HEWLETT PACKARD

Nested Input-Constrained Codes

Josh Hogan, Ron M. Roth[†], Gitit Ruckenstein[‡]
Information Storage Technology Laboratory
HPL-98-165
September, 1998

constrained systems, deterministic encoders, finite-state encoders, input-constrained channels, nested encoders

An input-constrained channel, or simply a constraint, is a set of S of words that is generated by a finite labeled directed graph. In the context of coding, the study of constraints mainly aims at designing encoders that map unconstrained input words into words of S in a lossless manner. In most applications, the encoders are finite-state machines that map in each time slot a fixed-length input block into a fixed-length channel block, and decoding is carried out by looking ahead at finitely-many upcoming channel blocks. The state diagram of a finite-state encoder provides a graph presentation of that encoder. In the special case where this graph is (output) deterministic, only the current channel block is needed for decoding the current input block.

In this work, the problem of designing encoders that can serve two constraints simultaneously is considered. Specifically, given two constraints S_1 and S_2 such that $S_1 \subseteq S_2$ and two prescribed rates, conditions are provided for the existence of respective deterministic finite-state encoders $\mathcal{E}_1$ and $\mathcal{E}_2$, at the given rates, such that (the state diagram of) $\mathcal{E}_1$ is a subgraph of $\mathcal{E}_2$. Such encoders are referred to as *nested* encoders. The provided conditions are also constructive in that they imply an algorithm for finding such encoders when they exist. The nesting structure allows to decode $\mathcal{E}_1$ while using the decoder of $\mathcal{E}_2$.

Recent developments in optical recording suggest a potential application that can take a significant advantage of nested encoders.

Internal Accession Date Only

[†] Hewlett-Packard Laboratories Israel, Technion City, Haifa 32000 Israel
[‡] Computer Science Department, Technion, Haifa 32000 Israel

1 Introduction

Input-constrained channels, also known as *constrained systems* or simply *constraints*, are widely-used models for describing the read-write requirements of secondary storage systems, such as magnetic disks or optical memory devices. A constraint S is defined as the set of (constrained) words obtained from reading the labels of paths in a finite labeled directed graph G. We say that G is a *presentation* of S.

As an example, for integers $0 \leq d \leq k$, the (d,k)-runlength-limited (RLL) constraint consists of the binary words in which the runs of 0's between consecutive 1's have length at least d, and no run of 0's has length greater than k. E.g., the current compact disk standard uses the constraint $(d,k) = (2,10)$, and the following is a word satisfying this constraint:

$$\cdots 00100001001000000001000 \cdots .$$

The parameter k is imposed to guarantee sufficient sign-changes in the recorded waveform which are required for clock synchronization during readback to prevent clock drifting. The parameter d is required to prevent inter-symbol interference. A (d,k)-RLL constraint will be denoted by $S_{(d,k)}$. A graph presentation of $S_{(d,k)}$ is shown in Figure 1.

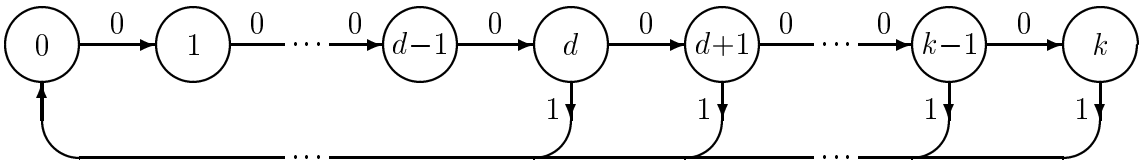

Figure 1: Graph presentation $S_{(d,k)}$.

One goal in the study of constraints is designing encoders that map unconstrained binary sequences, referred to as *source sequences,* into words of a given constraint S. A *rate* $p:q$ *finite-state encoder for* S encodes a binary p-block of source symbols into a q-block in S in a state-dependent and uniquely-decodable manner.

In the current emerging technology and development of erasable and writable dense optical disks, we may face the scenario where recorders will differ in their writing capabilities. More specifically, home recorders may be able to record data in a lower density compared to factory-stamped or manufacturer-recorded disks. This, in turn, implies that different recorders may need to use coding schemes of different constraints. Specifically, home recorders might use an encoder $\mathcal{E}_1$ at rate $p_1 : q_1$ for a constraint S_1, say $S_1 = S_{(d_1,k_1)}$; on the other hand, manufacturers of optical disks may be able to record data using an encoder $\mathcal{E}_2$ at a higher rate $p_2 : q_2$ for a constraint S_2 such that $S_1 \subseteq S_2$; e.g., $S_2 = S_{(d_2,k_2)}$

where $d_2 \leq d_1$ and $k_2 \geq k_1$. (The current values of the standard are $d_2 = 2$ and $k_2 = 10$, with $p_2 : q_2 = 8 : 17$ in the compact disk, and $p_2 : q_2 = 8 : 16$ in the read-only DVDs; see [3], [4].)

In spite of the different encoders, we would still like a disk player to have the capability of decoding both encoding schemes. One solution is to have on board a separate decoder for each encoder. In such a case, we will have a decoder $\mathcal{D}_1$ for $\mathcal{E}_1$ that decodes sequences of the constraint S_1 by dividing each sequence into nonoverlapping q_1-blocks of channel symbols, and mapping each q_1-block into an input binary p_1-block (the mapping can be state-dependent). A decoder $\mathcal{D}_2$ for $\mathcal{E}_2$ will decode each q_2-block in a sequence of S_2 into an input p_2-block.

An alternative approach, which we investigate in this work, is designing the encoders $\mathcal{E}_1$ and $\mathcal{E}_2$ so that their decoders can be combined to a great extent. To this end, we will assume that the alphabets of the constraints S_1 and S_2 are the same (e.g., both alphabets are binary, as is the case with RLL constraints). Furthermore, we will assume that q_1 and q_2 are equal to the same number q (and so $p_1 \leq p_2$). The decoder $\mathcal{D}_1$ will be obtained by cascading $\mathcal{D}_2$ with a combinational circuit (function) ψ that maps input p_2-blocks to input p_1-blocks, as shown in Figure 2. If such a combined decoding scheme exists, we will say that the encoder $\mathcal{E}_1$ is *(block) observable* from $\mathcal{E}_2$.

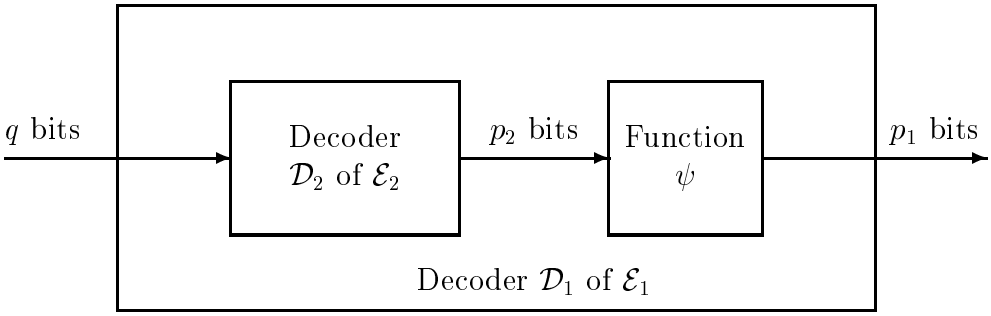

Figure 2: Encoder $\mathcal{E}_1$ observable from encoder $\mathcal{E}_2$.

This work will concentrate on the study of observable encoders that are *deterministic*, namely, at each state, distinct input p-blocks map into distinct q-blocks of channels symbols. There is, of course, a loss of generality in this assumption, and the study of the more general case is still an open research area. As we shall see, the special case of deterministic encoders is already elaborate as is. Of particular interest are *block decodable* encoders; these are deterministic encoders in which the decoding process of an input p-block is state-independent and requires only the knowledge of the current q-block of channel symbols. We will also assume that our encoders are *irreducible*, namely, every

state is reachable from every other state in the state diagrams. The basic definitions used throughout this work are presented formally in Section 2. Those definitions are demonstrated through examples in Section 3.

