Inferring Graphs from Walks

(Extended Abstract)

Javed A. Aslam* Ronald L. Rivest†
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

Abstract

We consider the problem of inferring an undirected, degree-bounded, edge-colored graph from the sequence of edge colors seen in a walk of that graph. This problem can be viewed as reconstructing the *structure* of a Markov chain from its output. (That is, we are not concerned with inferring the transition probabilities, but only the underlying graph structure of the Markov chain.) We present polynomial-time algorithms for the inference of underlying graphs of degree-bound 2 (linear chains and cycles), based on some surprising properties about the confluence of various sets of rewrite rules.

1 Introduction

Consider an undirected, edge-colored graph $G = (V, E, c)$ with vertex set V, edge set E, and edge coloring $c : E \mapsto \Sigma$. A *walk* of G starts at some vertex v_i and makes transitions from vertex to vertex by arbitrarily selecting some edge incident on the current vertex and traversing it. The output of such a walk is the sequence of colors of the edges traversed.

We ask: given the output of a walk and a degree-bound k, what is the smallest undirected, degree-bound k, edge-colored graph G consistent with this output? (A graph has degree-bound k if every vertex has degree at most k.)

For a particular output sequence, there may be many graphs G of varying degree-bounds that are consistent with this output. For example, the output sequence `abccbbcdeffe` could have been produced by a walk of any of the graphs in Figure 1. For a given output sequence and degree-bound k, we wish to find the smallest degree-bound k graph that is consistent with the output. We call this *minimum consistent inference*. In Figure 1, (a) is the minimum

We gratefully acknowledge support from NSF (grant CCR-8914428), ARO (grant DAAL03-86-K-0170), and the Siemens Corporation.

*Author's net address: jaa@theory.lcs.mit.edu
†Author's net address: rivest@theory.lcs.mit.edu

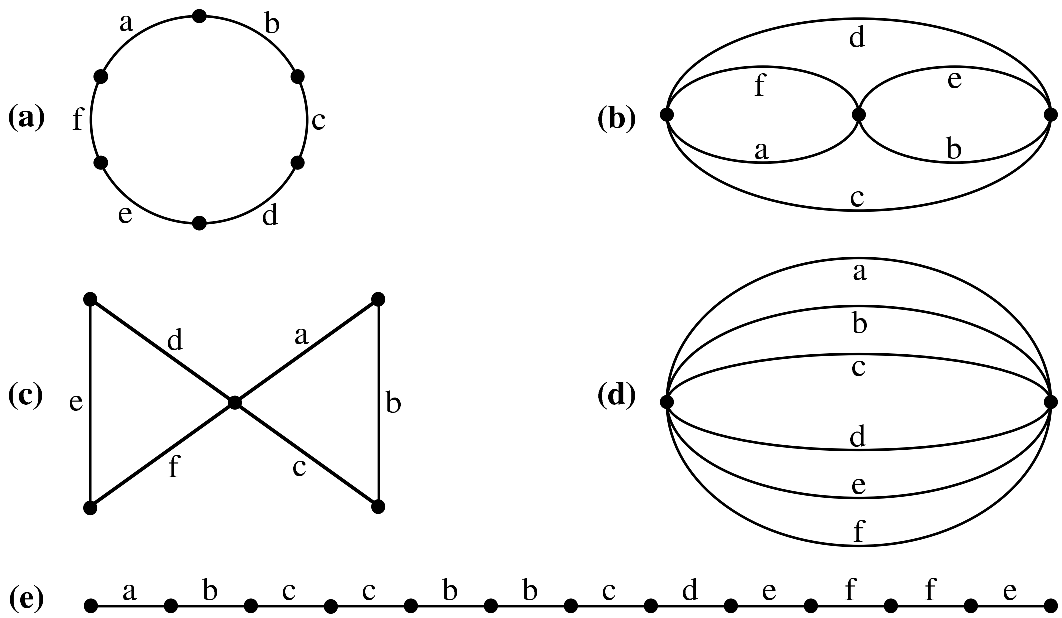

Figure 1: Graphs consistent with the walk `abccbbcdeffe`.

consistent degree-bound 2 graph, (b) is the minimum consistent degree-bound 4 graph, and (d) is the minimum consistent degree-bound 6 graph.

In this paper, we give polynomial-time algorithms for the minimum consistent inference of degree-bound 2 graphs. It is an open question whether the minimum consistent inference of degree-bound k graphs for $k > 2$ is tractable or NP-complete.

This work is related to other work in the computer science literature. Rudich [9] has considered the problem of inferring Markov chains from their output. He considers a restricted class of Markov chains whose output is binary. Given the binary output of a Markov chain, Rudich gives algorithms that, in the limit, reconstruct the underlying Markov chain structure as well as the associated transition probabilities. Our focus is, however, on the *efficient* (i.e. polynomial time) inference of the underlying graph structure, and not on the inference of the associated transition probabilities.

