Contemporary Issues in Technology and Teacher Education, 20(3), 562-573.

Gossip and Other Life Sentences

A Simple Grammar

Paul E. Goldenberg
Educational Development Center

Buzz, buzz!

Shakespeare

Hamlet, Act Il, Scene ii

What is the simplest structure of gossip?

At the very least, there must be a subject about which one gossips: it's no fun gossiping if you aren't gossiping about someone. There must also be an allegation (a predicate) that purports to be true about the subject: no fun just whispering names unless one can tie some juicy tidbit to those names. This is already the beginning of a "model" of gossip, a specification of its nature.

Technical Terms—Common Meanings

When the words *subject* and *predicate* are used close together, they seem so tied to grammar that one tends to forget that they have non-technical meanings as well. A subject is a topic; a predicate is what one has predicated, asserted, about the topic.

Sometimes the subject of gossip is an actor (like *Dale*) and the predicate an action (like *cheats*). By restricting ourselves to gossip of this type, we are further refining the model, agreeing that we won't deal with gossip like *Dale is a creep* (where there is no real action and therefore no actor). Here are some examples of gossip that follow our model. All the "actors" are people (a still further refinement).

Dale cheats.

Dana loves to walk. (But is *loves to walk* really an action?)

Chris talks a mile a minute.

Sandy yells.

It's convenient to have an abbreviated way of describing the model. Because we are building two-part sentences, it makes sense to have two-part descriptions.

In my first attempt at modeling gossip, I referred to its two components only as SUBJECT and PREDICATE, but that "code" doesn't capture the essence of gossip. After all, the sentence *Trees are made of wood* also has a subject and a predicate. When I decided to consider only certain kinds of subject and predicate, that clarified the model somewhat. That second model might be best expressed as ACTOR ACTION. A third refinement eliminated sentences like *Dogs bark* and might be encoded as PERSON ACTION. The words we choose as descriptors for the model depend on our purpose and our taste. Most of the time, we want to remind ourselves or clarify to others what our model means. We might choose any of those three descriptions or, for that matter, nonwords as well. For this WHO DOESWHAT seems to be particularly clear.

Etude 1.1 To write a program that can produce these sentences, create the following procedures. In most Logos, it is best to do this by first entering the editor.

```
TO GOSSIP
    OUTPUT (SENTENCE WHO DOESWHAT)
END

TO WHO
OUTPUT PICK [SANDY DALE DANA CHRIS]
END

TO DOESWHAT
OUTPUT PICK (CHEATS. [LOVES TO TALK.] →
[TALKS A MILE A MINUTE.] YELLS.]
END
```

If you used the editor, now get out of the editor, defining the procedures.

Like human languages, computer languages, including Logo, have existing vocabularies. Some computer languages prevent changes to these meanings; others (like human languages) allow changes, but at the risk of confusion or ambiguity. Some Logo implementations, for example, already have WHO defined with a very different meaning and purpose from what we would like for modeling gossip.

> If your language already has a vocabulary item—a procedure—with a name you would have liked to use, simply pick a different name. For example, you might choose PERSON instead of WHO and use it whenever WHO is used in this text. Note that this requires a change not only in a procedure's title (TO PERSON instead of TO WHO) but also in the instruction in GOSSIP.

Etude 1.2 Test the procedures you have written by typing PRINT GOSSIP a few times. It may do something like this:

```
?PRINT GOSSIP
CHRIS LOVES TO WALK.
?PRINT GOSSIP
DANA CHEATS.
?PRINT GOSSIP
DANA TALKS A MILE A MINUTE.
```

How many different sentences can GOSSIP, defined this way, produce? How many times do you have to run it before it produces all the possible combinations? You may find it easier to use Logo's REPEAT command than to type PRINT GOSSIP repeatedly.

```
?REPEAT 15 [PRINT GOSSIP]
.
.
.
```

Etude 1.3 You have examined the output from GOSSIP, but you haven't looked into its component parts. Try typing

```
REPEAT 15 [PRINT WHO]
```

or

```
REPEAT 15 [PRINT DOESWHAT]
```

Each procedure has a "life" of its own.