In Section 4, we show that for irreducible deterministic encoders and for constraints S_1 and S_2, the following two conditions are equivalent:

- There exists a rate $p_1 : q$ finite-state encoder for S_1 that is observable from a rate $p_2 : q$ finite-state encoder for S_2.
- There exists a rate $p_1 : q$ finite-state encoder for S_1 that is a subgraph of a rate $p_2 : q$ finite-state encoder for S_2. We say that the former encoder is *nested* in the latter.

This equivalence result motivates us to study the nesting property in more detail. In Section 5, we provide necessary and sufficient conditions for the existence of nested encoders in terms of the graph presentations of the constraints. The provided conditions are also constructive in the sense that they imply an algorithm for finding nested encoders when they exist. We point out, however, that the nesting property may sometimes be in conflict with other properties that we would like the encoders to possess, e.g., that they are block decodable (see Example 3.2 in Section 3).

It is known that a state diagram of a rate $p : q$ deterministic encoder for a constraint S can be obtained as a subgraph of (a power of) a certain graph presentation of S, provided that a deterministic encoder at rate $p : q$ does indeed exist (see Proposition 2.7 below). In Section 6, we attempt to generalize this property in what we call the *diamond condition set*. Yet, as we show, there is an additional condition that we need to assume about the constraints so that the generalization indeed holds.

2 Definitions and background

Many of the definitions here can be found in [6].

2.1 Graphs and constraints

A *finite labeled directed graph* $G = (V, E, L)$ consists of —

- a nonempty finite set of states $V = V_G$;
- a finite set of edges $E = E_G$ where each edge e has an *initial state* $\sigma_G(e)$ and a *terminal state* $\tau_G(e)$, both in V;

- edge labels $L = L_G : E \to \Sigma$ drawn from a finite alphabet Σ.

For simplicity, we will refer to a finite labeled directed graph as simply a *graph*. We will also use the notation $u \xrightarrow{a} v$ to denote an edge labeled a from state u to state v in G. The set of outgoing edges from state u in G will be denoted by $E_G(u)$, and $L_G(E_G(u))$ will stand for the set of labels of the edges in $E_G(u)$. The *minimum degree* of G is the smallest among the out-degrees of any state in G, namely, $\min_{u \in V_G} |E_G(u)|$. A graph G is *n-regular* if the out-degree of each state in G equals n.

A path π in a graph G is a finite sequence of edges $e_1 e_2 \dots e_\ell$ such that $\sigma_G(e_{j+1}) = \tau_G(e_j)$ for $i = 1, 2, \dots, \ell - 1$. The length of a path π is the number of edges along the path. A graph can be used to generate finite symbol sequences, by reading off the labels along paths in the graph, thereby producing *words*. If a path π is labeled by a word $\mathbf{w}$, we say that π generates $\mathbf{w}$. A word of length ℓ generated by G will be called an *ℓ-block*.

A *constrained system* (or *constraint*), denoted S, is the set of all words (i.e., finite sequences) obtained from reading the labels of paths in a graph G. We say that G *presents* S or is a *presentation* of S, and we write $S = S(G)$. The *alphabet* of S is the set of symbols that actually occur in words of S and is denoted $\Sigma = \Sigma(S)$.

Let $G_1 = (V_1, E_1, L_1)$ and $G_2 = (V_2, E_2, L_2)$ be graphs. We say that G_1 is a *subgraph* of G_2 if $V_1 \subseteq V_2$, $E_1 \subseteq E_2$, and L_1 is the restriction of L_2 to E_1. We will use the notation $G_1 \subseteq G_2$ and we will say that G_1 is *nested* in G_2. If E_1 consists of all edges in E_2 whose initial and terminal states are in V_1, we will say that G_1 is a subgraph of G_2 *induced* by V_1.

Two (finite labeled directed) graphs are *isomorphic* if there is a one-to-one and onto mapping from states to states and edges to edges that preserves initial states, terminal states, and labels.

A graph is *deterministic* if at each state the outgoing edges are labeled distinctly. It is known that every constraint has a deterministic presentation (see, e.g., [6, Section 2.2.1]).

A graph G has *finite anticipation* if there is an integer N such that any two paths of length $N+1$ with the same initial state and labeling must have the same initial edge. The *anticipation* $\mathcal{A}(G)$ of G is the smallest N for which this holds.

A graph is *lossless* if any two distinct paths with the same initial state and terminal state generate different words.

Let G be a graph. The *adjacency matrix* $A = A_G = [(A_G)_{u,v}]_{u,v \in V_G}$ is the $|V_G| \times |V_G|$ matrix where the entry $(A_G)_{u,v}$ is the number of edges from state u to state v in G. We denote by $\lambda(A_G)$ the largest absolute value of any eigenvalue of A_G. It is known that

$\lambda(A_G)$ is actually an eigenvalue of A_G (see, e.g., [11, Ch. 1]).

Let S be a constraint and let $N(\ell; S)$ denote the number of words of length ℓ in S. The *(Shannon) capacity* of S is defined by

$$\mathsf{cap}(S) \;=\; \lim_{\ell \to \infty} \frac{1}{\ell} \log N(\ell; S) \;,$$

where hereafter all logarithms are taken to base 2.

The following well-known result shows how to compute capacity (see [6, Section 3.2.2], [8]).

Theorem 2.1 *Let S be a constraint and let G be a lossless (for instance, deterministic) presentation of S. Then,*

$$\mathsf{cap}(S) \;=\; \log \lambda(A_G) \;.$$

Table 5.4 in [3, p. 91] lists the capacity values of several (d, k)-RLL constraints.

Let G be a graph. The *qth power of G*, denoted G^q, is the graph with the same set of states as G, but one edge for each path of length q in G, labeled by the q-block generated by that path. It is easy to see that $A_{G^q} = (A_G)^q$ and, so, $\lambda(A_{G^q}) = (\lambda(A_G))^q$. For a constraint S presented by a graph G, the *qth power of S*, denoted S^q, is the constraint presented by G^q. It follows from Theorem 2.1 that

$$\mathsf{cap}(S^q) \;=\; q \cdot \mathsf{cap}(S) \;. \tag{1}$$

2.2 Irreducibility

A graph G is *irreducible* (or *strongly connected*), if for any ordered pair of states u, v, there is a path from u to v in G.

An *irreducible component* of a graph G is a maximal (with respect to inclusion) irreducible subgraph of G. The irreducible components of G are the subgraphs of G that are induced by equivalence classes of the following relation defined over the states of G: $u \sim v$ if there is a path from u to v and a path from v to u. A *trivial irreducible component* is an irreducible component that consists of one state and no edges.

An *irreducible sink* is an irreducible component H such that any edge that originates in H must also terminate in H. Any graph can be broken down into irreducible components with 'transient' connections between the components, and every graph has at least one irreducible sink [11].

A constraint S is *irreducible* if for every pair of words $\mathbf{w}, \mathbf{w}'$ in S, there is a word $\mathbf{z}$ such that $\mathbf{wzw}'$ is in S. A constraint that is not irreducible is called *reducible*.

We will make use of the following result from [7]:

Lemma 2.2 *Let S be an irreducible constraint and let G be a graph such that $S \subseteq S(G)$. Then for some irreducible component G' of G, $S \subseteq S(G')$.*

It follows from this result that a constraint S is irreducible if and only if it can be presented by some irreducible (in fact, deterministic) graph.