Angluin [3] and Gold [6] have considered the problem of automaton identification from input/output behavior. Given the input/output behavior of an automaton, one would like to construct the directed graph corresponding to the state transition diagram of the smallest automaton compatible with the given data. Angluin and Gold have shown that this particular problem is NP-complete. Our work is different in that it is concerned with the class of *undirected* underlying graphs.

Section 2 provides some intuition for our method of inferring degree-bound 2 graphs from walks. Section 3 then supplies some necessary definitions regarding sets of rewrite rules. Section 4 shows how rewrite rules can be used to find minimum linear chains consistent with a given walk. Section 5 shows how to extend this result to infer cycles. Finally, Section 6 sketches the proofs of some key lemmas used in the correctness proofs (complete proofs will be given in the full paper).

2 Intuition

When we have a degree-bound of 2, we are restricted to inferring linear chains and cycles. It is convenient to identify a chain with the sequence of edge colors seen in a end-to-end traversal of the chain. Similarly, a cycle can be identified with any of the edge-color sequences seen in a single loop around the cycle.

Consider the output sequence $y = \texttt{abccbbcdeffe}$. Clearly, the minimum consistent chain for this example is identified by the sequence $z = \texttt{abcdef}$. How do we get from y to z? The subsequence `bccbbc` is of the form xx^Rx for $x = \texttt{bc}$, and can therefore be obtained by traversing `bc` in the forward direction, then in the bacward direction, and then again in the forward direction. By replacing `bccbbc` with the shorter sequence `bc` we obtain a shorter string that represents a chain that can be walked to obtain the original sequence. We can also replace the terminal subsequence `effe` in y with `ef`. Since `effe` is a final subsequence, the resulting shorter sequence can still be walked to obtain the original sequence.

This gives us a *replacement* strategy where substrings of the form xx^Rx are replaced by x, and suffixes of the form xx^R are replaced by x. What about more difficult cases where the xx^Rx patterns are nested inside one another? We can imagine a reasonably efficient strategy where the patterns are replaced, say, in order of smallest to largest. But what guarantees that the resulting string is, in fact, consistent and smallest? Questions similar to this have been studied in the context of eliminating redundant steps in computer programs [1, 2] and evaluating expressions in the Lambda calculus [4, 5]. In these contexts, properties of a particular replacement system are proved that guarantee "consistent" and "smallest". In the following section, we present the definitions, notation and basic results of replacement systems. Following this, we develop replacement systems to solve our particular problem, employing the results given in the next section.

3 Preliminaries

Let $\mathcal{S}$ be an arbitrary set. A *binary relation* on $\mathcal{S}$ is a subset of $\mathcal{S} \times \mathcal{S}$. Let $\rightarrow$ be a binary relation on $\mathcal{S}$. If $(x, y) \in \rightarrow$, we write $x \rightarrow y$. The following definitions and notation have been adapted from Huet [8], Huet and Oppen [7] and Aho, Sethi and Ullman [1]:

Let $\imath$ be the *identity relation* on $\mathcal{S}$, $\imath = \{(x, x) \mid x \in \mathcal{S}\}$. Given two binary relations $\rightarrow_{\text{A}}$ and $\rightarrow_{\text{B}}$, we define their *composition* as $\rightarrow_{\text{A}} \cdot \rightarrow_{\text{B}} = \{(x, y) \mid \exists z,\, x \rightarrow_{\text{A}} z \;\&\; z \rightarrow_{\text{B}} y\}$. An element $x \in \mathcal{S}$ is *irreducible* if $\nexists y \in \mathcal{S}$ such that $x \rightarrow y$. Inductively, we define the following:

$$
\begin{aligned}
\xrightarrow{0} &= \imath \\
\xrightarrow{\epsilon} &= \rightarrow \cup\, \imath \\
\xrightarrow{i} &= \rightarrow \cdot \xrightarrow{i-1} \ \forall i > 0 \\
\xrightarrow{+} &= \bigcup_{i>0} \xrightarrow{i} \\
\xrightarrow{*} &= \xrightarrow{+} \cup\, \imath \\
\xrightarrow{\hat{}} &= \{(x,y) \mid x \xrightarrow{*} y \ \& \ y \text{ is irreducible}\}
\end{aligned}
$$

If $x \xrightarrow{*} y$ and y is irreducible, then y is called the $\rightarrow$ *normal form* of x. We shall denote the normal form of x by $\hat{x}$.

