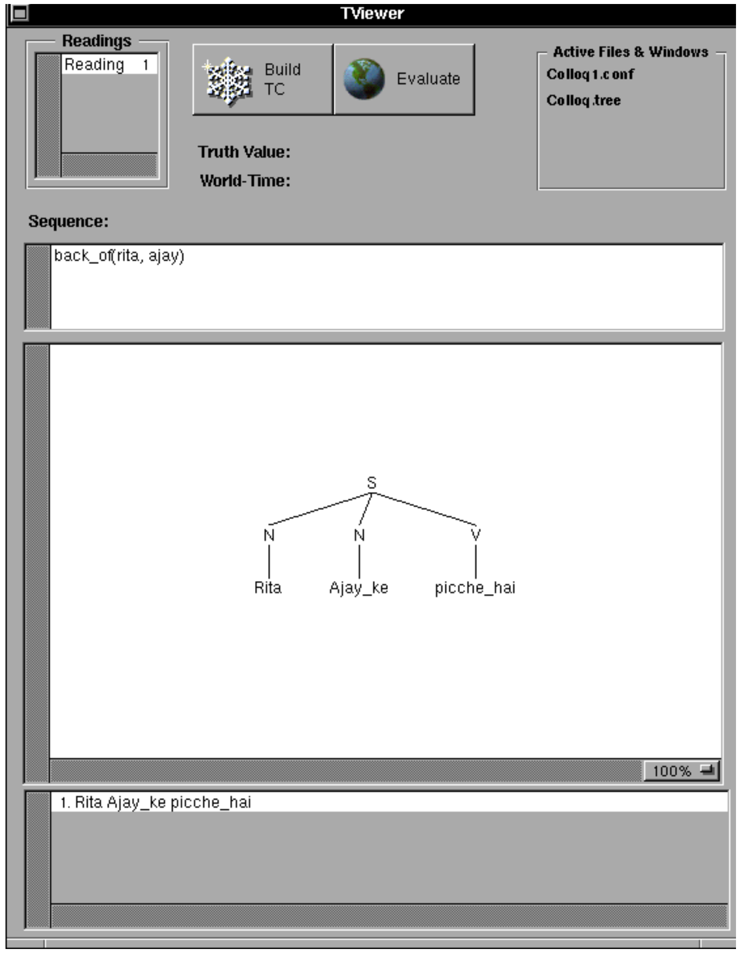
"Rita"
"Ajay"
"picche_hai"

Cat: V

Val (<x,y> , "picche_hai") iff

back_of(x,y)

The user selects an input tree (here the tree for *Rita Ajay ke picche hai*), and instructs *Semantica* to build truth conditions. Results are displayed in TViewer window:



TViewer

Readings: Reading 1

Active Files & Windows: Colloq1.conf, Colloq.tree

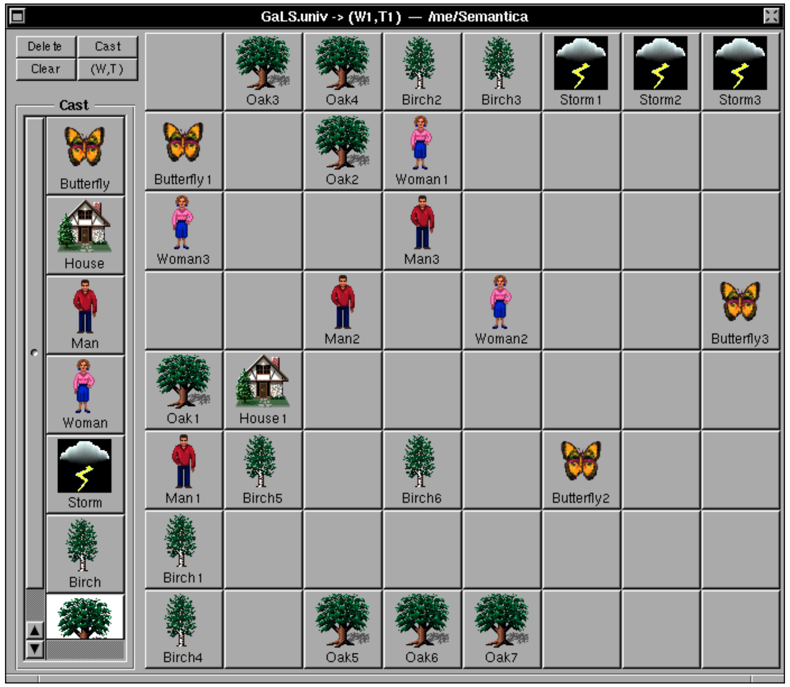
Truth Value:
World-Time:

Sequence:

back_of(rita, ajay)

1. Rita Ajay_ke picche_hai

Semantica also permits TCs to be evaluated against pictorial worlds containing a cast of characters that the user constructs. Suppose rita is identified as Woman1 and ajay as Man1. Then, *Semantica* will evaluate *Rita Ajay ke picche hai* as TRUE! in this world:



The visualization function in *Semantica* plays a broader role than in *Syntactica*:

- In syntax, trees are potentially complex, but the data that motivate the trees are fairly simple (judgments of acceptability, ambiguity, etc.). It's the former that needs to be visually represented.

