The Unique Decipherability Problem

Copyright 2010 Dan Gusfield

Basic Definitions: A *code* C is just a finite set of finite length strings, called *codewords*. A *message* M is a string created by concatenating codewords from C, with repetitions allowed. So there is no bound on the length of a message. A message M is called *uniquely decipherable (UD)* if there is only one way to write M as the concatenation of strings in C. Writing M as the concatenation of codewords in C is called a *parse* of M. For example, if $C = \{aba, a, abb, ab, ba\}$ then the message $abbab$ is UD, while the message aba is not UD. A code C is called UD if and only if *every* message that can be created from C is UD. So, the code C in the example is not UD, but if we remove the strings ba and ab from C, the resulting code would be UD. Note that the definition of a code being UD is over an infinite set of messages.

The **UD problem**: Given a code C, determine if C is UD.

At first, it may seem that the UD problem would be undecidable (it seems similar to the famous undecidable Post Correspondence Problem). But, in fact it has a polynomial-time solution, and it is even conjectured to have a linear-time solution. There are several algorithms for the UD problem (and variants of it), but they are all similar and can be thought of as a path problem on a graph.

1 A graph-based solution

Given C, we build a graph G with one node for each string that occurs as a suffix of any codeword in C, including a node for each complete codeword (the trivial suffix). Note that the same string might occur as a suffix of more than one codeword, but it is only represented by a single node in G. We label each node by the suffix (string) it represents. In graph G, there is a directed edge from node u to node v if there is a codeword c such that $c = uv$ (this is an L1, or Type 1, edge), or if $u = cv$ (this is an L2, or Type 2, edge).

Theorem: C is not UD if and only if there is a directed path in G with at least one edge, starting at a node representing a codeword, and ending at a node representing a codeword. The start and end nodes can be the same, but don't have to be.

Proof: First we will prove that if the code is not UD, then there is such a directed path in G. Assuming C is not UD, there must be a message M that can be parsed in two different ways. Over all such M, choose one that is shortest. Let c_1 and c_1' denote the first codewords in the two parses of M. Note that those two codewords cannot have the same length, or else they would be identical, and if we removed that codeword from the head of M, we would have shorter message that is also not UD, a contradiction.

We will show the existence of the claimed directed path P in G by building it constructively while examining the two different parses of M, going left to right in M, adding one additional codeword at a time. We examine the parses starting with the empty parse (no codewords added yet) and then add in the longer of the two codewords, say c_1'. At that point, one parse has length $|c_1'| > 0$ and the other has length 0. Naturally, one parse is called the "longer" parse, and the other is called the "shorter" parse. The substring consisting of the part of M contained in the longer parse, but not contained in the shorter parse, is called the "overhang". So, at this point, the overhang consists of c_1'.

In general, we extend the examination of the two parses by adding in the next codeword, call it cc, of the (currently) *shorter* parse (see Figure 1). There are three cases to consider: either the length of cc is strictly greater than the length of the current overhang; or the length of cc is strictly less than the length of the current overhang; or cc and the current overhang have the same length. See Figure 1. In the first case, the addition of cc extends the currently shorter parse to the right of the end of the currently longer parse (in which case the extended parse becomes the longer parse); in the second case, it only extends the shorter parse to a point strictly to the left of the currently longer parse (in which case the extended parse remains the shorter parse); or it extends the parse exactly to the end of the longer parse. In the third case, the two parses each spell out the same complete string with two different parses, and so that string must be M. Otherwise, M was not a shortest string with two different parses. Hence, this third case can only happen once in the examination of the two parses. In either of the first two cases, the overhang changes. The overhangs before and after the addition of cc will be called the "old" and "new" overhangs. It is easy to see (established more formally by induction, below) that each overhang generated in this way is a suffix of a codeword, possibly an entire codeword.