Definition 1 *A binary relation* $\rightarrow$ *is* noetherian *if and only if there is no infinite sequence* $x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_n \rightarrow \ldots$

Definition 2 *A binary relation* $\rightarrow$ *is* confluent *if and only if* $\forall w, x, y \ w \xrightarrow{*} x \ \& \ w \xrightarrow{*} y \Rightarrow \exists z, \ x \xrightarrow{*} z \ \& \ y \xrightarrow{*} z.$

Definition 3 *A binary relation* $\rightarrow$ *is* locally confluent *if and only if* $\forall w, x, y \ w \rightarrow x \ \& \ w \rightarrow y \Rightarrow \exists z, \ x \xrightarrow{*} z \ \& \ y \xrightarrow{*} z.$

It should be noted that the confluence property is equivalent to the widely known Church-Rosser property. The following two results are well known and can be found in [8]:

Theorem 1 *A noetherian relation is confluent if and only if it is locally confluent.*

Theorem 2 *If a binary relation is confluent, then the normal form of any element, if it exists, is unique.*

We are concerned with a specific subclass of noetherian relations defined as follows.

Definition 4 *A binary relation* $\rightarrow$ *is* strictly decreasing *if* $\forall x, y \ x \rightarrow y \Rightarrow |y| < |x|.$

Clearly, every strictly decreasing relation is noetherian. We can now show the following:

Theorem 3 *If* $\rightarrow$ *is confluent and strictly decreasing, then* $\forall x, y \ x \xrightarrow{*} y \Rightarrow |y| \geq |\hat{x}|.$

Proof: By contradiction, assume that $\exists y$ such that $x \xrightarrow{*} y$ and $|y| < |\hat{x}|$. Clearly, $x \xrightarrow{*} \hat{x}$. By confluence, $\exists z$ such that $y \xrightarrow{*} z$ and $\hat{x} \xrightarrow{*} z$. Since $\hat{x}$ is irreducible, $z = \hat{x}$. Therefore $y \xrightarrow{*} \hat{x}$, but this is not possible since $|y| < |\hat{x}|$ and $\rightarrow$ is strictly decreasing. ∎

4 Inference of Linear Chains

In this section we examine the inference of linear chains. We define a *linear chain* as degree-bound 2, undirected, edge-colored graph $G = (V, E, c)$ with vertices $V = \{v_1, v_2, \ldots, v_N\}$, edges $E = \{(v_1, v_2), (v_2, v_3), \ldots, (v_{N-1}, v_N)\}$ and edge colors $c(u, v)\ \forall (u, v) \in E$. If we let Σ be the set of all colors, then we may denote a linear chain by a string $z \in \Sigma^+$ corresponding to the sequence of edge colors $c(v_1, v_2), c(v_2, v_3), \ldots, c(v_{N-1}, v_N)$.

4.1 Inference of Linear Chains from End-to-End Walks

An *end-to-end walk* on a linear chain z is a sequence of colors corresponding to the edges traversed in some walk on z that begins at the left vertex v_1 and ends at the right vertex v_N. If we let Σ be the set of letters, then $z =$ `abcda` is a linear chain, and $y_1 =$ `abcccda` and $y_2 =$ `abccbbcda` are both end-to-end walks on z. The string y is an end-to-end walk on z if and only if it *embeds* or folds into z. The embedding of y in z partitions y into *segments* corresponding to substrings between the folds in y necessary to embed y in z. For $y =$ `abccbbcda` and $z =$ `abcda`, the embedding of y in z partitions y as follows: `abc`, `cb`, `bcda`.

For the set Σ^+, we define a binary relation $\rightarrow_{\mathrm{B}} = \{(pxx^Rxq, pxq) \mid p, q \in \Sigma^*, x \in \Sigma^+\}$ where x^R is the reverse of the string x. If $y \xrightarrow{*}_{\mathrm{B}} z$, then z is called a *B-contraction* of y. (The letter "B" is used because this relation affects the *body* of the string.) Conversely, y is called a *B-expansion* of z. Clearly, $\rightarrow_{\mathrm{B}}$ is strictly decreasing since $\forall y, z \in \Sigma^+$, if $y \rightarrow_{\mathrm{B}} z$ then $|z| < |y|$.

Lemma 1 *For all strings $y, z \in \Sigma^+$, y is an end-to-end walk on the linear chain $z \iff y$ is a B-expansion of z.*

Proof ($\Rightarrow$): By contradiction, assume that the $\Rightarrow$ implication is false. Then there must exist some shortest counterexample y such that y is an end-to-end walk on some linear chain z, and y is not a B-expansion of z. Let s be the shortest segment of y in the embedding of y in z. Clearly, the segment preceding s must end in s^R and the segment following s must begin with s^R. Thus, y may be written as pxx^Rxq for some $p, q \in \Sigma^*$ and $x = s^R$. Further, pxq is also an end-to-end walk on z. If $pxq \xrightarrow{*}_{\mathrm{B}} z$, we then have $y = pxx^Rxq \rightarrow_{\mathrm{B}} pxq \xrightarrow{*}_{\mathrm{B}} z$ which implies that y is a B-expansion of z, a contradiction. If pxq is not a B-expansion of z, then pxq is a shorter couterexample than y, also a contradiction. ∎