How GOSSIP Works

It is useful to think of the three procedures you have just created as language-generating machines, *sources* of language. I will represent sources like these as boxes, with *spouts* out of which some language product emerges. (See Figure 1.1.)

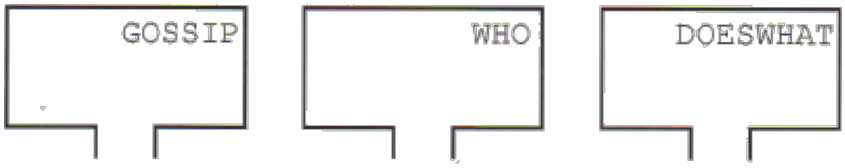

Figure 1.1: Each of these source procedures is represented with a spout through which it outputs some piece of language.

These diagrams do not show what is inside the machines. We will x-ray the machines a bit later. For now, all that is important is the external anatomy.

Other machines such as PRINT, are not sources of information: rather, they are consumers of information and cannot operate without getting some input. (PRINT cannot work without being told what to print.) They receive the information, do something with it (such as put it on the screen), and then stop without producing any new objects (for example, words, sentences, or numbers) for other machines to use. We can think of machines like these as *destinations*. Their diagrams show this graphically: they do not have spouts at the bottom because they do not output, but they do need one or more hoppers at the top through which they receive their input. PRINT has one hopper in which the object to be printed is placed (See figure 1.2.)

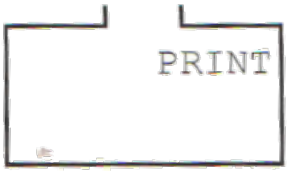

Figure 1.2: PRINT *has a hopper for input but no spout for output. Thus it is an object's final destination.*

Everything that WHO, DOESWHAT, and GOSSIP need in order to operate properly is built directly into them, so they need no hoppers for inputs.

The Logo statement PRINT GOSSIP uses two machines and is diagrammed as shown in figure 1.3.

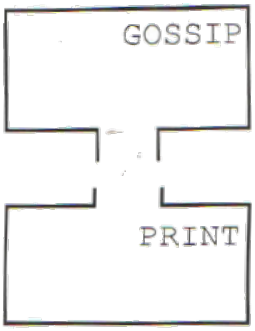

Figure 1.3: Output from the source flows into the destination's hopper. Without the destination, Logo wouldn't know what to do with the object produced by the source.

By showing which machines are sources and which are destinations, the diagrams also make clear how the machines fit together.

Now let's look inside GOSSIP to see how it is constructed and how it works. A diagram of an x-ray of GOSSIP shows WHO and DOESWHAT machines along with a SENTENCE machine and a wavy arrow. (See figure 1.4.)

As you can see, SENTENCE is a third kind of a machine. Like PRINT, it is a consumer of information. But it has an output: it is not a final destination. It performs a combining operation, gluing together the objects dropped into hoppers, and outputting the assembled product through its spout.

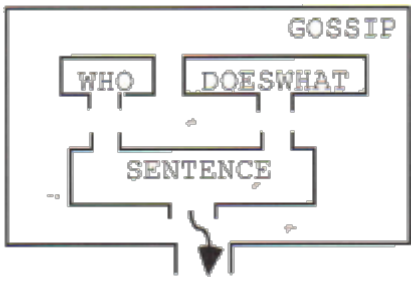

Figure 1.4: An x-ray of the GOSSIP *machine.*

In mathematics, SENTENCE would be called a function because for any given input — in this case, *pair* of inputs — its output is unique. In computer science and programming, the word *function* tends to be less restricted so, to avoid confusion, we will choose a different term and just call this kind of machine a processor. Figure 1.5 is a diagram of SENTENCE all by itself.

Figure 1.5 A diagram of SENTENCE.

To express in Logo that we want to create a SENTENCE of the components generated by WHO and DOESWHAT, we write (SENTENCE WHO DOESWHAT) and we diagram it as shown in Figure 1.6.