2.3 Shannon cover

Let u be a state in a graph G. The *follower set* of u in G, denoted $\mathcal{F}_G(u)$, is the set of all (finite) words that can be generated from u in G. Two states u and u' in a graph G are said to be *follower-set equivalent* if they have the same follower set. It is easy to verify that follower-set equivalence satisfies the properties of an equivalence relation. A graph G is called *reduced* if no two states in G are follower-set equivalent.

If a graph G presents a constraint S, we can construct a reduced graph H (called the *reduction* of G) that presents the same constraint S by merging states in G that are follower-set equivalent [6, Section 2.6].

A *Shannon cover* of a constraint S is a deterministic presentation of S with a smallest number of states. For irreducible constraints we have the following result (see [6, Section 2.6.4]).

Theorem 2.3 *Let S be an irreducible constraint.*

(a) The Shannon cover of S is unique, up to labeled graph isomorphism. In fact, the Shannon cover is the unique presentation of S which is irreducible, deterministic, and reduced.

(b) For any irreducible deterministic presentation G of S, the follower sets of G coincide with the follower sets of the Shannon cover.

We will also make use of the following lemma.

Lemma 2.4 *[7] Let G and H be two irreducible deterministic graphs.*

(a) If $S(H) \subseteq S(G)$, then for every $v \in V_H$ there exists $u \in V_G$ such that $\mathcal{F}_H(v) \subseteq \mathcal{F}_G(u)$.

(b) If there are $v \in V_H$ and $u \in V_G$ such that $\mathcal{F}_H(v) \subseteq \mathcal{F}_G(u)$, then $S(H) \subseteq S(G)$.

2.4 Finite memory

A deterministic graph G is said to have *finite memory* if there is an integer N such that the paths in G of length N that generate the same word all terminate in the same state. The smallest N for which this holds is called the *memory* of G. Each state u in G can be associated with a set $W_G(u)$ of all words in S of length m that are generated in G by paths that lead to that state. The following lemma is easily verified.

Lemma 2.5 *Let G be an irreducible deterministic graph with finite memory m. There is an edge $u \xrightarrow{a} u'$ in G if and only if there are words $\mathbf{w} \in W_G(u)$ and $\mathbf{w}' \in W_G(u')$ such that $\mathbf{w}a = b\mathbf{w}'$ for some $b \in \Sigma(S(G))$ and $\mathbf{w}a \in S(G)$.*

An irreducible constraint S has *finite memory* if its Shannon cover has finite memory. Such constraints are also called *shifts of finite type* [5]. The memory of S is defined as the memory of its Shannon cover.

2.5 Finite-state encoders

Let S be a constraint and n be a positive integer. An (S, n)-*encoder* is a graph $\mathcal{E}$ such that —

- $\mathcal{E}$ is n-regular;
- $S(\mathcal{E}) \subseteq S$; and —
- $\mathcal{E}$ is lossless.

Each row in the adjacency matrix $A_{\mathcal{E}}$ of $\mathcal{E}$ sums up to n. For such matrices, it is known that $\lambda(A_{\mathcal{E}}) = n$ [11, Ch. 1]. Therefore, by Theorem 2.1 we have the following.

Proposition 2.6 *Let $\mathcal{E}$ be an (S, n)-encoder. Then*

$$\mathsf{cap}(S) \;\geq\; \mathsf{cap}(S(\mathcal{E})) \;=\; \log n \;.$$

A *tagged* (S, n)-*encoder* is an (S, n)-encoder $\mathcal{E}$ where the outgoing edges from each state in $\mathcal{E}$ are assigned distinct *input tags* from an alphabet of size n. We will denote by Υ_n a standard alphabet of size n that will be used to tag (S, n)-encoders. Typically, $n = 2^p$ and Υ_n will consist of all binary p-blocks. The notation $u \xrightarrow{s/a} v$ stands for an edge in $\mathcal{E}$ from state u to state v which is labeled $a \in \Sigma(S)$ and tagged by $s \in \Upsilon_n$. Hence, if we

fix some initial state u_0 of a tagged encoder $\mathcal{E}$, then such an encoder defines a mapping from all the unconstrained words over Υ_n to words in S of the same length: an input tag sequence $\mathbf{s} = s_1 s_2 \dots s_\ell$ defines a path π of length ℓ in $\mathcal{E}$ starting at u_0, and the image of $\mathbf{s}$ is the word $\mathbf{w} = w_1 w_2 \dots w_\ell$ that is generated by π. By the losslessness property, knowing the terminal state of π allows to reconstruct the input word, over Υ_n, from the output constrained word in S.

Encoders have finite anticipation or are nested according to whether their underlying *untagged* graphs do.

In virtually all applications, the encoders should have finite anticipation, so that they can be decoded online. Indeed, if $\mathcal{A}(\mathcal{E}) < \infty$, then a state-dependent decoder for $\mathcal{E}$ can be obtained by retracing the edge sequence followed by the encoder. We will adopt in this work a model of a state-dependent decoder through a finite-state look-ahead machine $\mathcal{D}$ which operates as follows. The input sequence to $\mathcal{D}$ is a constrained (channel) word in $S(\mathcal{E})$. At each time slot r, the machine $\mathcal{D}$ gets as input a symbol of that word, but is also allowed to see the upcoming $\mathcal{A}(\mathcal{E})$ channel symbols. Assuming that the decoder knows the state u of the encoder at time r, by the definition of anticipation, the decoder can reconstruct the encoder edge e that was traversed from state u. By simulating the encoder, the decoder can now reconstruct the next encoder state (at time slot $r+1$). Hence, given an initial state u_0 of the encoder and a word $\mathbf{w}$ of length $\ell \geq \mathcal{A}(\mathcal{E})$ that is generated from that state, the decoder can reconstruct (uniquely) the first $\ell - \mathcal{A}(\mathcal{E})$ edges of any path in $\mathcal{E}$ that starts at u_0 and generates $\mathbf{w}$. If we now tag $\mathcal{E}$, then reconstructing those edges also means reconstructing the first $\ell - \mathcal{A}(\mathcal{E})$ input tags that were mapped by $\mathcal{E}$ into $\mathbf{w}$. Such a decoding scheme will be called *state-dependent* decoding.

When the current symbol forms with the upcoming $\mathcal{A}(\mathcal{E})$ symbols a word that cannot be generated from the currently retraced encoder state, then the input is invalid and we will assume (say) that $\mathcal{D}$ halts in this case (in practice, this will be an indication of an error in the decoded sequence). Note that a state-dependent decoder $\mathcal{D}$ requires knowledge of the particular initial encoder state (hence the name); still if $\mathcal{E}$ has finite anticipation, then for *any* initial state chosen, there is a finite-state look-ahead machine $\mathcal{D}$ that decodes $\mathcal{E}$.

The anticipation of an encoder measures the number of channel symbols we need to look-ahead in order to decode the current input tag. We could trade look-ahead decoding with decoding delay. However, for the sake of simplicity, we prefer adopting the convention that the time axes of the tag sequences and the channel symbol sequences are aligned in the encoder and the decoder: the decoder output at time slot r should equal the encoder input at time slot r.

Deterministic encoders have anticipation 0. For such encoders we have the following result taken from [6, Section 3.5].

Proposition 2.7 *Let S be a constraint with a deterministic presentation G. Then there exists a deterministic (S,n)-encoder if and only if there exists such an encoder which is a subgraph of G.*

A *rate $p:q$ finite-state for S* is a tagged $(S^q, 2^p)$-encoder, where we assume that the input tags are binary p-blocks. By Proposition 2.6 and Equation (1) it follows that a rate $p:q$ finite-state for S exists only if $p/q \leq \mathsf{cap}(S)$.

The following coding result, essentially due to Adler, Coppersmith and Hassner [1], is a converse of Proposition 2.6.