- In semantics, we produce truth-conditions for a tree. So there is the tree to be represented. Beyond this, the judgments forming the data are far more complex (situations that verify or falsify). The data itself must be visually represented. Consider the judgments for *Every man is between a woman and two trees*. In determining what this sentence means you might want to test the predictions of your theory against situations where you could intuitively judge truth. If your theory predicts truth and falsity correctly, it is supported.

V. *SEMANTICA* : How Its Used in Our Course

A. Modeling Grammatical Competence

We want students to understand the idea of attributing knowledge of a formal system as a way for explaining competence and abilities. *Semantica* provides a concrete example of a formal system, and a potential model of speakers' semantical competence.

B. Constructing and Comparing Semantic Theories

1. Like *Syntactica*, *Semantica* provides a useful workbench for building, testing, and refining theories (represented in *Semantica* as config rule files or config rule files + lexicon files). Its window structure also visually separates the parts of an explanation:

- the data (input trees)
- the theory proposed to explain the data (what's in the rule windows)
- the predictions of the theory (the TCs and their TVs in worlds)

2. *Semantica* also provides a convenient workbench for comparing theories Competing rule files can be quickly loaded and tested against a given input to check what TCs are assigned.

Example: In *LIN 346* students explore alternative semantic analyses of proper nouns, verbs, adjectives, etc.:

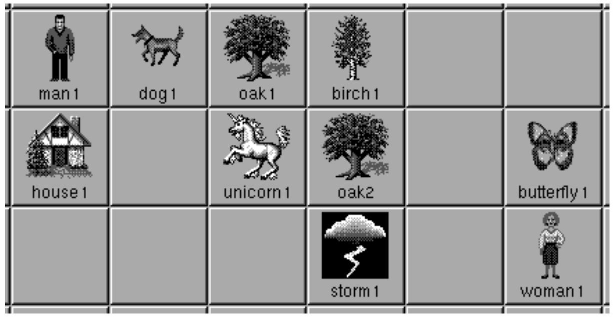
Val(x, "Galileo") iff	x = galileo
Val(x, "Galileo") iff	dropped_cannonball_from_tower_of_Pisa(x) & discovered_telescope(x) & discovered_moons_of_Jupiter(x) & said_"eppur_si_muove"(x)

Val(x, "runs")	iff runs(x)
Val(<e,x>, "runs")	iff running(e) & Agent(e,x)

C. Mental Representation of Semantic Rules

We want students to understand what it means when we attribute a piece of semantic knowledge to a speaker - for example, lexical knowledge. *Semantica*'s world module allows for the evaluation of six static positional relations (in_front_of, back_of, left_of, right_of, adjoins, between). Students explore the rules "known" by *Semantica* that are responsible for its evaluations.

Example: When we say someone knows TCs like back_of(rita,ajay), plausibly we're attributing concepts corresponding to the pieces: individual concepts for rita and ajay, and a concept of "between". What are the concepts? With "between" we can explore it by checking when *Semantica* assigns truth and falsity. Intuitively, which objects are between man1 and woman1 in the following picture? Which objects does *Semantica* count as between?



Example: *Semantica* has no concepts for actions like running, laughing, hitting, giving, etc. or for states like knowing, believing, sitting, having, etc. But it provides a useful framework for thinking about them: What are our concepts of these actions/states? What would be the crucial features in representing them graphically? How do other elements interact with them; e.g., how do we recognize/represent running-quickly, giving-quickly, etc.?

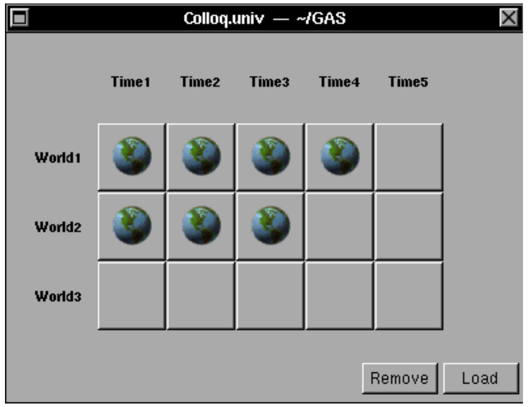
D. Investigating Specific Constructions

Students explore many specific natural language constructions, including names, predicates, pronouns, tenses, modals, quantifiers, and adverbs. *Semantica* assists in this process. The Lexicon and Configuration windows let the user begin with a basic referential semantics for words and phrases and expand that theory to include worlds, times and assignments of values to variables (sequences). The universe window aid in visualizing the semantic import of these parameters.