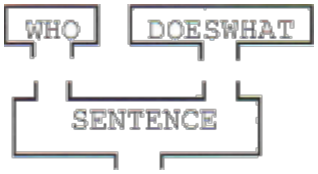

Figure 1.6: Combining the outputs of two sources into a single sentence.

The Logo word OUTPUT is not quite the same as the others. It states that the machine that contains it has a spout, and it indicates what is to come out of that spout. In the x-ray of GOSSIP above, OUTPUT is represented by the wavy arrow.

To summarize, GOSSIP produces and OUTPUTs a SENTENCE composed of an actor (WHO) and an action (DOESWHAT). (SENTENCE is one of many Logo operations that combine their inputs.)

And how do the WHO and DOESWHAT that you created in Etude 1.1 work? Each contains the processor PICK that picks an object randomly from the list given as its input. WHO and DOESWHAT have their lists built into them. WHO's list is [DALE DANA CHRIS SANDY]. (Your own WHO procedure may of course contain a list of different names.)

The square brackets indicate two things. First, they indicate that the words inside them are to be considered together as a single list, one input to PICK. More subtly, the brackets also state that their contents are not to be treated as machines that will generate an input for PICK but are themselves the very objects that PICK may pick among. (Recall that in GOSSIP, SENTENCE's inputs were not the words WHO and DOESWHAT but rather the objects that those machines produced.)

DOESWHAT's list contains four items:

CHEATS.
[LOVES TO WALK.]
[TALKS A MILE A MINUTE]
YELLS.

(Again, you may have chosen different members for this list—perhaps more than four of them.) Like WHO's list, DOESWHAT's list is also enclosed in square brackets, indicating that together these four items form a single list of actions from which to PICK. Two of the items in this list are themselves bracketed because each contains more than one word. Again, the brackets tell Logo that what they enclose is to be considered a single object, a phrase, not several unrelated words.

Figure 1.7 shows an x-ray of DOESWHAT.

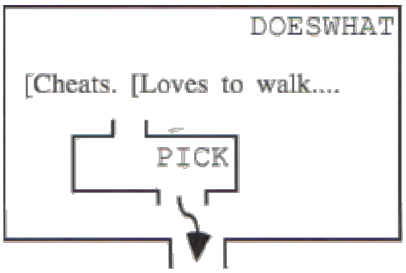

Figure 1 7: An x-ray of DOESWHAT shows that it contains only one machine. The other words in the procedure definition are just text that PICK manipulates for DOESWHAT.

Of course, gossip is often more complex than the kind produced in this model. For one thing, there can be more parts to a sentence of gossip: for example, an object of the subject's attention (or inattention).

Dale cheats DANA.
Dana loves to walk with SANDY.
Chris talks a mile a minute to DANA.
Sandy yells at CHRIS.

We might at first be inclined to code this new model as WHO DOESWHAT TOWHOM, but in our example, as in much gossip, the people giving the attention (the subjects) and the people receiving the attention (the objects) belong to the same clique. So, in describing this model of gossip, it may make sense to use the same code word for the subject and object:

WHO DOESWHAT WHO

A closer look, however, shows that the words expressing the actions represented by DOESWHAT in our model are not exactly the same as they were before. To accommodate the object, wordings like

loves to walk
talks a mile a minute
yells

have to be changed to

loves to walk *with*
talks a mile a minute *to*
yells *at*

Exploration 1.1 Just as the expression (SENTENCE WHO DOESWHAT) represents the combined results of WHO and DOESWHAT, the expression (SENTENCE WHO DOESWHAT WHO) represents a three-constituent sentence. Edit and change your procedures so that you have a procedure that can create sentences like the following:

Sandy loves to walk with Chris
Dale cheats Chris
Chris yells at Dale

(Appendix C contains strategies including full, or full enough, listings of the programming needed in *Explorations*. Sometimes more than one approach is suggested, with some commentary about the relative merits of each one.)

