Term Rewrite Systems

1 What is a rewrite system?

A rewrite system has two features. Firstly, a set of terms. Secondly a relation from terms to terms. That's it. We typically notate them in the following way. Consider a set of terms $\{\mathbf{A}, \mathbf{B}, \mathbf{C}\}^*$. That is, finite sequences of A, B, and C. A simple rule set, which we'll call Simple.

$\mathbf{AB} \to \mathbf{C}$

These mean that there's for any term x where **AB** appears as a subsequence, x is related to x with **AB** replaced by **C**.

1.1 Normal Forms

This system has several features we should already discuss. We will often use graph terminology here, where each node is a term, and each rule adds some edges. Note the edges are directed. Consider the term **C**. This term has 0 outdegree, in other words, no rules apply to it. We call such a term a *normal form*. We also define the $\to^*$ relation which is the transitive, reflexive closure. That is, paths through this graph. In general, we say that **CC** is a normal form of **ABAB** because $\mathbf{ABAB} \to^* \mathbf{CC}$

For an example of a term that doesn't have a normal form, lets introduce a new rulset which we will call Expander

$\mathbf{A} \to \mathbf{AA}$

Here, we can see that any term with an **A** can always have more rules applied. A yet stronger property we sometimes desire is that every path from x eventually reaches a normal form. We say x is *strongly normalizing* or *terminating*.

1.2 Confluence

Sometimes, however, we get choices. Consider the system Brancher

$\mathbf{A} \to \mathbf{B}$

$\mathbf{A} \to \mathbf{C}$

The term **A** has two rules that apply to it. Once with pick $\mathbf{A} \to \mathbf{B}$, we are stuck with that choice forever.

On the other hand, consider the Simple system, and the term **ABAB**. There is a choice of what rule we apply

$$\mathbf{ABAB} \to \mathbf{ABC} \to \mathbf{CC}$$

But also

$$\mathbf{ABAB} \to \mathbf{CAB} \to \mathbf{CC}$$

This is a slightly more favorable system. Even though we have choices, the choices don't matter. This is is called confluence. Formally, a term a is weakly confluence if we have

$$\forall b, c,\ a \to^* b \wedge a \to^* c \Rightarrow \exists d,\ b \to^* d \wedge c \to^* d$$

That is, if you ever make a choice, you can always get back to the same point eventually.

The Simple system has a somewhat stronger property than that, though. That is,

$$\forall b, c,\ a \to b \wedge a \to c \Rightarrow \exists d,\ b \to^* d \wedge (c \to d \vee c = d)$$

in other words, if we deviate by one step, then things are fixable in one step (or zero).

1.3 Confluence and Normalizing

Confluence is nice, but it doesn't guarantee that you know how to pick that d. In fact, deciding what the correct "evaluation strategy" AKA path through the graph, is impossible in general. But sometimes we are in a wonderful setting where every term has a normal form, and every term is confluence, normal forms are unique!

To see that, consider a term x, first observe that normal forms are unique. If y and z are normal forms, then

$$x \to^* y \text{ and } x \to^* z$$

But there are no out edges, so for there to be a d where

$$y \to^* d \text{ and } z \to^* d$$

both paths must be length 0. In other words $y = d = z$.

2 Who cares?

Confluence is an nice setting because evaluations either run forever, or stop at a unique value. In other words, there is a function from normalizing terms to their normal form.

From a language design perspective, this is extremely desirable because it means your programs are "defined".

Many important rewrite systems fail this:

1. Compiler optimizations are rewrite systems but the order of passes changes the final program! No obvious way to run to a fixpoint
2. Macros are basically rewrite systems, but often order of application matters.

It does not, however, give you a computational strategy. Consider the system Mixed

$\mathbf{A} \to \mathbf{B}$

$\mathbf{B} \to \mathbf{A}$

$\mathbf{A} \to \mathbf{C}$

There is an infinite sequence of rewrites you could run

$$\mathbf{A} \to \mathbf{B} \to \mathbf{A} \to ...$$

Or you could do

$$\mathbf{A} \to \mathbf{C}$$

Or even

$$\mathbf{A} \to \mathbf{B} \to \mathbf{A} \to \mathbf{C}$$

So even though there is a normal form, it's not always clear how to reach it.

Now, if our system is both strongly normalizing and weakly confluent, we have the best case.

1. every term has a normal form
2. that normal form is unique
3. every evaluation strategy eventually ends at that normal form

Naturally, a system with these properties is restricted in important ways. It can't be turing complete, after all. So finally we can start down our intended path. We are building a framework to make tradeoffs between expressibility, and analyzability in general terms.