Teaching Syntax with *Trees*

Colin Phillips, University of Delaware
colin@udel.edu

to appear in GLOT International, 3.7 (September 1998)

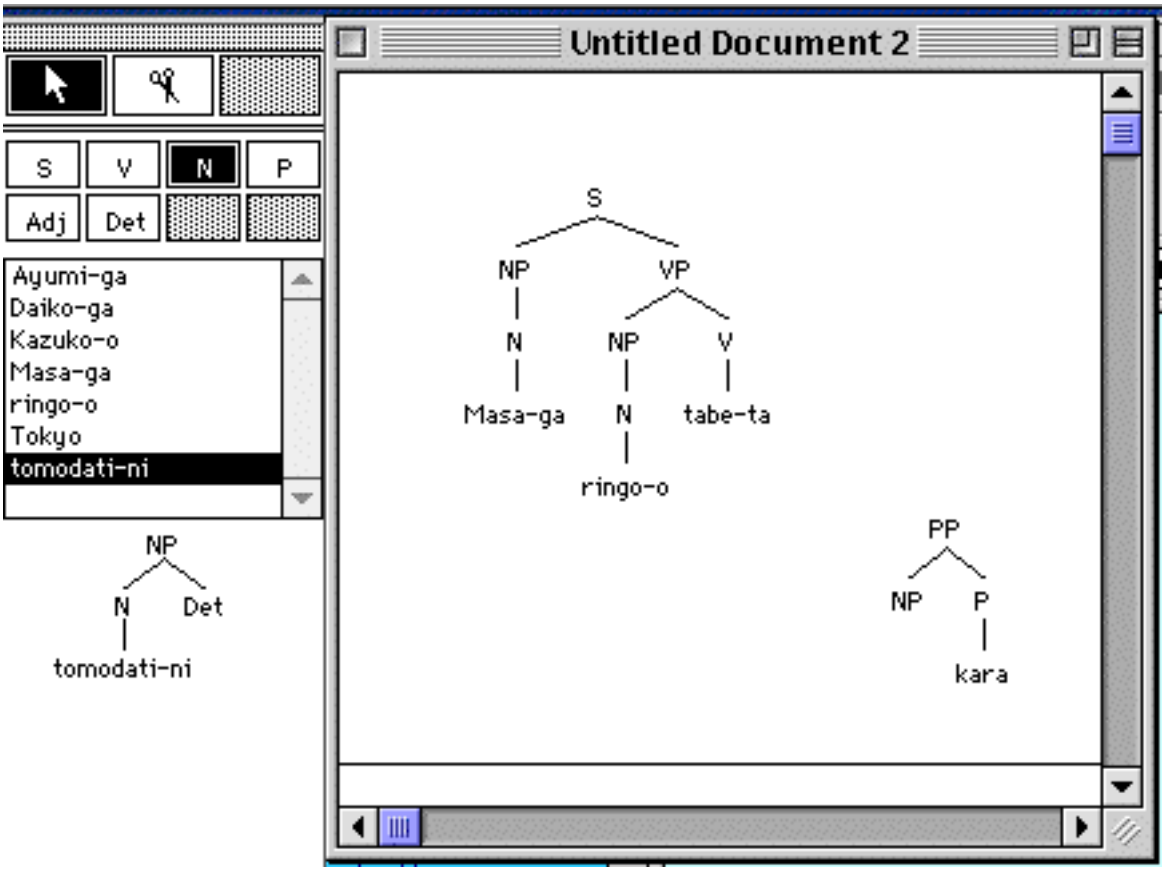
I do not have particularly good motor coordination, and it took me many long years to learn how to swim. Part of the problem was that I was so worried about sinking and breathing in lungfuls of chlorinated water that I couldn't really concentrate on how to paddle and kick. But as soon as somebody gave me a buoyancy aid and plugged up my nose, I got the hang of swimming in no time, and soon was one of the most regular kids at our local pool.