Programming the English Language with the Programming Language Logo

A good model is worth a thousand facts.
M. Malitza (Marx and Tøth, 1984, p. 3)

Practically every discipline has invented its own special notational systems. There is certainly nothing new to linguists about creating diagrams and codes to express abstract relationships in the study of language. (See figure 1.8.)

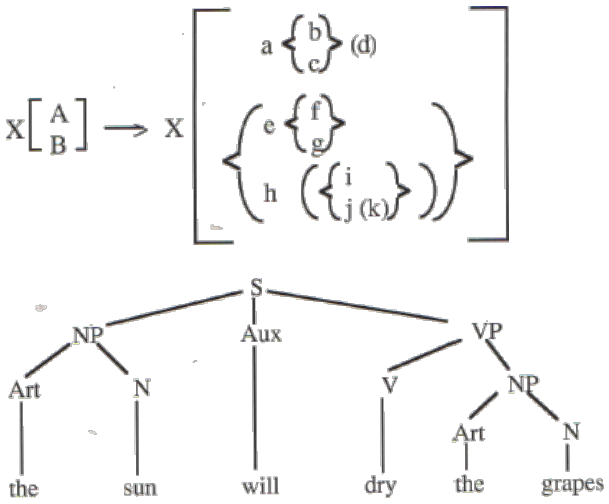

Figure 1.8: Two examples of linguistic and diagrams.
(From Rach, 1964 and from Akmajian, Demers and Hamish, 1979.)

What is relatively new is the easy accessibility of computers to interpret these symbols and allow dynamic experiments with grammars.

For computers to be most useful to the student of language, however, they must not distract from the real goals of studying language. They must enable the expression of linguistic structures and processes in a fairly natural way. Computers that cannot accept symbolic systems that are well suited to such linguistics processing are likely to be just such a distraction.

Logo was designed with the needs of language manipulation in mind, and Logo expressions like (SENTENCE WHO DOESWHAT WHO) are close reflections of the desired linguistic structure.

Parentheses and Brackets

The parentheses tell SENTENCE how many input hoppers it has, how many things it must put together. (See figure 1.9.) In the definition of GOSSIP those inputs are represented by WHO and DOESWHAT. I say "represented by" because, as mentioned earlier, it is not the words *who* and *doeswhat* that are being combined, but the words or phrases that procedures WHO and DOESWHAT produce.

Before SENTENCE can combine its inputs, WHO and DOESWHAT must be executed (in fact, in our current GOSSIP model, WHO must be executed twice) to see what those inputs are.

Figure 1.9: Some processors can be told how many input hoppers to have. Normally, SENTENCE has only two, but when more inputs are parenthesized with it, it "grows" extra hoppers to accommodate the inputs.

Note that parentheses and not brackets are used in the GOSSIP procedure. Recall that beyond showing an intended grouping, brackets also tell Logo that what they contain are inert objects — things, information, data — and not running machines. WHO and DOESWHAT are not inert.

SENTENCE is one of only a few Logo machines that can handle a variable number of input hoppers: two in the case (SENTENCE WHO DOESWHAT) and three in the case (SENTENCE WHO DOESWHAT WHO). That's why it needs the parentheses. You don't have to say (PRINT GOSSIP) to show that PRINT requires one input, nor (WHO) to show that WHO uses no inputs at all. Note that the machine name goes inside the parentheses, along with the inputs. This differs from the *f(x,y)* function notation in algebra.

What Is a Word?

How accurate is it to think of the three structural elements of these sentences as subject, verb, and object? Of course, from the point of view of conventional grammars, the designation "verb" certainly does not include things as complex as *loves to walk with*.

In sentences like *Chris yells at Dale* or *Sandy looks for Dana* or *Dale swims behind Sandy*, schoolbook grammars have traditionally grouped the last two words as a prepositional phrase rather than grouping the middle two words as some kind of complex verb form. But these three cases somehow feel different to many people.