Now we want to describe how the desired path P is built up as the two

parses of M are being examined. Codeword c'_1 is the first codeword added into the parses, so the first overhang is just c'_1 itself. Note, trivially, that c'_1 is a suffix of a codeword. Next, codeword c_1 is added into the parses, since it is the first codeword of the shorter parse (which has length zero before c_1 is added in). Now $|c'_1| > |c_1|$, so c_1 is a prefix of c'_1; let v denote the *suffix* of c'_1 starting at position $|c_1| + 1$ of c'_1. Therefore $c'_1 = c_1 v$. But c_1 is a suffix of a codeword (namely, itself), and c'_1 is a codeword, so c'_1 and c_1 demonstrate the existence of a Type 1 edge in G from the node representing c_1 to the node representing suffix v. So path P begins with an edge from the node for codeword c'_1 to the node for suffix v, which is the new overhang at this point. Note that the new overhang, v, is the suffix of a codeword.

At the general step, codeword cc is added into the shorter parse, changing the overhang. We use u to denote the current overhang before cc is added. We assert, inductively, that u is the suffix of a codeword, and that there is a directed path in G from the node for c'_1 to the node for u. We will see that the new overhang, call it v, is also a suffix of a codeword, and that it demonstrates the existence in G of an edge from the node for suffix u to the node for suffix v. There are three cases, depending on the lengths of cc and u.

Case 1: If $|cc| > |u|$, then $cc = uv$, establishing the existence of the directed, Type 1, edge (u, v) in G. Note that again the new overhang, v, is a suffix of a codeword, e.g. cc. This also establishes the existence of a directed path from the node for c'_1 to the node for v.

Case 2: If $|cc| < |u|$, then $u = ccv$, establishing the existence of the directed, Type 2, edge (u, v) in G. It also establishes that the new overhang, v, is a suffix of a codeword, because inductively u is a suffix of a codeword. This also establishes the existence of a directed path from the node for c'_1 to the node for v.

Case 3: If $|cc| = |u|$, then the two parses of M are complete, and this case can happen only once. So up to this point, all of the other additions are instances of Case 1 or Case 2, and since u is the current overhang (before cc is added), inductively, there is a directed path from the node for c'_1 to the node for u. But $|cc| = |u|$ and both are suffices of M, so $cc = u$, and hence u is a codeword. Therefore the path from c'_1 to u is the path P, claimed to exist in the statement of the theorem.

This proves one direction of the theorem. The proof of the other direction

is very similar. We assume the existence of a directed path P, of length at least one, that goes from a node for a codeword to a node for a codeword, and use P to form a message M and two different parses of M. We leave the details as an exercise.

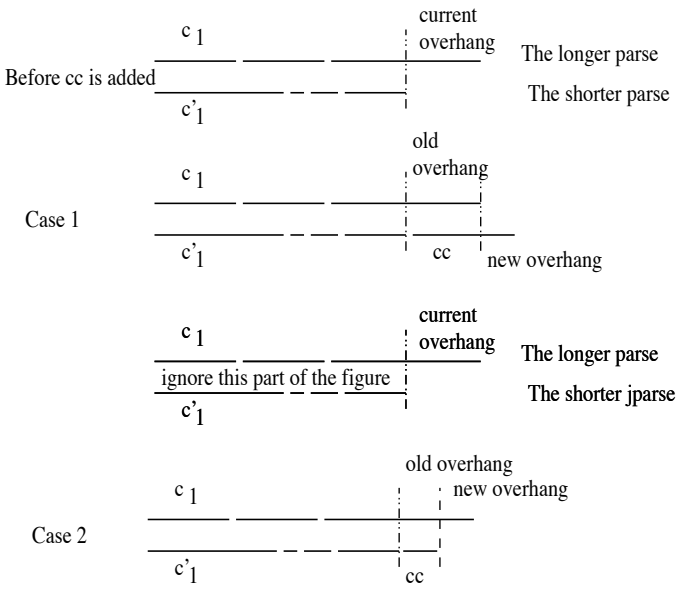

Figure 1: Examining the two parses of M, and adding in the new codeword cc to the shorter parse. The first two cases. Each horizontal line segment represents a codeword. There is some garbage in this figure that I couldn't remove, and I was too lazy to redo the figure.