Theorem 2.8 *Let S be a constraint and n a positive integer. If $\log n \leq \mathsf{cap}(S)$ there exists an (S,n)-encoder. Furthermore, there is such an encoder with finite anticipation.*

The paper [1] presents an algorithm, known as the *state-splitting algorithm*, which effectively provides an encoder guaranteed by Theorem 2.8.

A tagged (S,n)-encoder is *block decodable* if edges labeled by the same symbol are tagged by the same input tag. A tagged block decodable (S,n)-encoder $\mathcal{E}$ can be decoded through a function $g : \Sigma(S(\mathcal{E})) \to \Upsilon_n$ which maps a label w to the tag assigned to any edge labeled w. We say that g is a *block decoder* for $\mathcal{E}$. Note that the decoding process of a block decodable encoder requires only the knowledge of the current channel symbol; in particular, it does not require knowledge of the initial state of the encoder. The advantage of using block decodable encoders is limiting error propagation and having simple decoding structure. A block decodable encoder is necessarily deterministic.

2.6 Observable encoders and nested encoders

Let $\mathcal{E}_1$ be a tagged (S_1,n_1)-encoder and let $\mathcal{E}_2$ be a tagged (S_2,n_2)-encoder such that $S_1 \subseteq S_2$. We say that $\mathcal{E}_1$ is *(block) observable* from $\mathcal{E}_2$ if there exist a (possibly state-dependent) finite-state look-ahead decoder $\mathcal{D}_2$ and a function $\psi : \Upsilon_{n_2} \to \Upsilon_{n_1}$, such that when ψ is applied to the tag sequence (over Υ_{n_2}) generated by $\mathcal{D}_2$, we obtain a finite-state look-ahead decoder $\mathcal{D}_1$ of $\mathcal{E}_1$ (see Figure 2). We will formally write $\mathcal{D}_1$ as a composition of the form $\psi \circ \mathcal{D}_2$. We allow the existence of the function ψ to depend on a particular choice of a pair of initial states in $\mathcal{E}_1$ and $\mathcal{E}_2$, respectively (but see the discussion on irreducible encoders below). Note that we do assume here that the time axes of the two encoders and their decoders are aligned: the output of $\psi \circ \mathcal{D}_2$ at time slot r should equal the input to $\mathcal{E}_1$ at time slot r. Since $\mathcal{D}_2$ halts on input which is not in $S(\mathcal{E}_2)$, our model implies the containment $S(\mathcal{E}_1) \subseteq S(\mathcal{E}_2)$. On the other hand, as a consequence of our assumption that ψ is a function that does not affect the execution of $\mathcal{D}_2$, the decoder

$\mathcal{D}_1 = \psi \circ \mathcal{D}_2$ may decode input sequences that do not belong to $S(\mathcal{E}_1)$, thereby deviating from our previous convention that the decoder halts in this case. We could allow the indication of sequences that do not belong to $S(\mathcal{E}_1)$ through an "error tag" that would be added to the range of ψ. However, such an indication might make ψ much more complex (but see Example 3.1).

We will assume in our discussion that $\mathcal{E}_1$ and $\mathcal{E}_2$ are irreducible. Note that if an (S, n)-encoder is reducible, then any of its irreducible sinks is an irreducible (S, n)-encoder. As mentioned in Section 1, most of our results will concentrate on deterministic encoders (in particular, block decodable encoders).

Under the assumption of deterministic and irreducible encoders, the existence of a function ψ for a particular pair of initial states in $\mathcal{E}_1$ and $\mathcal{E}_2$ implies that for every state in $\mathcal{E}_1$ which is chosen as an initial state, there is a choice of an initial state in $\mathcal{E}_2$ such that the scheme in Figure 2 holds, with respect to the same function ψ and the same decoders $\mathcal{D}_1$ and $\mathcal{D}_2$ (except that now the decoders operate assuming the new initial states). Indeed, suppose that $\langle u_\psi, v_\psi \rangle$ is a particular pair of initial states that corresponds to ψ, and let u be some other state in $\mathcal{E}_1$. Since $\mathcal{E}_1$ is irreducible, there is a path in $\mathcal{E}_1$ from u_ψ to u. Let $\mathbf{w}$ be the word generated by that path. Now, the word $\mathbf{w}$ must also be generated in $\mathcal{E}_2$ by a path that starts at v_ψ and terminates in some state v. We can now use $\langle u, v \rangle$ as an alternate initial pair of states. (On the other hand, there may be states in $\mathcal{E}_2$ with which no state in $\mathcal{E}_1$ forms an initial pair of states consistent with ψ.)

If $\mathcal{E}_1$ is observable from $\mathcal{E}_2$ then $S(\mathcal{E}_1) \subseteq S(\mathcal{E}_2)$ and, so, $\mathsf{cap}(S(\mathcal{E}_1)) \leq \mathsf{cap}(S(\mathcal{E}_2))$. By Proposition 2.6 we have $\mathsf{cap}(S(\mathcal{E}_i)) = \log n_i$; so, $n_1 \leq n_2$. The case $n_1 = n_2$ is vacuous since it allows us to choose $\mathcal{E}_1 = \mathcal{E}_2$ in the first place. We will therefore assume that n_1 is strictly smaller than n_2. In practice, $n_1 = 2^{p_1}$, $n_2 = 2^{p_2}$, and S_1 and S_2 are qth powers of some constraints S_1' and S_2', respectively. This means that $\mathcal{E}_1$ is a rate $p_1 : q$ finite-state encoder for S_1' and $\mathcal{E}_2$ is a rate $p_2 : q$ finite-state encoder for S_2', where $S_1' \subseteq S_2'$ and $p_1 < p_2$.

3 Examples

We provide here several examples that demonstrate some of the terms we defined in Section 2.

Example 3.1 The capacity of the $(2, 3)$-RLL constraint is approximately 0.2878. A rate $1 : 4$ finite-state encoder for $S_{(2,3)}$ is shown in Figure 3. This encoder, denoted $\mathcal{E}_1$, is a block decodable $(S^4_{(2,3)}, 2)$-encoder with a block decoder $g_1 : \Sigma(S(\mathcal{E}_1)) \to \{0, 1\}$ which satisfies

$$g_1(0001) = g_1(0100) = 0 \quad \text{and} \quad g_1(0010) = g_1(1001) = 1$$

(we specify the values of g_1 only for words that label edges in $\mathcal{E}_1$).

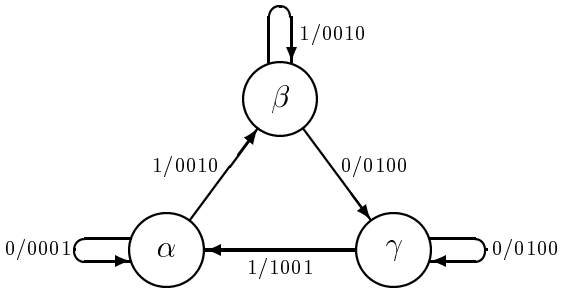

Figure 3: Rate $1:4$ finite-state encoder for $S_{(2,3)}$.

The capacity of the $(1,3)$-RLL constraint is approximately 0.5515, and a rate $2:4$ finite-state encoder for $S_{(1,3)}$ is shown in Figure 4. This encoder, denoted $\mathcal{E}_2$, is a block decodable $(S^4_{(1,3)}, 4)$-encoder with a block decoder $g_2 : \Sigma(S(\mathcal{E}_2)) \to \{00, 01, 10, 11\}$ which satisfies

$$g_2(0001) = g_2(1010) = 00,\ g_2(0010) = g_2(1001) = 01,\ g_2(0100) = 10, \text{ and } g_2(0101) = 11.$$

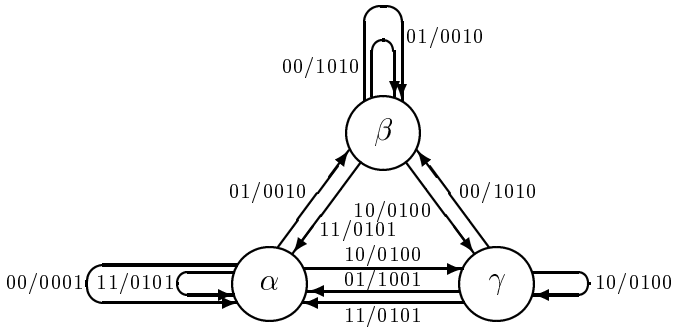

Figure 4: Rate $2:4$ finite-state encoder for $S_{(1,3)}$.