Proof ($\Leftarrow$): By definition, if y is a B-expansion of z then $y \xrightarrow{*}_{\mathrm{B}} z$. Since $\rightarrow_{\mathrm{B}}$ is strictly decreasing, we may write $y = w_n \rightarrow_{\mathrm{B}} w_{n-1} \rightarrow_{\mathrm{B}} \ldots \rightarrow_{\mathrm{B}} w_2 \rightarrow_{\mathrm{B}} w_1 = z$ for some finite n. We show that w_i is an end-to-end walk on z by induction on i. Clearly, w_1 is a (trivial) end-to-end walk on z. Assume w_{i-1} is an end-to-end walk on z. Since $w_i \rightarrow_{\mathrm{B}} w_{i-1}$, we have $w_i = pxx^Rxq$ and $w_{i-1} = pxq$ for some $p, q \in \Sigma^*$, $x \in \Sigma^+$. The string w_i may be embedded in z as follows: embed px in the same manner that px is embedded for w_{i-1}, embed x^R by tracing the embedding of x backwards, and finally embed xq in the same manner as xq is

```
Input:
  y - string of symbols
  n - length of y
Output:
  z - string of symbols corresponding to ŷ
  m - length of z
Procedure:
  m = 0;
  for i = 1 to n
    m = m + 1;
    z[m] = y[i];
    if z has a suffix of the form xx^R x then
      l = length of the xx^R x suffix;
      m = m - (2/3)l;
    endif
  end
```

Figure 2: Algorithm for obtaining the B-normal-form of y.

embedded for w_{i-1}. Therefore w_i is an end-to-end walk on z, and by induction $y = w_n$ is an end-to-end walk on z. ∎

Lemma 2 *The binary relation $\rightarrow_{\mathrm{B}}$ is confluent.*

Since $\rightarrow_{\mathrm{B}}$ is strictly decreasing, we need only show that $\rightarrow_{\mathrm{B}}$ is locally confluent. This proof is given in Section 6.1.

Theorem 4 *If y is an end-to-end walk, then the shortest linear chain that can produce y is $\hat{y}$, the B-normal-form of y.*

Proof: Since $y \xrightarrow{*}_{\mathrm{B}} \hat{y}$, y is an end-to-end walk on $\hat{y}$ by Lemma 1. Assume that y is an end-to-end walk on some z and $|z| < |\hat{y}|$. By Lemma 1, $y \xrightarrow{*}_{\mathrm{B}} z$. Now, Theorem 3 implies that $|z| \geq |\hat{y}|$, a contradiction. ∎

Thus, in order to obtain the minimum consistent linear chain that can produce an end-to-end walk y, we must find $\hat{y}$. Since $\rightarrow_{\mathrm{B}}$ is confluent, $\hat{y}$ may be obtained by repeatedly replacing substrings of the form xx^Rx in y by x (by Theorem 2). Since $\rightarrow_{\mathrm{B}}$ is strictly decreasing, this process is guaranteed to terminate.

Claim 1 *On input y, the algorithm in Figure 2 produces $\hat{y}$, the B-normal-form of y.*

Proof: Let z_i be the string z after the ith iteration of the **for** loop. Clearly, $z_1 = y[1]$ and $z_{i-1}y[i] \xrightarrow{\epsilon}_{\mathrm{B}} z_i \ \forall i > 1$. Thus $y = z_1y[2]y[3]\cdots y[n] \xrightarrow{\epsilon}_{\mathrm{B}} z_2y[3]y[4]\cdots y[n] \xrightarrow{\epsilon}_{\mathrm{B}} \ldots \xrightarrow{\epsilon}_{\mathrm{B}} z_n = z$ which implies that $y \xrightarrow{*}_{\mathrm{B}} z$. We next show that z_i is irreducible for all i by induction. Clearly, $z_1 = y[1]$ is irreducible. Given that z_{i-1} is irreducible, we must show that z_i is irreducible. We have two cases: either a suffix of the form xx^Rx was found during the ith iteration or one was not. In the former case let $\alpha y[i]y[i]\alpha^R\alpha y[i]$, $\alpha \in \Sigma^*$ be the suffix of the form xx^Rx found. Then $z_{i-1} = \beta\alpha y[i]y[i]\alpha^R\alpha$ and $z_i = \beta\alpha y[i]$, $\beta \in \Sigma^*$. Assume z_i is not irreducible.