Example: The truth conditions of sentences with modals and tense are potentially complex. Consider *Some dog possibly was always between Jill and Chris*. The semantics of this sentence is standardly taken to require reference to other worlds and times, e.g.:

|{x : POS(-(PST(-(between(x, man1, woman1)))))} /\ {y : dog(y)}| > 0

Semantica allows for the evaluation of such quantified modal expressions in universes consisting of worlds arranged by time and possibility. A typical universe window looks like this:



The worlds are opened by clicking on their cells. Once opened, a world can be declared as the current world-time and evaluation made with respect to it. The universe interface provides a convenient tool for building models of world circumstances and for testing the semantic rules which yield the truth conditions that are evaluated in them.

VI. Some Lessons

The courses in the *Grammar as Science* project are now three years old. The syntax course employing *Syntactica*, and the semantics course employing *Semantica* have been offered three times. We are developing assessment instruments based on the clinical interview model worked out in Honda (1994). Preliminary, anecdotal evidence is positive. Below are some general lessons we've learned about the process of recasting a traditional lecture class and adopting a more exploratory, lab-based course model.

1. Students report that the initial burden of the courses is heavy since it involves mastering both new content material as well as a new computer tool.

It takes some time to become fluent with the applications, and hence for their value as tools for exploring the subject matter to become natural and clear. Furthermore, learning a new user environment (even one as transparent as NeXTSTEP) takes a bit of time.

2. The lab-format demands a considerably heavier time commitment than a traditional lecture-based course, both in terms of preparation and support.

We found that for a courses of the kind instituted at Stony Brook to be effective, students must be instructed carefully in the use of the new tools. Students cannot be simply turned loose with them, and left to manage as best they can.

3. The use of software instructional tools does not decrease staffing demands, but rather <u>increases</u> them.

In our courses, use of the new tools has entailed not only the use of a graduate student teaching assistant, even for classes as small as 20 students, but also one or more undergraduate student lab assistants, generally recruited from past semesters in the course. This increased undergrad participation was an unlooked for benefit. Students like very much being lab TAs, hanging out with and helping their friends.

4. The use of exploratory tools produces opportunities for students to work together cooperatively that were not present in the lecture format.

Assignment of exercises that are worked in a lab of networked computers not only furnishes students with a natural reason for gathering outside of class, it also provides additional avenues for helping and communicating with each other. Debugging of grammars and semantic rules, coming up with data to test alternatives, is very easy to do in a setting where one can work together in front of a single monitor, or send results and rule sets quickly across a network.

5. The use of exploratory tools of the kind developed in GAS produces a high level of engagement by students.

The task of producing rules that generate trees or interpret them correctly takes on an objective and very concrete character in the context of getting a machine to produce a desired result. Students appear to find this setting both personally challenging, but also, in some important sense, nonjudgmental. This latter dimension seems particularly important for students who do not view themselves as "science-" or "analytical-types". In general we have found that students often appear to be captured by working on a problem in this way.

6. The production of of exploratory tools offers many avenues for student participation.

Undergrad students were extensively involved in the layout and testing of the *Syntactica* and *Semantica* software. They not only did routine work like finding bugs, constructing text and graphics for Help files, they also actively participated in interface design, and in the design of the accompanying text materials. This gave them opportunities for Lab experience in Linguistics, and also provide many opportunities for interaction with the Computer Science Dept.

7. Unless you have a stable of programmers (which brings its own coordination problems) retained on a long-term basis, design in a <u>high-level environment</u>.

GAS chose NeXTSTEP as its design and development environment. This environment is not widely used (and was not in 1991 at start up), however it offers remarkable

advantages in ease and speed with which apps can be proto-typed, tested and refined. This had two big advantages:

• Attention to what matters. The project group could stay focused on the central questions of how the app should behave, how it should be used, what we want to be able to do, etc. It allowed us to avoid low-level programming questions of how to get a window with a certain button do what we wanted.

• Programmer continuity. Our programming was done by students. Students are only around for a limited amount of time. Furthermore, one programmer is rarely happy with another programmer's code. Using a high-level environment it was possible for two persons to do all of the interface programming on *Syntactica* and *Semantica* (Freire and Gomez), and finish it in four years.

VII. Where to Now?

1. The project is preparing texts for the two courses described here.

LIN211	*Grammar as Science*
LIN346	*Semantics as Science*

These texts use an innovative graphical layout style, which was developed and is being overseen by one of our former undergrad project members, Ms. Kimiko Ryokai, now at the MIT Media Lab.

2. Porting of *Syntactica* and *Semantica* from NeXTSTEP to the Windows environment is underway. We have a nearly complete beta-version of *Syntactica* for WIN 95. *Semantica* will be done by next year. This will make our results available nationally.

3. Modularizing *Semantica*. Our general goal is to extend Semantica by adding "modules" comparable to those added to *Mathematica* over the years

We are currently applying for funds to support the creation of an Events Module for *Semantica*, in which actions and states are represented compositionally and graphically, and against which natural language sentence involving actions and states can be evaluated. A crucial idea pursued in the project is that event-types corresponding to the linguistically significant verb classes of natural language constitute the appropriate objects of symbolic depiction. Rather than representing different kinds of motion events, for example, we give a representation of the type: motion-event, corresponding to the linguistically motivated class of motion verbs