For many beginning linguistics students, getting the hang of syntax can be rather like learning to swim was for me. Watching other people diagram sentences on a blackboard makes it look all too easy, but when you sit down to try to do it for yourself, it's too easy to be held back by incidental details and pitfalls (e.g., can lines cross? is NP S V' a rule of English?). This can make the bare basics of syntax so frustrating that it becomes impossible to progress to anything *interesting*, and as a result the field can appear quite pointless.

Trees 2.1.2 is an elegant little program that may just have what it takes to get students around the frustrating pitfalls of basic syntax, and make it much easier to give students a hands-on introduction to genuinely interesting problems in syntax. The program was created by Sean Crist and Tony Kroch at the University of Pennsylvania. At present Trees runs only on Macintoshes, but a version that will run on Windows PCs is currently under development. A demonstration version may be downloaded for free from the *Trees* web page at http://www.ling.upenn.edu, a personal license can be purchased for $25, and group licenses may be purchased for higher amounts. At the University of Delaware we purchased an institutional license, which allows the program to be installed on any campus computer, and allows students and faculty to install the program at home or in a dormitory room. Last semester the program was used by about 200 University of Delaware students in two large introductory linguistics classes, so we have some experience with using the program with large groups. I have used *Trees* for the 2-week syntax component in an introductory class with 90 students.

The core of *Trees* is a graphic interface which allows users to select tree 'fragments' from a lexicon and combine them to form complete phrase markers in the *Trees* workspace. Once assembled, the trees can be saved, printed, or inserted into papers or other documents. What makes the program more than just a tool for drawing phrase structure trees (other tools exist for this purpose, such as Cascadilla's *Arboreal* font, see GLOT 3.1) is the fact that the range of available tree fragments, lexical items, and how they may combine, is entirely under the control of the user. The user – or the instructor – may create grammars entirely to their specification, and then use *Trees* to demonstrate or test what the grammar generates.

Learning to construct simple trees using a pre-built grammar is trivially easy. Simply use the mouse to drag tree fragments from the lexicon window into the *Trees* workspace, and then connect the tree fragments by dragging a pair of identical nodes over each other, e.g. by dragging a tree fragment dominated by an NP node over an NP daughter of S. If the grammar currently in use allows the nodes to join, then the tree fragments will automatically merge into one. The program's graphics have a simple elegance – the length and angle of tree branches adjusts automatically to prevent lines from overlapping or becoming too crowded, and the result is an invariably well-proportioned tree diagram. [This feature makes the program attractive for use in writing papers.] I gave a brief demonstration of the program to my class using a computer projector, and none of the students had any problems in later using the program on his/her own for homework assignments (well *almost* nobody).



Creating a simple grammar for my students to use was less trivial, but still fairly easy to learn. After following the basic instructions provided with the program and looking at a couple of the sample grammars available from the *Trees* web site, I set out to build a grammar which would generate the sentences in one of the syntax exercises in Fromkin & Rodman's *Introduction to Language*. Learning the basics took about half an hour, and building the grammar took about another hour. It took a little more time to create instructions for the students to use when running the program, and to get it installed in various campus computer labs.

Given that I could have just asked the students to do the homework with pen and paper, was the extra effort worth it?

It was well worth it, as a very informal experiment that I conducted shows. Students were strongly encouraged to use *Trees* for their sentence-diagramming homework, but I also gave them the option of not using the program. When we read the completed homework assignments, we found that students who did not use the program made *six times* as many mistakes as students who did use the program. Even allowing for some bias due to the fact that the students themselves chose whether to use the program or not, the contrast is massive. Among the students who did not use the *Trees* program, the standard range of mistakes, most of them irrelevant to the task at hand, was well represented – in particular failure to use the grammar rules that were provided. Students who did use *Trees* made almost no errors; the main problem among this group of students was that a few students did not understand that the tree fragments had to be connected at all.

Why the difference between the two groups of students? Simple. The *Trees* grammar was constructed so as to make it *impossible* to make irrelevant errors, and as a result it allowed students to focus on the more interesting syntactic questions. Like my primary school swimming instructor, the program eliminated irrrelevant barriers to learning. The other valuable contribution of the program is that it forced the students to think about sentences in structural terms – because pieces of structure, rather than words, are the building blocks used in the program. Students who did not use the program typically found it much harder to move away from thinking about sentences as linear strings of words, and phrase markers as no more than a bunch of mysterious lines drawn above the words. In making this critical conceptual shift easier for students, I think that *Trees* already repaid the extra time required in preparing the class materials.