Then z_{i-1} is also not irreducible since z_i is a substring of z_{i-1}, a contradiction. In the latter case we have $z_{i-1} = \beta$ and $z_i = \beta y[i]$, $\beta \in \Sigma^+$. Assume that z_i is not irreducible. Since no suffixes of z_i were of the form xx^Rx, some substring of β must be of the form xx^Rx. This implies that $z_{i-1} = \beta$ is not irreducible, a contradiction. Therefore, z_i is irreducible $\forall i$, and in particular $z = z_n$ is irreducible. Thus, we have $y \xrightarrow{*}_{\mathrm{B}} z$ and z irreducible which implies that z is the normal form of y. ■

The worst case running time of the algorithm in Figure 2 occurs when y is itself irreducible. In this case, no suffix of the form xx^Rx is found in step 5 of the algorithm. If we search for this suffix in the obvious manner, then $O(i^2)$ time is required in the ith iteration. This yields an overall running time of $O(n^3)$.

4.2 Inference of Linear Chains from General Walks

A *walk* on a linear chain z is a sequence of colors corresponding to the edges traversed in some walk on z that begins at some vertex v_i and at some point passes through the left vertex v_1 and right vertex v_N. We no longer require that the walk start at the left vertex or end at the right vertex. For the linear chain $z =$ `abcda`, $y_1 =$ `baabcccdaadc` and $y_2 =$ `abcdaadcbb` are both walks on z. The string y is a walk on the linear chain z if an only if it embeds in z. For $y =$ `baabcccdaadc` and $z =$ `abcda`, the embedding of y in z partitions y as follows: `ba`, `abc`, `c`, `cda`, `adc`.

For the set Σ^+, we define new binary relations $\rightarrow_{\mathrm{H}} = \{(x^Rxq, xq) \mid q \in \Sigma^*, x \in \Sigma^+\}$ and $\rightarrow_{\mathrm{T}} = \{(pxx^R, px) \mid p \in \Sigma^*, x \in \Sigma^+\}$. (The letters "H" and "T" are used because these relations affect the *head* and *tail* of the string, respectively.) Let $\rightarrow_{\mathrm{HBT}} = \rightarrow_{\mathrm{B}} \cup \rightarrow_{\mathrm{H}} \cup \rightarrow_{\mathrm{T}}$. If $y \xrightarrow{*}_{\mathrm{HBT}} z$, then z is called an *HBT-contraction* of y. Conversely, y is called an *HBT-expansion* of z. Clearly, $\rightarrow_{\mathrm{HBT}}$ is strictly decreasing since $\forall y, z \in \Sigma^+$, if $y \rightarrow_{\mathrm{HBT}} z$ then $|z| < |y|$.

Lemma 3 *For all strings $y, z \in \Sigma^+$, y is a walk on the linear chain $z \iff y$ is an HBT-expansion of z.*

Proof ($\Rightarrow$): By contradiction, assume that $\Rightarrow$ implication is false. Then there must exist some shortest counterexample y such that y is a walk on some linear chain z, and y is not an *HBT*-expansion of z. Consider the embedding of y in z. Without loss of generality, assume that the walk on z visits the left vertex before the right vertex. Let the *head* of y, y^H, be the prefix of y corresponding to all edges traversed until the left vertex is reached for the first time. Let the *body* of y, y^B, be the subsequent portion of y until the right vertex is reached for the last time. Let the *tail* of y, y^T, be the remaining portion of y. Clearly, y^B is an end-to-end walk on z, y^H is a walk on α^R where α is a prefix of z, and y^T is a walk on β^R where β is a suffix of z. Let $z = \alpha\gamma_1 = \gamma_2\beta$. If y^H is not an *HBT*-expansion of α^R then y^H is a shorter counterexample than y, a contradiction. Similarly, if y^T is not an *HBT*-expansion of β^R then y^T is a shorter counterexample than y, also a contradiction. We therefore have $y^H \xrightarrow{*}_{\mathrm{HBT}} \alpha^R$ and $y^T \xrightarrow{*}_{\mathrm{HBT}} \beta^R$. By Lemma 1, $y^B \xrightarrow{*}_{\mathrm{B}} z$ and therefore $y^B \xrightarrow{*}_{\mathrm{HBT}} z$. We now have the following: $y = y^H y^B y^T \xrightarrow{*}_{\text{HBT}} y^H \alpha\gamma_1 y^T \xrightarrow{*}_{\text{HBT}} \alpha^R \alpha\gamma_1 y^T \rightarrow_{\text{HBT}} \alpha\gamma_1 y^T = \gamma_2\beta y^T \xrightarrow{*}_{\text{HBT}} \gamma_2\beta\beta^R \rightarrow_{\text{HBT}} \gamma_2\beta = z$. Thus $y \xrightarrow{*}_{\text{HBT}} z$, a contradiction. ∎

