The Joys of Teaching Formal Language Theory to Children

James V. Rauff
Millikin University
jrauff@mail.millikin.edu

Introduction

I have long been of the opinion that formal language theory is one of the best entries to theoretical computer science for elementary school children because it offers the images of graphs, the essential notion of an algorithm, and the fundamentals of deductive proof all in an easily accessible package. In this short note I am pleased to relate some of the joys I experienced while introducing two grade school children to wonders of formal languages.

The Students

My two students, who I will call Adam and Kay, had both just finished the sixth grade. Adam, a boy, was 11 and Kay, a girl, was 12 at the time of the sessions. I wanted to get the two children to the point where they could write simple formal grammars that generated languages with specified properties and to be able to determine whether or not a given word was accepted by a given grammar. I had no plans to get into the Chomsky hierarchy nor to say much, if anything, about automata unless it came up. Both students had worked some with turtle programming in previous sessions with me so that the notion of an algorithm and programming was not foreign to them. Indeed, Adam had already written several programs in BASIC.

Color Tile Alphabets, Rules, and Words

In order to make formal grammars more of a manipulative activity, we used color tiles as the elements of the alphabets for our grammars. The tiles can easily be rearranged to explore derivations and pose rewriting rules. For example, one of the first grammars we explored was the ever-popular regular grammar on $\{R, Y\}$:

$$S \longrightarrow RGY$$
$$G \longrightarrow RGY$$

which generates the language $\{R^nGY^n, n > 0\}$.

To further clarify the processes I wished to teach and to make use of the color tiles I replaced the customary start symbol, S, with the phrase "Always start with." Thus, the rule $S \longrightarrow RGY$ would be given as “Always start with RGY.” The grammar would be given to the students in this form:

Tiles you can use: R, G, Y
Rules:
(1) Always start with RGY
(2) Replace a G with RGY
(3) Stop anytime.

Using the tiles, the students would first lay out a 3-tile sequence *Red-Green-Yellow* and then create new words by replacing the green tile by the same *Red-Green-Yellow* sequence. In this grammar we made no distinctions between terminals and non-terminals. We did make the distinction in other grammars as well as using tile-removal-rules that correspond to ϵ- or λ-productions. For example, the previous grammar could be modified to:

Tiles you can use: R, G, Y
Rules:
(1) Always start with RGY
(2) Replace a G with RGY
(3) Remove a G
(4) Stop only when there are no G's in the pattern.

This grammar generates the language $\{R^nY^n, n > 0\}$. Note that we identify certain tiles as nonterminals by prohibiting them in the final patterns.

Proof, Design, and Joy

After giving the students practice in using the rules to generate patterns I posed questions to them which I hoped would test and extend their abilities. The first group of questions asked them to prove that certain tile patterns could be generated by a given set of rules. Solving these problems required, in essence, deductive proof and the students were splendidly successful. Here's an example.

Given the tile pattern rules:
(1) Always start with RG
(2) Replace a G with YB
(3) Replace a G with RG
(4) Replace a B with RB
(5) Replace a B with R
(6) Stop only when there are no B's or G's in the pattern,

prove that the following patterns can be made with these rules:

$RYRRRRR$
$RRRYRRRRR$
$R^{100}YR^{2500}$.

Kay gave a marvelous proof of the last pattern. She said, "Rule ♯3 always lets us put down as many red tiles as we want at the beginning and then Rule ♯2 puts the yellow down and then Rule ♯4 lets us put as many reds as we want to at the end. *So, we can always get another red at the beginning and another at the end.*" Kay invented an induction proof.

The second group of questions that I posed challenged the students to create grammars that generated particular languages. Again the students rose to the challenge. They handled several starter languages with ease and then looked at some context-sensitive rules. I then gave them what I thought would be a real difficult challenge. Working together they came up with a grammar that produced a tile version of the copy language on two letters. Their grammar read like this (the colors are *Red*, *Pink*, *Blue*, *Orange*, *Yellow*, *Green*, *White*, and *bLack*):

(1) Always start with $OWGB$
(2) Replace OWG with $ROWGR$
(3) Replace OWG with $YOWGY$
(4) Replace WGR with LGP
(5) Replace WGY with LGO
(6) Replace LGR with LGP
(7) Replace LGY with LGO
(8) Replace PR with RP
(9) Replace PY with YP
(10) Replace PB withBR
(11) Replace OR with RO
(12) Replace OY with YO
(13) Replace OB with BY
(14) Remove $OLGB$
(15) Stop when you only have R's and Y's in the pattern.

This was indeed a fine piece of work and I was ecstatic with the result. Even more joyful to me were their explanations. Adam wrote, "We couldn't figure out how to make the back end copy the front because it always came out backwards. So, we decided to just take the backwards copy and turn it around." Kay wrote, "The pink tiles kind of remember that you're moving a red tile to the back and the orange ones remember yellow. That way we didn't mix up the reds and yellows we hadn't moved yet with those we were moving." This was great!

Reflections

I must admit that Adam and Kay achieved much more in formal language theory than I expected. In addition to the tasks I posed to them, Adam and Kay also invented some of their own formal language theory along the way. They wrote a grammar that required the rules to be used in a particular order (a programmed grammar) and they pointed out that you could get some "really cool patterns" by choosing rules by flipping a coin or throwing a die (stochastic grammars). Adam suggested that we could draw a pictures of the rules using "lines with arrows" to help us see what was going on with the grammars. He, in effect, came up with the notion of a finite state automaton.

Their persistance in manipulating the tiles and writing the rules to achieve and prove patterns was inspiring. I have never enjoyed teaching formal languages more. Their excitement was contagious and reminded me once again that we should never assume that any area of mathematics is "too advance" for children.