It is easy to see that $\mathcal{E}_1$ is nested in $\mathcal{E}_2$. Furthermore, $\mathcal{E}_1$ is observable from $\mathcal{E}_2$. Indeed, let $\psi : \{00, 01, 10\} \to \{0, 1\}$ be given by

$$\psi(00) = \psi(10) = 0 \quad \text{and} \quad \psi(01) = 1\ .$$

Then, $g_1(\mathbf{w}) = \psi(g_2(\mathbf{w}))$ for every $\mathbf{w} \in \{0001, 0010, 0100, 1001\}$. In principle, the function ψ needs to be defined also for the input tag 11 of $\mathcal{E}_2$. However, g_2 never produces that tag for the labels of $\mathcal{E}_1$. Hence, in practice, we can define $\psi(11) = ?$ to indicate an error while decoding sequences generated by $\mathcal{E}_1$.

We can simplify the encoder $\mathcal{E}_2$ by eliminating state β in $\mathcal{E}_2$ and redirecting all its incoming edges (excluding self-loops) into state γ. This yields a smaller $(S^4_{(1,3)}, 4)$-encoder $\mathcal{E}'_2$, which can be decoded by the same block decoder g_2. The encoder $\mathcal{E}_1$ is therefore observable from $\mathcal{E}'_2$ even though it is not nested in it. □

Example 3.2 Let S_1 be the constraint presented by the graph $\mathcal{E}_1$ in Figure 5. It is easy to see that $\mathsf{cap}(S_1) = \log 2$ and that $\mathcal{E}_1$ is in fact a deterministic $(S_1, 2)$-encoder. Furthermore, we can make $\mathcal{E}_1$ block decodable by assigning one tag of Υ_2 to the edges labeled a and c, and a second tag to the edges labeled b and d; namely, Υ_2 induces the partition $\{a, c\}, \{b, d\}$ on $\Sigma(S_1)$. In fact, this is essentially the only assignment of tags to labels—and, thereby, to edges—that can make $\mathcal{E}_1$ block decodable: the edges labeled by $\{a, c\}$ must all be assigned with the same tag of Υ_2, and the edges labeled by $\{b, d\}$ must be assigned with the other tag. Assuming that $\Upsilon_2 = \{0, 1\}$, the respective essentially unique block decoder is a function $g_1 : \{a, b, c, d\} \to \Upsilon_2$, where

$$g_1(a) = g_1(c) = 0 \quad \text{and} \quad g_1(b) = g_1(d) = 1 \ . \tag{2}$$

By Theorem 2.3, every irreducible deterministic $(S_1, 2)$-encoder can be reduced through state merging to $\mathcal{E}_1$. It follows from this that for every irreducible deterministic $(S_1, 2)$-encoder there is a unique assignment of tags to labels that makes such an encoder block decodable (in particular, this applies to the irreducible sinks of reducible deterministic $(S_1, 2)$-encoders).

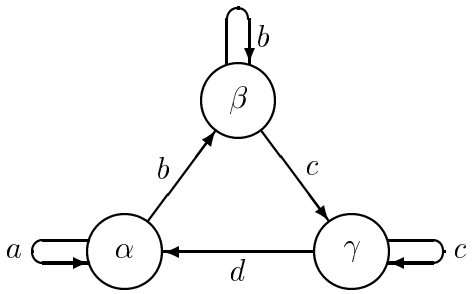

Figure 5: Graph $\mathcal{E}_1$ for Example 3.2.

Let S_2 be the constraint presented by the graph $\mathcal{E}_2$ in Figure 6. Here $\mathsf{cap}(S_2) = \log 3$ and $\mathcal{E}_2$ is a deterministic $(S_2, 3)$-encoder. We can make $\mathcal{E}_2$ block decodable in a unique manner by partitioning $\Sigma(S_2)$ into $\{a, d\}, \{b\}, \{c\}$ and tagging the edges of $\mathcal{E}_2$ accordingly with the elements of Υ_3. Every irreducible deterministic $(S_2, 3)$-encoder can be made block decodable by an essentially unique tag assignment to the edge labels, and the respective block decoder is the function $g_2 : \{a, b, c, d\} \to \Upsilon_3$, where

$$g_2(a) = g_2(d) = 0 \ , \quad g_2(b) = 1 \ , \quad \text{and} \quad g_2(c) = 2 \tag{3}$$

(assuming that $\Upsilon_3 = \{0, 1, 2\}$).

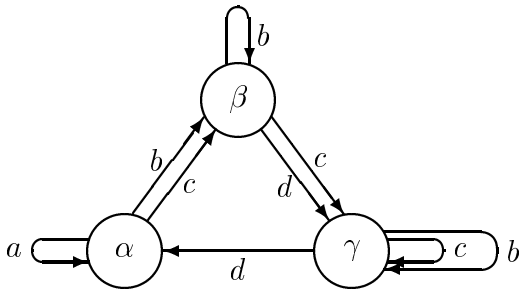

Figure 6: Graph $\mathcal{E}_2$ for Example 3.2.

It is straightforward to see that (the untagged version of) $\mathcal{E}_1$ is nested in (the untagged version of) $\mathcal{E}_2$. By Proposition 4.1 that we prove in Section 4 it will thus follow that there exists a deterministic $(S_1, 2)$-encoder which is observable from a deterministic $(S_2, 3)$-encoder.

However, in our example, it is impossible to have both observability and block decodability. Indeed, let $\mathcal{E}_1'$ be a block decodable $(S_1, 2)$-encoder with the block decoder g_1 defined by (2), and let $\mathcal{E}_2'$ be a block decodable $(S_2, 3)$-encoder with the block decoder g_2 defined by (3). Suppose to the contrary that $\mathcal{E}_1'$ were observable from $\mathcal{E}_2'$. Then there had to be a function $\psi : \Upsilon_3 \to \Upsilon_2$ such that $g_1(w) = \psi(g_2(w))$ for all $w \in \{a, b, c, d\}$. However, this is impossible, since $g_2(a) = g_2(d)$ whereas $g_1(a) \neq g_1(d)$. □

Example 3.3 A two-state $(S^8_{(1,3)}, 17)$-encoder, denoted $\mathcal{E}$, is shown in Figure 7. Each row in the table corresponds to an input tag, and each column corresponds to one of the encoder states, α or β. The entry at row s and column α (respectively, β) in the table contains the label and terminal state of the outgoing edge from state α (respectively, β) that is tagged by s. It can be readily verified that $\mathcal{E}$ is block decodable. The capacity of

the $(1,3)$-RLL constraint is approximately 0.5515, whereas the 'equivalent rate' of $\mathcal{E}$ is $(\log 17)/8 \approx 0.5109$.

	label, terminal state of edges	
input tag	from state α	from state β
0	$00100101, \alpha$	$10010101, \alpha$
1	$00101001, \alpha$	$10100101, \alpha$
2	$00101010, \beta$	$10010010, \beta$
3	$01001001, \alpha$	$10101001, \alpha$
4	$01001001, \alpha$	$01001001, \alpha$
5	$01010010, \beta$	$01010010, \beta$
6	$01010101, \alpha$	$01010101, \alpha$
7	$00010001, \alpha$	$10001001, \alpha$
8	$00010010, \beta$	$10001010, \beta$
9	$00010100, \beta$	$10010100, \beta$
10	$00010101, \alpha$	$10010001, \alpha$
11	$00100010, \beta$	$10100010, \beta$
12	$00100100, \beta$	$10100100, \beta$
13	$01000100, \beta$	$01000100, \beta$
14	$01000101, \alpha$	$01000101, \alpha$
15	$01010001, \alpha$	$01010001, \alpha$
16	$01010100, \beta$	$01010100, \beta$

Figure 7: $(S_{(1,2)}, 7)$-encoder nested in an $(S_{(1,3)}, 17)$-encoder $\mathcal{E}$.