Proof (⇐): By definition, if y is an HBT-expansion of z then $y \xrightarrow{*}_{\text{HBT}} z$. Since $\rightarrow_{\text{HBT}}$ is strictly decreasing, we may write $y = w_n \rightarrow_{\text{HBT}} w_{n-1} \rightarrow_{\text{HBT}} \ldots \rightarrow_{\text{HBT}} w_2 \rightarrow_{\text{HBT}} w_1 = z$ for some finite n. We show that w_i is a walk on z by induction on i. Clearly, w_1 is a (trivial) walk on z. Assume w_{i-1} is a walk on z. Since $w_i \rightarrow_{\text{HBT}} w_{i-1}$, we have either $w_i \rightarrow_{\text{B}} w_{i-1}$ or $w_i \rightarrow_{\text{H}} w_{i-1}$ or $w_i \rightarrow_{\text{T}} w_{i-1}$. The first case was examined in the proof of Lemma 1. If $w_i \rightarrow_{\text{T}} w_{i-1}$ then $w_i = pxx^R$ and $w_{i-1} = px$ for some $p \in \Sigma^+, x \in \Sigma^*$. The string w_i may be embedded in z as follows: embed px in the same manner that px is embedded for w_{i-1}, and embed x^R by tracing the embedding of x backwards. The case $w_i \rightarrow_{\text{H}} w_{i-1}$ is similar. Therefore w_i is a walk on z, and by induction $y = w_n$ is a walk on z. ∎

Lemma 4 *The binary relation $\rightarrow_{HBT} = \rightarrow_B \cup \rightarrow_H \cup \rightarrow_T$ is confluent.*

Since $\rightarrow_{\text{HBT}}$ is strictly decreasing, we need only show that $\rightarrow_{\text{HBT}}$ is locally confluent. This proof is given in Section 6.2.

Theorem 5 *If y is a walk, then the shortest linear chain that can produce y is $\hat{y}$, the HBT-normal-form of y.*

Proof: Similar to proof of Theorem 4. ∎

Lemma 5 *If $w \hat{\rightarrow}_B x \hat{\rightarrow}_H y \hat{\rightarrow}_T z$, then z is the HBT-normal-form of w.*

Proof: We have $w \xrightarrow{*}_{\text{B}} x \xrightarrow{*}_{\text{H}} y \xrightarrow{*}_{\text{T}} z$ which implies that $w \xrightarrow{*}_{\text{HBT}} z$. Clearly, z is irreducible under $\rightarrow_{\text{T}}$. Also, z is a prefix of y by the definition of $\rightarrow_{\text{T}}$. Suppose that z is not irreducible under $\rightarrow_{\text{H}}$. Then y is not irreducible under $\rightarrow_{\text{H}}$, which contradicts the definition of $\hat{\rightarrow}_{\text{H}}$. Further, z is a substring of x by the definition of $\rightarrow_{\text{H}}$ and $\rightarrow_{\text{T}}$. Suppose that z is not irreducible under $\rightarrow_{\text{B}}$. Then x is not irreducible under $\rightarrow_{\text{B}}$ which contradicts the definition of $\hat{\rightarrow}_{\text{B}}$. Therefore, z is irreducible under $\rightarrow_{\text{B}}$ & $\rightarrow_{\text{H}}$ & $\rightarrow_{\text{T}}$ and hence under $\rightarrow_{\text{HBT}}$. Thus z is the HBT-normal-form of w. ∎

To obtain the minimum consistent linear chain that can produce a walk y, we must find the HBT-normal-form of y. By Lemma 5, we may obtain this string in three stages. The algorithm in Figure 2 accomplishes the first stage as previously noted. We now need algorithms to find the H-normal-form and T-normal-form of y. Clearly, only one is necessary since $\rightarrow_{\text{H}}$ and $\rightarrow_{\text{T}}$ are symmetric.

Claim 2 *On input y, the algorithm in Figure 3 produces $\hat{y}$, the T-normal-form of y.*

Proof: Let $y_1, y_2, y_3, \ldots$ be the sequence of strings obtained during a run of this algorithm. Each new y_i is obtained in step 4 when the **if** statement in step 3 is true, and clearly $y_{i-1} \rightarrow_{\text{T}} y_i$. Therefore the output of this algorithm is an T-contraction of the input. The

```
Input:
  y - string of symbols
  n - length of y
Output:
  y - string of symbols corresponding to ŷ
  n - length of y
Procedure:
  l = 1;
  while 2l ≤ n do
    if y has a length 2l suffix of the form xx^R then
        n = n − l;
        l = 1;
    else
        l = l + 1;
    endif
  end
```

Figure 3: Algorithm for obtaining the T-normal-form of y.

algorithm does not terminate until the final y_i is found to have no suffixes of the form xx^R. Thus, the output of this algorithm is the T-normal-form of the input. ∎

The worst case running time of this algorithm is $O(n^2)$. If the input is irreducible, $O(n^2)$ time is be spent trying to find a suffix of the form xx^R. If the input is not irreducible, $O(i^2)$ time is spent finding each xx^R suffix of length $2i$ or verifying that one does not exist. We thus obtain the recurrence $T(n) = \max_i\{T(n-i) + O(i^2)\}$, whose solution is $O(n^2)$.