With *yells at*, one can argue either way. The traditional view is that the *at* introduces a prepositional phrase answering where Chris's action is directed, and *yells*, by itself, is that action. However, because one can easily believe that *yells at* has a one-word synonym in many languages — for example, *scolds* in English —

one could plausibly argue that its two-word status in this sentence is more an artifact of word choice than any indication about the underlying grammatical structure of the sentence. In *Sandy looks for Dana*, the two different interpretations are stark: The most natural interpretation of the sentence is *Sandy seeks Dana*, not *In order to help Dana out, Sandy is taking on the task of doing all the looking*. We naturally hear *looks for* as "one verb," not a verb further qualified by a prepositional phrase.

Grammar as a Model of Mind

The argument might go as follows: to have grammatical names and categories just for the sake of having them seems pretty uninteresting,

Grammatical categories matter only when we believe that they are not purely arbitrary, that they really help us describe, for example, something about how brains process language. It seems unlikely that the rules that live in our brains distinguish between *Chris yells at Dale* and *Chris scolds Dale*. A model of mind that treats these two very differently seems unnecessarily complex.

Like *yells at*, *looks for* can feel like a single unit, though some traditional grammars would argue otherwise. It is not difficult even in English to find one-word synonyms for *looks for* and *yells at*, and it is easy to believe that in some languages the most common way of expressing these ideas uses only one word. However, *swims behind* seems too specific to have its own special word in any language except perhaps a Dolphin's language. *Follows* specifies the relationship between the subject and object but doesn't say whether they are swimming, driving, or walking. Swims, by itself, specifies the kind of locomotion but not a relationship with another swimmer. The same distinction can be made between *house boat* and *light switch*, which can feel like single objects (even though there are other kinds of boats and switches), and *green sock*, which generates no such feeling.

The difference is important. For one thing, which is more certain to you: that the correct way of typing *house boat* is not *house-boat* or that the correct way of typing *green sock* is not *green-sock*? What about the correct typing of *greenhouse*?

The choices of list items one makes in procedures such as DOESWHAT (and in the various structural elements used in more complex sentence generators) can become a rich source of material for analyzing what are meaningful categories in a model of mind.

Another simple form of gossip we discussed consisted of sentences like these:

Dale stinks.
Dana is a creep.
Sandy is disgusting.
Chris is rotten to the core.

or their opposites

Sandy is an angel.
Chris is beautiful.

or even a structure composed of these simpler structures

> Chris is beautiful but Chris is rotten to the core.
> Sandy is an angel but Dana is a creep.

The model of this last piece of gossip might be described this way:

> PRAISE [BUT] INSULT

I put the BUT in brackets to show that it is a fixed piece of text, static, unlike the other two constituents, which *stand for* but *are not* the desired text, just as WHO and DOESWHAT *stood for* but *were not* the actual text.

Exploration 1.2 Produce a program that has this behavior:

```
?PRINT INSULT
DALE STINKS
?REPEAT 5 [PRINT INSULT]
CHRIS STINKS
DANA STINKS
```

… and so on.

Look for the simplest way get *only* the one word "stinks" to follow the subject. Don't change your WHO procedure, but let the new INSULT procedure make use of it.

Exploration 1.3 Produce a more varied and colorful insult program.

Exploration 1.4 produce a PRAISE program that has this behavior:

```
?REPEAT 20 [PRINT PRAISE]
DALE IS AN ANGEL
CHRIS IS BEAUTIFUL
```

Again, use the same WHO part that you used in GOSSIP and INSULT.

Exploration 1.5 Use your PRAISE and INSULT procedures to write a program that has this behavior:

```
?REPEAT 3 [PRINT COMPARE]
DALE IS AN ANGEL BUT CHRIS MUGS KITTENS
DANA WALKS ON WATER BUT DANA IS ROTTEN TO THE CORE
CHRIS MAKES MY HEART THROB BUT DALE MAKES MY HEAD THROB
```

Note that each sentence consists of only three parts, a praise, a "but", and an insult. Create the one COMPARE procedure without altering any procedures you created earlier and without creating any other new procedures.