Compression of Words over a Partially Commutative Alphabet

Serap A. Savari†
Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor

Introduction

Multiprocessor configurations, distributed systems and communication networks are examples are systems consisting of a collection of distinct processes which communicate with one another but are also partly autonomous. Here certain events are allowed to occur independently while others must happen in a predetermined order. In other words, there is a *partial ordering* of events rather than a total ordering [1].

The events of sequential processes are well modeled by a string of events. A recent award-winning paper in the computer architecture literature [2] applies a grammar-based data compression scheme to the sequence of events that occur while a computer program runs and uses the hierarchical structure inferred by the algorithm to better understand the program's dynamic behavior and improve its performance. The implicit assumption in using lossless data compression for this application is that there is a well-defined total ordering of event occurrences. Trace theory [3] is one way to generalize the notion of a string in order to model the executions of concurrent processes. The sequential observer of a concurrent system is provided with a set of atomic actions together with a labeled and undirected *dependence relation* or *noncommutation graph* indicating which actions can be performed independently or concurrently. For a noncommutation graph G with vertex set V two words are *congruent* if one can be transformed into the other by a sequence of steps each of which consists of interchanging two consecutive letters that are nonadjacent vertices in G. For example, if the noncommutation graph G is given by a—b—c—d, then the two words $dabac$ and $abdca$ are equivalent since $dabac \equiv_G adbac \equiv_G adbca \equiv_G abdca$. To generalize lossless data compression to concurrent systems, [4] introduces a compression problem where it is only necessary to reproduce a string which is equivalent to the original string and provides some heuristic compression schemes. We mention in passing that this compression problem also arises in the compression of executable code [5]. In [6], we initiate a study of this compression problem from an information theoretic perspective.

Let us assume we are given a discrete, memoryless source that emits symbols belonging to the vertex set V, and let $P(v)$ denote the probability of $v \in V$. Let G denote a graph on vertex set V. Our goal is to minimize the average number of bits per symbol needed to represent the congruence class containing a word emitted from the source as the word length approaches infinity. We call the limit of the best achievable rate as L approaches

†This work was done while the author was with the Computing Sciences Research Center, Bell Labs, Lucent Technologies.

infinity the *interchange entropy* of the source, which exists and which we will denote by $H_\iota(G, P)$.

Basic Properties

Let E denote the edge set of a graph.

Proposition 1 (Monotonicity) *If F and G are two graphs on the same vertex set and $E(F) \subseteq E(G)$, then for any probability distribution P we have $H_\iota(F,P) \ \leq \ H_\iota(G,P)$.*

Proposition 2 (Subadditivity) *Let F and G be two graphs on the same vertex set V and define $F \cup G$ to be the graph on V with edge set $E(F) \cup E(G)$. For any fixed probability distribution P we have $H_\iota(F \cup G, P) \ \leq \ H_\iota(F,P) + H_\iota(G,P)$.*

Proposition 3 (Disjoint Components) *Let the subgraphs G_j denote the connected components of the graph G. For a probability distribution P on $V(G)$ define the probability distributions $P_j(x) = P(x)[P(V(G_j))]^{-1}$, $x \in V(G_j)$. Then $H_\iota(G,P) = \sum_j P(V(G_j)) H_\iota(G_j, P_j)$.*

Theorem 6 *Suppose V is of the form $V = V_1 \cup V_2 \cup \ldots \cup V_k$ with $|V_i| = m_i$, $i \in \{1, 2, \ldots, k\}$ and label the elements of V_i as $v_{i,j}$, $i \in \{1, 2, \ldots, k\}$, $j \in \{1, 2, \ldots, m_i\}$. For the complete k-partite graph $K_{m_1, m_2, \ldots, m_k}$ there is an edge corresponding to every pair of vertices $\{v_{i,j}, v_{l,n}\}, v_{i,j} \in V_i, v_{l,n} \in V_l$, $l \neq i$, and no two vertices from the same subset V_i are adjacent for any $i \in \{1, 2, \ldots, k\}$. Define $Q_i = \sum_{j=1}^{m_i} P(v_{i,j})$, $i \in \{1, 2, \ldots, k\}$. Then*
$H(P) - H_\iota(K_{m_1, m_2, \ldots, m_k}, P) \ =$

$$\sum_{S=2}^{\infty} \log_2(S) \sum_{i: m_i \geq 2} (1 - Q_i) \left({Q_i}^S - \sum_{j=1}^{m_i} \left(\frac{P(v_{i,j})}{1 - Q_i + P(v_{i,j})} \right)^S \right).$$

References

[1] L. Lamport, Time, clocks, and the ordering of events in a distributed system, *Comm. ACM*, 21, 558–565, 1978.

[2] J. Larus, Whole program paths, *ACM SIGPLAN Conf. Prog. Lang. Des. Implem.* 259–269, Atlanta, GA, May 1999.

[3] A. Mazurkiewicz, Trace theory, in W. Brauer et. al., ed., *Petri Nets, Applications and Relationship to other Models of Concurrency*, Lecture Notes in Computer Science 255, 279–324, Springer, Berlin, 1987.

[4] R. Alur, S. Chaudhuri, K. Etessami, S. Guha and M. Yannakakis, Compression of partially ordered strings, *Proc. CONCUR 2003* (14th International Conference on Concurrency Theory), Marseille, France, September 2003.

[5] M. Drinić and D. Kirovski, PPMexe: PPM for compressing software, *Proc. 1997 I.E.E.E. Data Comp. Conf.* 192–201, Snowbird, UT, March 2002.

[6] S. A. Savari, Compression of words over a partially commutative alphabet, *IEEE Transactions on Information Theory,* 50(7):1425–1441, July 2004.