We can therefore find the HBT-normal-form of y in $O(n^3)$ time by applying the algorithm in Figure 2 ($O(n^3)$ time) followed by two applications of the algorithm in Figure 3 ($O(n^2)$ time).

5 Inference of Cycles from Walks

In this section we examine the inference of cycles from walks. We define a *cycle* as a degree-bound 2, undirected, edge-colored graph $G = (V, E, c)$ with vertices $V = \{v_1, v_2, \ldots, v_N\}$, edges $E = \{(v_1, v_2), (v_2, v_3), \ldots, (v_{N-1}, v_N), (v_N, v_1)\}$ and edge colors $c(u, v)\ \forall (u, v) \in E$. Let Σ be the set of all colors. If we "break" a cycle C at vertex i, the graph we obtain is a linear chain which we denote C_i. C_i may be represented by a string in Σ^+ corresponding to the sequence of edge colors $c(v_i, v_{i+1}), c(v_{i+1}, v_{i+2}), \ldots, c(v_{i-1}, v_i)$.

A walk on a cycle C is a sequence of colors corresponding to the edges traversed in some walk on C that begins at some vertex v_i and eventually crosses all edges in C (i.e. *completes* the cycle). If we let Σ be the set of letters, then $C_1 =$ `abcd` is a linear chain corresponding to a cycle C, and $w_1 =$ `bcdabbbc` and $w_2 =$ `cdaadcba` are both walks on C. The string w is a walk on a cycle C if and only if it embeds in C. For $w =$ `bcdabbbc` and $C_1 =$ `abcd`, the embedding of w in C partition w as follows: `bcdab`, `b`, `bc`.

In order to develop an algorithm for inferring the minimum consistent cycle C from a given walk w, we must first give some properties of minimum consistent cycles. Let w be a walk and let C be the minimum consistent cycle corresponding to w. By definition, w must embed in C, and w must eventually cross every edge in C. Consider the embedding of w in C. There must exist some vertex k corresponding to that vertex reached when the walk w completes the cycle C for the first time. Let x be the prefix of w corresponding to all edges traversed until vertex k is reached for the first time. Let y be the subsequent portion of w corresponding all edges traversed until vertex k is again reached and the cycle is first completed. Let z be the remaining portion of w. We observe the following facts: x^R must embed in C starting at vertex k, y is an end-to-end walk on C_k, and z must embed in C starting at vertex k. If y is an end-to-end walk on C_k, then $y \xrightarrow{*}_{\text{B}} C_k$ by Lemma 1.

Lemma 6 *C_k is irreducible.*

Proof sketch: Suppose that C_k is not irreducible. We then have $C_k = ptt^Rtq$ and $y \xrightarrow{*}_{\text{B}} ptt^Rtq \rightarrow_{\text{B}} ptq$. Therefore, y is an end-to-end walk on ptq and must embed in ptq. By arguments similar to those used in the proof of Lemma 1, we claim that x^R and z embed in the cycle C' corresponding to the linear chain ptq. We now have a cycle C' consistent with w that is smaller than the minimum consistent cycle C, a contradiction. ∎

Consider the following algorithm: On input w, let $s \in \Sigma^+$ be a substring of w, and let $p, q \in \Sigma^*$ be strings such that $w = psq$. Compute $\hat{s}$, the B-normal-form of s and consider the cycle K such that $K_1 = \hat{s}$. If p^R and q embed in K starting at vertex 1, consider this cycle a *success*. Repeat for all possible substrings s and output the smallest success.

Clearly, w is a walk on each cycle that is a success, and the minimum consistent cycle C is among the successes. Thus, the output of this algorithm is the minimum consistent cycle C.

On input w (where $|w| = n$), the algorithm requires $O(n^2)$ iterations, one for each possible substring s. Each iteration requires $O(n^3)$ time to calculate $\hat{s}$ and $O(n^2)$ time to check if p^R and q embed in K (by calculating the "reachable" vertex set for successive prefixes of the string in question). Thus, the algorithm runs in polynomial time.

6 Confluence Results

In this section we give proof sketches for the confluence results used above. Since each of the binary relations used is strictly decreasing, to prove confluence we need only show local confluence.

6.1 B Is Confluent

We have defined $\rightarrow_{\text{B}} = \{(pxx^Rxq, pxq) \mid p, q \in \Sigma^*, x \in \Sigma^+\}$. For $\rightarrow_{\text{B}}$ to be locally confluent, we must have that $\forall w, u, v \; w \rightarrow_{\text{B}} u \;\&\; w \rightarrow_{\text{B}} v \Rightarrow \exists z, \; u \xrightarrow{*}_{\text{B}} z \;\&\; v \xrightarrow{*}_{\text{B}} z$. If $w \rightarrow u$, then $w = p_1xx^Rxq_1$ and $u = p_1xq_1$. If $w \rightarrow v$, then $w = p_2yy^Ryq_2$ and $v = p_2yq_2$. Notice that w contains *both* an xx^Rx substring *and* a yy^Ry substring. If these substrings do not overlap, then there trivially exists a z where $u \rightarrow_{\text{B}} z$ and $v \rightarrow_{\text{B}} z$. The cases where the xx^Rx and yy^Ry substrings do overlap are somewhat more difficult.