The first seven rows in Figure 7 define a two-state $(S^8_{(1,2)}, 7)$-encoder which is nested in $\mathcal{E}$. Here $(\log 7)/8 \approx 0.3509$, compared to the capacity of the $(1,2)$-RLL constraint which is approximately 0.4057 (Proposition 2.7 implies that there are no deterministic $(S^8_{(1,2)}, 8)$-encoders). □

Example 3.4 The capacity of the $(2,10)$-RLL constraint is approximately 0.5418, and there exist rate $8:16$ block decodable encoders for $S_{(2,10)}$. An example of such an encoder is the one used in the DVD standard [4].

The capacity of the $(3,10)$-RLL constraint is approximately 0.4460. By Theorem 2.8 there exist rate $7:16$ encoders for $S_{(3,10)}$. However, by Proposition 2.7 it can be verified that the largest integer n for which there is a deterministic $(S^{16}_{(3,10)}, n)$-encoder is $n = 84$, so none of the rate $7:16$ encoders for $S_{(3,10)}$ is deterministic. On the other hand, there is a particular construction of a rate $6:16$ four-state block decodable encoder

for $S_{(3,10)}$, denoted $\mathcal{E}_{(3,10)}$, which is observable from a particular rate $8:16$ four-state block decodable encoder for $S_{(2,10)}$. This rate $8:16$ encoder, which we denote by $\mathcal{E}_{(2,10)}$, is different from the one presented in [4]; still, like the latter, it also possesses certain properties that allow for DC control in addition to producing sequences that satisfy the $(2,10)$-RLL constraint (see [3, Section 2.5]). Those DC-control properties carry over also to $\mathcal{E}_{(3,10)}$. The function ψ that is associated with $\mathcal{E}_{(2,10)}$ and $\mathcal{E}_{(3,10)}$ just truncates the two trailing bits of the 8-block input tags of $\mathcal{E}_{(2,10)}$. The details of the construction of $\mathcal{E}_{(2,10)}$ and $\mathcal{E}_{(3,10)}$ are contained in [2] and will be presented in a future work.

4 Nesting and observability

In this section, we show that observability of encoders is equivalent to the nesting property if the encoders are deterministic and irreducible.

The following result shows that nesting implies observability even under much weaker assumptions.

Proposition 4.1 *Let $\mathcal{E}_1$ be an untagged (S_1, n_1)-encoder and $\mathcal{E}_2$ be an untagged (S_2, n_2)-encoder such that $\mathcal{E}_1 \subseteq \mathcal{E}_2$ and $\mathcal{A}(\mathcal{E}_2) < \infty$. Then $\mathcal{E}_1$ and $\mathcal{E}_2$ can be tagged so that $\mathcal{E}_1$ is observable from $\mathcal{E}_2$.*

Proof. Without loss of generality assume that $\Upsilon_{n_1} \subseteq \Upsilon_{n_2}$. Let $\psi : \Upsilon_{n_2} \to \Upsilon_{n_1}$ be a mapping that is one-to-one (and onto) when restricted to the domain Υ_{n_1}. If u is a state in $\mathcal{E}_1$ (and in $\mathcal{E}_2$), we tag the outgoing edges from u that belong to $\mathcal{E}_1$ (and $\mathcal{E}_2$) by Υ_{n_1}, and the remaining outgoing edges from u in $\mathcal{E}_2$ are tagged by $\Upsilon_{n_2} \setminus \Upsilon_{n_1}$. If u is a state in $\mathcal{E}_2$ but not in $\mathcal{E}_1$, then the outgoing edges from u are tagged arbitrarily by Υ_{n_2}. Since $\mathcal{E}_2$ has finite anticipation, there is a (possibly state-dependent) finite-state look-ahead decoder $\mathcal{D}_2$ of $\mathcal{E}_2$. Assuming without loss of generality that the initial state of $\mathcal{E}_2$ is a state in $\mathcal{E}_1$, the composition $\mathcal{D}_1 = \psi \circ \mathcal{D}_2$ is a finite-state look-ahead decoder of $\mathcal{E}_1$. $\square$

Referring to the notations in the last proof, when $n_1 = 2^{p_1}$ and $n_2 = 2^{p_2}$, we can let Υ_{n_2} be the set of all binary blocks of length p_2, and let Υ_{n_1} be the set of all binary p_2-blocks with some fixed suffix of length $p_2 - p_1$. In practice, this suffix can be deleted from each tag in $\mathcal{E}_1$, in which case the function ψ simply truncates the trailing $p_2 - p_1$ bits.

Let G and H be two graphs. We define the *fiber product* of G and H as the graph $G * H$, where

$$V_{G*H} \;=\; V_G \times V_H \;=\; \{\langle u, u' \rangle \,|\, u \in V_G,\; u' \in V_H\}\;,$$

and $\langle u, u' \rangle \stackrel{a}{\longrightarrow} \langle v, v' \rangle$ is an edge in $G * H$ if and only if there are edges $u \stackrel{a}{\longrightarrow} v$ and $u' \stackrel{a}{\longrightarrow} v'$ in G and H, respectively. It is easy to verify that for every $\langle u, u' \rangle \in V_{G*H}$ we

have

$$\mathcal{F}_{G*H}(\langle u, u' \rangle) \;=\; \mathcal{F}_G(u) \cap \mathcal{F}_H(u') \;. \tag{4}$$

Hence, $G * H$ presents the intersection of the constraints defined by G and H, namely, $S(G * H) = S(G) \cap S(H)$.

Lemma 4.2 *Let S_1 and S_2 be irreducible constraints such that $S_1 \subseteq S_2$. There exist irreducible deterministic graphs H_1 and H_2 (not necessarily reduced) such that $S(H_1) = S_1$, $S(H_2) = S_2$, and $H_1 \subseteq H_2$.*

Proof. Denote by G_1 and G_2 the Shannon covers of S_1 and S_2, respectively, and let $G_1 * G_2$ be the fiber product of G_1 and G_2. Since $S(G_1 * G_2) = S_1 \cap S_2 = S_1$, every irreducible component of $G_1 * G_2$ generates a constraint that is contained in S_1. Combining this with Lemma 2.2, there is an irreducible component H_1 in $G_1 * G_2$ such that $S_1 = S(H_1)$.

We now follow [9] and [10] and construct a deterministic graph $H = (V_H, E_H, L_H)$ that contains H_1 as a subgraph, as follows. Let $V_H = V_{H_1} \cup V$, where

$$V \;=\; \{\, \langle \phi, v \rangle \;:\; v \in V_{G_2} \,\} \;,$$

and let $E_H = E_{H_1} \cup E$, where the edges of E_{H_1} in E_H inherit their labeling from H_1, and E is defined as follows:

- For every state $\langle u, v \rangle \in V_{H_1}$, we endow H with an edge $\langle u, v \rangle \xrightarrow{a} \langle \phi, v' \rangle$ if $v \xrightarrow{a} v'$ is an edge in G_2 and there is no edge $\langle u, v \rangle \xrightarrow{a} \langle u'', v' \rangle$ in H_1 for any $u'' \in V_{G_1}$.
- For every $\langle \phi, v \rangle \in V$ we endow H with an edge $\langle \phi, v \rangle \xrightarrow{a} \langle u', v' \rangle$ if $v \xrightarrow{a} v'$ is an edge in G_2 and u' is the first state u'' in V_{G_1}, if any, such that $\langle u'', v' \rangle \in V_{H_1}$ (here we assume some ordering on V_{G_1}). If no such u'' exists, then $u' = \phi$.