One thing that I plan to do differently the next time I use *Trees* for an introductory class is make sure to mix exercises in which students use the program with exercises in which they transfer that knowledge to pen and paper tree diagramming. *Trees* does a great job of holding the learner's hand, and preventing irrelevant mistakes, but students need to learn to work without the helping hand as well, and some of my students became too dependent on the program's help, as we discovered in a few cases at exam time. With clever design of grammars, the program can be used to gradually give students more independence, by building in the possibility of ungrammatical structures which the students must recognize, and avoid. My colleague Caroline Heycock built a nice little *Trees* grammar to do just that also available from the Delaware Linguistics web site).

So far I have only talked about extremely basic applications of *Trees* to the very earliest stages of syntax teaching, which is where I have used it most. The program can also be used for more advanced courses. I have not done this yet, but the *Trees* web site at U. Penn. contains links to a number of grammars which Tony Kroch and Beatrice Santorini have created for use in more advanced courses. My simple grammars contained only rewrite rules and no transformational operations, and conformed to an early 1980s

style X-bar theory. For such simple grammars, grammar-building involves little more than specifying the range of possible tree fragments in the grammar's lexicon. However, a wide variety of more sophisticated grammatical models may be used, in which case the user needs to learn more about the *Trees* scripting language. This will be enjoyable for some, but may be formidable for others. A variety of transformational operations may be allowed, and these may be constrained in many different ways, from the most hip to the most antiquated formalisms. The downloadable sample grammars on the Trees web site show a number of examples of transformational grammars for *Trees*.

I should add that the program seems to be very robust: out of myself and the almost 200 other people who used *Trees* at the University of Delaware last Fall, there were no reports of bugs or system crashes, and almost no problems of compatibility with different Mac models and operating system versions.

What would I like to see to make *Trees* even more useful?

From the perspective of the grammar-consumer, the program is already excellent. Even by Macintosh standards it's a very easy-to-use program, with an extremely elegant interface. One nice addition would be greater flexibility in saving trees as picture files for use in other applications. For in-class presentations, the possibility of highlighting or re-coloring specific nodes or tree branches would be useful. But these are minor points.

What I would most like to see is extra resources which make life easier for the grammar creator. The programming manual which is included with the program is comprehensive in its coverage, and extremely clear in the most basic steps of grammar creation, but could usefully provide more guidance for users between novice and expert levels, which is where most users will quickly find themselves. An on-line library of pre-built grammars to download, inspect, and modify would also be very useful. Tony Kroch has already created a small library of sample grammars, available from the *Trees* web site, and hopefully other users will contribute grammars as the program becomes more widely used. The extreme flexibility of the program means that it is not hard to spend a good deal of time specifying the basic properties of individual lexical entries. This process is improved by a feature of the program which allows the grammar creator to edit existing lexical items to create new items. Any additional features which would streamline the process of grammar creation for novice-to-intermediate level users would be very useful.

All in all, though, I was very impressed with *Trees*, and plan to continue using it in my classes. I would highly recommend it to others, especially for introductory level syntax teaching, provided that your institution has the computing support to make classroom demonstration possible, and has enough computers to make student access feasible. Until the Windows version of Trees is available (which should be in early 1999) substantial resources for Macintoshes are needed: in this respect we are quite lucky at the University of Delaware, and I imagine that North American universities are in general more likely to have good Macintosh support than universities in Europe or other parts of the world. *Trees* probably won't convert all of your students into syntax fanatics, but it

could make beginning syntax a whole lot less frustrating (a number of our students rate the program as one of the most useful aspects of the course). This is the point where I should say that it allows you to do for your introductory syntax students what my swimming instructor did for me, and at some level it does – but I can't in all honesty say that skill in syntax saves as many lives as skill in swimming does.

The official *Trees* web site at UPenn is at http://www.ling.upenn.edu.
My course materials for *Trees* are at http://www.ling.udel.edu/colin.