Let $\rightarrow^i_{\mathrm{B}} = \{(pxx^R xq, pxq) \mid p, q \in \Sigma^*, x \in \Sigma^+, |x| \leq i\}$. To show that $\rightarrow_{\mathrm{B}}$ is locally confluent, we show that $\rightarrow^i_{\mathrm{B}}$ is locally confluent for all i by induction. The base case $\rightarrow^1_{\mathrm{B}}$ is trivial. We now assume that $\rightarrow^{n-1}_{\mathrm{B}}$ is locally confluent (hence confluent) and show that this implies that $\rightarrow^n_{\mathrm{B}}$ is locally confluent. Consider $xx^R x$ and $yy^R y$ substrings within w which overlap. By examining the various ways in which these two substrings can overlap, it can be shown that the non-trivial cases always have the following property: if $w \rightarrow^n_{\mathrm{B}} u$ and $w \rightarrow^n_{\mathrm{B}} v$, then $w \overset{*}{\rightarrow}^{n-1}_{\mathrm{B}} u$ and $w \overset{*}{\rightarrow}^{n-1}_{\mathrm{B}} v$. Since $\rightarrow^{n-1}_{\mathrm{B}}$ is confluent, this implies that $\exists z$ such that $u \overset{*}{\rightarrow}^{n-1}_{\mathrm{B}} z$ and $v \overset{*}{\rightarrow}^{n-1}_{\mathrm{B}} z$. But $\rightarrow^{n-1}_{\mathrm{B}} \subset \rightarrow^n_{\mathrm{B}}$ which implies that $u \overset{*}{\rightarrow}^{n}_{\mathrm{B}} z$ and $v \overset{*}{\rightarrow}^{n}_{\mathrm{B}} z$. Therefore, $\rightarrow^n_{\mathrm{B}}$ is locally confluent. Thus, $\rightarrow^i_{\mathrm{B}}$ is locally confluent for all i which implies that $\rightarrow_{\mathrm{B}}$ is locally confluent.

6.2 *HBT* Is Confluent

Given the union of a number of strictly decreasing binary relations, to prove that this union is locally confluent it is sufficient (though not necessary) to show that each of the relations is locally confluent and that they are pairwise locally confluent. We have already shown that $\rightarrow_{\mathrm{B}}$ is locally confluent. By using methods similar to those above, we can show that $\rightarrow_{\mathrm{H}}$ is locally confluent (and hence $\rightarrow_{\mathrm{T}}$ by symmetry). Again, by employing methods similar to those above we can also show that $\rightarrow_{\mathrm{B}} \cup \rightarrow_{\mathrm{H}}$ (and hence $\rightarrow_{\mathrm{B}} \cup \rightarrow_{\mathrm{T}}$ by symmetry) as well as $\rightarrow_{\mathrm{H}} \cup \rightarrow_{\mathrm{T}}$ are locally confluent. This yields the desired result.

7 Conclusion

We have shown that it is possible to efficiently infer minimum consistent degree-bound 2 graphs from walks. Our algorithms run in polynomial time, and their correctness is based on some surprising properties of various sets of rewrite rules . It is an open question whether the minimum consistent inference of degree-bound k graphs for $k > 2$ is tractable or NP-complete.

References

[1] A. V. Aho, R. Sethi, and J. D. Ullman. Code optimization and finite Church-Rosser theorems. In R. Rustin, editor, *Design and Optimization of Compilers*, pages 89–105. Prentice-Hall, 1972.

[2] A. V. Aho and J. D. Ullman. Transformations on straight line programs. In *Conference Record Second Annual ACM Symposium on Theory of Computing*, pages 136–148, 1970.

[3] Dana Angluin. *An application of the theory of computational complexity to the study of inductive inference*. PhD thesis, University of California, Berkeley, 1976.

[4] Henk P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic*. North-Holland, 1981. Revised Edition, 1984.

[5] A. Church and J. B. Rosser. Some properties of conversion. *Transaction of AMS*, 39:472–482, 1936.

[6] E. Mark Gold. Complexity of automaton identification from given data. *Information and Control*, 37:302–320, 1978.

[7] G. Huet and D. C. Oppen. Equations and rewrite rules: a survey. In R. Book, editor, *Formal Language Theory*, pages 349–393. Academic Press, 1980.

[8] Gérard Huet. Confluent reductions: abstract properties and applications to term rewriting systems. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 30–45. IEEE, 1977.

[9] Steven Rudich. Inferring the structure of a Markov chain from its output. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*, pages 321–326. IEEE, 1985.