By construction, there is a path

$$\langle u_0, v_0 \rangle \xrightarrow{a_1} \langle u_1, v_1 \rangle \xrightarrow{a_2} \ldots \xrightarrow{a_\ell} \langle u_\ell, v_\ell \rangle \tag{5}$$

in H only if there is a path

$$v_0 \xrightarrow{a_1} v_1 \xrightarrow{a_2} \ldots \xrightarrow{a_\ell} v_\ell \tag{6}$$

in G_2. Conversely, for every path (6) in G_2 there are $u_0, u_1, \ldots, u_\ell$ such that (5) is a path in H. Therefore, $S(H) = S_2$.

Since H_1 is an irreducible subgraph of H, there is a unique irreducible component H_2 of H that contains H_1 as a subgraph. Clearly, $S(H_2) \subseteq S(H) = S_2$. We next show that $S_2 \subseteq S(H_2)$, thereby establishing the equality $S(H_2) = S_2$.

Let $\mathbf{w}$ be a word in S_2 that is generated in G_2 by a path that starts at state v and terminates in state v_0. Let v' be a state in G_2 such that $\langle u', v' \rangle \in V_{H_1}$ for some $u' \in V_{G_1}$. Since G_2 is irreducible, there is a word $\mathbf{z}$ that can be generated in G_2 by a path that starts at v' and terminates in v. By construction of H, there is a path π in H that generates $\mathbf{zw}$, starting at $\langle u', v' \rangle$ and terminating in $\langle u_0, v_0 \rangle$, for some $u_0 \in V_{G_1} \cup \{\phi\}$. We now show that there is a path in H that starts at $\langle u_0, v_0 \rangle$ and terminates in V_{H_1}; this, in turn, will imply that $\langle u', v' \rangle$ and $\langle u_0, v_0 \rangle$ belong to the same irreducible component of H and, as such, the path π is entirely contained in H_2. Let $\mathbf{z}'$ be a word that is generated in G_2 by a path that starts at v_0 and terminates in v'. Then, $\mathbf{z}'$ is also generated in H by a path

$$\langle u_0, v_0 \rangle \longrightarrow \langle u_1, v_1 \rangle \longrightarrow \ldots \longrightarrow \langle u_\ell, v_\ell \rangle \ ,$$

where $v_\ell = v'$. We claim that there must be an index $j \in \{0, 1, \ldots, \ell\}$ such that $u_j \in V_{G_1}$ (i.e., $\langle u_j, v_j \rangle \in V_{H_1}$). Indeed, if no such index $j < \ell$ exists, then, in particular, $u_{\ell-1} = \phi$; but there is a state $u' \in V_{G_1}$ such that $\langle u', v_\ell \rangle = \langle u', v' \rangle$ is in V_{H_1}; so, by construction of H we have $\langle u_\ell, v_\ell \rangle \in V_{H_1}$. □

Recall that, by definition, if $\mathcal{E}_1$ is observable from $\mathcal{E}_2$, then $S(\mathcal{E}_1) \subseteq S(\mathcal{E}_2)$. Hence, by the following result we will get that observability implies nesting in the irreducible deterministic case.

Proposition 4.3 *Let $\mathcal{E}_1$ be an irreducible deterministic (S_1, n_1)-encoder and let $\mathcal{E}_2$ be an irreducible deterministic (S_2, n_2)-encoder such that $S(\mathcal{E}_1) \subseteq S(\mathcal{E}_2)$. Then there exists an irreducible deterministic (S_1, n_1)-encoder $\mathcal{E}_1'$ that is nested in an irreducible deterministic (S_2, n_2)-encoder $\mathcal{E}_2'$ such that $S(\mathcal{E}_1) = S(\mathcal{E}_1')$ and $S(\mathcal{E}_2) = S(\mathcal{E}_2')$.*

Proof. Without loss of generality we can assume that $\mathcal{E}_1$ and $\mathcal{E}_2$ are reduced; i.e., they are the Shannon covers of $S(\mathcal{E}_1)$ and $S(\mathcal{E}_2)$, respectively. Apply Lemma 4.2 with $S_1 \leftarrow S(\mathcal{E}_1)$ and $S_2 \leftarrow S(\mathcal{E}_2)$ to obtain $\mathcal{E}_1' \leftarrow H_1$ and $\mathcal{E}_2' \leftarrow H_2$. □

Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be encoders that satisfy the conditions in Proposition 4.3 and suppose further that $\mathcal{E}_1$ is observable from $\mathcal{E}_2$ through a decoding scheme that comprises decoders $\mathcal{D}_1$ and $\mathcal{D}_2$ as in Figure 2. Based on the proof of Lemma 4.2, we can obtain a respective decoding scheme for $\mathcal{E}_1'$ and $\mathcal{E}_2'$ as in Figure 2 by slightly modifying $\mathcal{D}_1$ and $\mathcal{D}_2$. However, to this end, we need to elaborate more on the construction contained in the proof of Lemma 4.2, and we do this next.

Let G_1, G_2, H_1, and H_2 be as in the proof of Lemma 4.2, and let $\langle u, v \rangle$ be a state in H_1. We claim that

$$\mathcal{F}_{H_1}(\langle u, v \rangle) \ = \ \mathcal{F}_{G_1}(u) \ . \tag{7}$$

By the construction of H_1 (in particular, since H_1 is irreducible and deterministic), it suffices to prove (7) for a particular state $\langle u, v\rangle$. Let $u \in V_{G_1}$ be such that $\mathcal{F}_{G_1}(u)$ does not contain a follower set of any other state in G_1. Now, by Theorem 2.3(b), the sets of follower sets of the states in G_1 and H_1 are the same; so there must be a state u' in G_1 such that $\mathcal{F}_{H_1}(\langle u, v\rangle) = \mathcal{F}_{G_1}(u')$. On the other hand, $\mathcal{F}_{H_1}(\langle u, v\rangle) \subseteq \mathcal{F}_{G_1}(u)$. By the choice of u we must therefore have $u = u'$. By a similar line of argument we can show that for every state $\langle u, v\rangle$ in H_2 we have

$$\mathcal{F}_{H_2}(\langle u, v\rangle) \;=\; \mathcal{F}_{G_2}(v)\,. \tag{8}$$

We now return to the encoders of Proposition 4.3, with $G_1 \leftarrow \mathcal{E}_1$, $G_2 \leftarrow \mathcal{E}_2$, $\mathcal{E}'_1 \leftarrow H_1$, and $\mathcal{E}'_2 \leftarrow H_2$. From (7) it follows that there is a one-to-one label-preserving mapping from the outgoing edges from state u in $\mathcal{E}_1$ onto the outgoing edges from state $\langle u, v\rangle$ in $\mathcal{E}'_1$. Furthermore, the terminal states of an edge in $\mathcal{E}_1$ and its image in $\mathcal{E}'_1$ are follower-set equivalent. Such a mapping induces a tagging of the edges of $\mathcal{E}'_1$ from the edges of $\mathcal{E}_1$. A similar mapping exists by (8) from the edges of $\mathcal{E}_2$ to the edges of $\mathcal{E}'_2$, and we use that mapping to define a tagging of $\mathcal{E}'_2$.

