15-312 Foundations of Programming Languages
Recitation 2: Rule Induction

Daniel Spoonhower
spoons+@cs

September 3, 2003

1 Matching Parentheses

Recall from lecture our original definition (through inference rules) of the language of matching parentheses.

$$\frac{}{\varepsilon\ M}\ M_1 \qquad \frac{s_1\ M \qquad s_2\ M}{s_1\, s_2\ M}\ M_2 \qquad \frac{s\ M}{(s)\ M}\ M_3$$

Recall also our "parser" for this language, given in terms in the following judgment and inference rules.

$$\frac{}{0 \rhd \varepsilon}\ \rhd_1 \qquad \frac{k+1 \rhd s}{k \rhd (s}\ \rhd_2 \qquad \frac{k-1 \rhd s}{k \rhd)s}\ \rhd_3\ (k > 1)$$

We would like to show that the languages defined by M and $\rhd$ are one and the same, and we began in lecture with a proof of the following: if $s\ M$ then $0 \rhd s$.

We will continue today be showing inclusion in the opposite direction. In particular, that

Theorem 1. *If* $0 \rhd s$ *then* $s\ M$.

Proof. By rule induction on the derivation of $0 \rhd s$. We consider each case in turn.

(Rule $\rhd_1$) Then $s = \varepsilon$.

$s\ M$ — By M_1

(Rule $\rhd_2$) Then $s = (s'$.

$k + 1 \rhd s'$ — Subderivation

?

What's gone wrong? Normally, this is the point of the proof where we'd cleverly apply the induction hypothesis – why can we not do so in this case? What, according to a recent lecture, are our alternatives if we find ourselves stuck in such a situation?

The first answer here is to generalize the induction hypothesis. To claim the equivalence of the languages M and $\rhd$, our theorem is strong enough, but it is not strong enough for us to carry out our proof. Let's try it again.

Theorem 1 (Revised). *If* $k \rhd s$ *then* $\underbrace{(\cdots(}_{k} s\ M$.

Proof. By rule induction on the derivation of $k \rhd s$. We consider each case in turn.

(Rule $\rhd_1$) (as above)

(Rule $\rhd_2$) Then $s = (s'$.

$k+1 \rhd s'$	Subderivation
$\underbrace{(\cdots(}_{k+1} s'\ M$	By i.h.
$\underbrace{(\cdots(}_{k} s\ M$	Since $s = (s'$.

(Rule $\rhd_3$) Then $s =)s'$ and $k > 1$.

$k-1 \rhd s'$	Subderivation
$\underbrace{(\cdots(}_{k-1} s'\ M$	By i.h.
$()\ M$	By M_1, M_3
$\underbrace{(\cdots(}_{k-1} ()s'\ M$	By ???

We're so close this time! We'd like to conclude $\underbrace{(\cdots(}_{k-1} ()s'\ M$ (equivalently $\underbrace{(\cdots()}_{k} s'\ M)$, but we're not quite there yet. Intuitively, this should work out: we should be able to add a pair of (balanced) parentheses anywhere within a string of whose parentheses are already matched. (Are you convinced? Try some examples.) We'll use another strategy from lecture: we'll prove a lemma! To keep the syntax under control, I'll use l, r, and c instead of just s to stand for strings of parentheses.

Lemma 2. *If* $l\,r\ M$ *and* $c\ M$ *then* $l\,c\,r\ M$.

(Before you read on, think about how we will go about proving this? By induction? Over what?)

Proof. By rule induction on the derivation of $l\,r\ M$. We consider each case in turn.

(Rule M_1) Then $l\,r = \varepsilon$.

$c\ M$	By assumption
$l\,c\,r\ M$	Since $l = r = \varepsilon$

(Rule M_2)

One might think that the derivation of $l\,r\ M$ looks something like this:

$$\dfrac{\begin{array}{cc}\vdots & \vdots \\ l\ M & r\ M\end{array}}{l\,r\ M}$$

Why is this not the case? Just because we have chosen to break our string into two parts l and r doesn't mean that they each have matching parentheses. (Think about where we'd like to use this lemma and about a statement of the form $l \rhd r$. Must l (in particular) and r have matching parentheses?)

To complete this case, we must consider a number of subcases, one for each way that $l\,r$ might be broken down into two strings of matching parentheses. First we take the case where l is split.

(Rule M_2, Subcase 1) Let $l = l_1\,l_2$.

$$\dfrac{\begin{array}{cc}\vdots & \vdots \\ l_1\ M & l_2\,r\ M\end{array}}{l_1\,l_2\,r\ M}$$

$l_2\,c\,r\ M$	By i.h.
$l_1\,l_2\,c\,r\ M$	By M_2
$l\,c\,r\ M$	Since $l = l_1\,l_2$

(Rule M_2, Subcase 2) Let $r = r_1\,r_2$.

$$\dfrac{\begin{array}{cc}\vdots & \vdots \\ l\,r_1\ M & r_2\ M\end{array}}{l\,r_1\,r_2\ M}$$

(as above)

(What if the split really was between l and r? Do we need a separate case for this?)

(Rule M_3)

Again, we must consider each of the ways that $l\,r$ might be split in a derivation that ends with

$$\frac{\vdots \\ s\ M}{(s)\ M}$$

(Rule M_3, Subcase 1) Let $l = (l'$ and $r = r')$.

$$\frac{\vdots \\ l'\,r'\ M}{(l'\,r')\ M}$$

$l'\,r'\ M$	Subderivation
$l'\,c\,r'\ M$	By i.h.
$(l'\,c\,r')M$	By M_3
$l\,c\,r\ M$	Since $l = (l'$ and $r = r')$

(Rule M_3, Subcase 2) Let $l = \varepsilon$ and $r = (r')$.

$l\,r\ M$	By assumption
$r\ M$	Since $l = \varepsilon$
$c\,r\ M$	By M_2
$l\,c\,r\ M$	Since $l = \varepsilon$

(Rule M_3, Subcase 3) Let $l = (l')$ and $r = \varepsilon$. (as above) □

Given this lemma, we can now return to our main theorem. In fact, we now have all the right tools to complete the proof: the last case goes through easily using our new lemma.

1.1 Alternatives to Rule Induction?

We have focused this time on *rule* induction, but there are other properties of strings that we might reason about. In many cases, we might want to carry out some proof by reasoning inductively over the *lengths* of strings. (Quick: think of a handful from 212!) Reconsider the case from our lemma where we split l into two pieces l_1 and l_2. The end of the derivation looked something like this:

$$\frac{\begin{matrix}\vdots & \vdots \\ l_1\ M & l_2\,r\ M\end{matrix}}{l_1\,l_2\,r\ M}$$

What if $l_1 = l_2 = \varepsilon$? Then $l_2\, r$ is not any shorter than $l_1\, l_2\, r$! If we were to reason about the lengths of the strings in this case, we could *not* apply the induction hypothesis. Here (and in many proofs in this class) rule induction will prove to be the better choice.