With such tagging, every state-dependent (in particular, state-independent) decoder $\mathcal{D}_1$ of $\mathcal{E}_1$ can be easily transformed into a decoder $\mathcal{D}'_1$ of $\mathcal{E}'_1$: the decoder $\mathcal{D}'_1$, reconstructing an input from a state $\langle u, v\rangle$ of $\mathcal{E}'_1$, will act the same way as the decoder $\mathcal{D}_1$ does while reconstructing an input from state u. Similarly, every finite-state look-ahead decoder $\mathcal{D}_2$ of $\mathcal{E}_2$ can be made into a decoder $\mathcal{D}'_2$ of $\mathcal{E}'_2$, where inputs from state $\langle u, v\rangle$ are treated as if they are from state v.

This, in principle, may suggest that if $\mathcal{D}_1 = \psi \circ \mathcal{D}_2$, then $\mathcal{D}'_1 = \psi \circ \mathcal{D}'_2$. Recall, however, that ψ assumes some pair of initial states u_ψ and v_ψ in $\mathcal{E}_1$ and $\mathcal{E}_2$, respectively, and $\mathcal{F}_{\mathcal{E}_1}(u_\psi) \subseteq \mathcal{F}_{\mathcal{E}_2}(v_\psi)$. Now, if $\langle u_\psi, v_\psi\rangle$ is a state in $\mathcal{E}'_1$, then, indeed, we can start the encoding of both $\mathcal{E}'_1$ and $\mathcal{E}'_2$ at that state, in which case we will have $\mathcal{D}'_1 = \psi \circ \mathcal{D}'_2$.

Consider now the case where $\langle u_\psi, v_\psi\rangle$ is not in $V_{\mathcal{E}'_1}$. Since $\langle u_\psi, v_\psi\rangle$ is a state in $\mathcal{E}_1 * \mathcal{E}_2$, there must be a path in $\mathcal{E}_1 * \mathcal{E}_2$ from $\langle u_\psi, v_\psi\rangle$ to a state $\langle u'_\psi, v'_\psi\rangle$ that belongs to an irreducible sink $\mathcal{E}''_1$ of $\mathcal{E}_1 * \mathcal{E}_2$. Now, from $\mathcal{F}_{\mathcal{E}_1}(u_\psi) \subseteq \mathcal{F}_{\mathcal{E}_2}(v_\psi)$ we have $\mathcal{F}_{\mathcal{E}_1}(u'_\psi) \subseteq \mathcal{F}_{\mathcal{E}_2}(v'_\psi)$. This implies by (4) the equality $\mathcal{F}_{\mathcal{E}''_1}(\langle u'_\psi, v'_\psi\rangle) = \mathcal{F}_{\mathcal{E}_1}(u'_\psi)$; so, by Lemma 2.4(b) we have $S(\mathcal{E}''_1) = S(\mathcal{E}_1)$. Thus, in the construction of Lemma 4.2, we can take the sink $\mathcal{E}''_1$ as our (S_1, n_1)-encoder $\mathcal{E}'_1$. Furthermore, we can take u'_ψ and v'_ψ as an alternate pair of initial states in $\mathcal{E}_1$ and $\mathcal{E}_2$, respectively, while using the decoder $\mathcal{D}_2$ for the latter and $\mathcal{D}_1 = \psi \circ \mathcal{D}_2$ for the former. At this point, we got back to the case where $\langle u'_\psi, v'_\psi\rangle$ is in $V_{\mathcal{E}'_1}$.

We point out that the encoders $\mathcal{E}'_1$ and $\mathcal{E}'_2$ that we obtain in Proposition 4.3 are not necessarily reduced, even when the original encoders $\mathcal{E}_1$ and $\mathcal{E}_2$ are. Of course, for practical applications, there might be no advantage having nested encoders if the nesting prop-

erty requirement implies more complex encoders (e.g., increasing the number of states). Nevertheless, from what we have just shown, it follows that the original decoders of $\mathcal{E}_1$ and $\mathcal{E}_2$ (as in Figure 2) can be applied to decode the new nested encoders $\mathcal{E}_1'$ and $\mathcal{E}_2'$.

A natural question to ask is whether Proposition 4.3 can be generalized to nondeterministic encoders. This question remains still open. However, we show in the Appendix an example of an irreducible deterministic (S_1, n_1)-encoder $\mathcal{E}_1$ that is observable from an irreducible (S_2, n_2)-encoder $\mathcal{E}_2$ with anticipation 1; on the other hand, we also show that if $\mathcal{E}_1'$ is an (S_1, n_1)-encoder that is nested in an (S_2, n_2)-encoder $\mathcal{E}_2'$, then $\mathcal{E}_2'$ must have anticipation at least 2. We also provide a pair of nested encoders $\mathcal{E}_1' \subseteq \mathcal{E}_2'$ where $\mathcal{E}_2'$ has infinite anticipation.

5 Construction of deterministic nested encoders

5.1 (G, n)-subgraphs and approximate eigenvectors

Let G be a graph and let n be a positive integer. A (G, n)-*subgraph* is a subgraph of G that has minimum degree at least n. Clearly, for a given graph G, there are values of n for which (G, n)-subgraphs do not exist. A necessary condition for having a (G, n)-subgraph is $n \leq \lambda(A_G)$ (see below).

In case (G, n)-subgraphs exist, then there is a unique *maximal* (G, n)-subgraph, namely a (G, n)-subgraph that is not a proper subgraph of any other (G, n)-subgraph. Indeed, if we take the union of the sets of states and the union of the sets of edges of two (G, n)-subgraphs, the resulting graph is also a (G, n)-subgraph.

A maximal (G, n)-subgraph can be found through the following algebraic tool, which is commonly used in connection with finite-state encoders. An (A_G, n)-*approximate eigenvector* is a nonnegative nonzero integer vector $\mathbf{x}$ such that $A_G\mathbf{x} \geq n\mathbf{x}$, where the inequality holds component by component. There exist (A_G, n)-approximate eigenvectors if and only if $n \leq \lambda(A_G)$ [6, Section 3.1.3]. The set of all (A_G, n)-approximate eigenvectors with entries restricted to $\{0, 1\}$ will be denoted by $\mathcal{B}(A_G, n)$.

For every (G, n)-subgraph H we can associate an indicator vector $\mathbf{x}_H \in \mathcal{B}(A_G, n)$ of the set V_H as a subset of V_G. In fact, the mapping $H \mapsto \mathbf{x}_H$ is *onto* $\mathcal{B}(A_G, n)$ (but not necessarily one-to-one: (G, n)-subgraphs that are mapped to the same vector $\mathbf{x}$ have the same sets of states; however, their sets of edges might be different). Since (A_G, n)-approximate eigenvectors exist if and only if $n \leq \lambda(A_G)$, a necessary (but not sufficient) condition for having a nonempty $\mathcal{B}(G, n)$ is $n \leq \lambda(A_G)$. It is known that $\mathcal{B}(A_G, n)$, if nonempty, contains a unique maximal vector $\mathbf{x}_{\max}$; i.e., $\mathbf{x}_{\max}$ is a vector in $\mathcal{B}(A_G, n)$ such that $\mathbf{x} \in \mathcal{B}(A_G, n)$ implies $\mathbf{x} \leq \mathbf{x}_{\max}$, where the inequality holds component by

component [6, Section 3.1.4]. That vector $\mathbf{x}_{\max}$ is the indicator vector of the set of states of the unique maximal (G, n)-subgraph: the latter is the subgraph of G induced by the set of states indicated by $\mathbf{x}_{\max}$. The vector $\mathbf{x}_{\max}$ can be found using Franaszek's algorithm [6, Section 3.1.4], when given as input the matrix A_G, the integer n, and the all-one vector $\mathbf{1}$.

An irreducible (G, n)-subgraph is a (G, n)-subgraph that is also irreducible. A (G, n)-*component* is an irreducible (G, n)-subgraph that is not a proper subgraph of any other irreducible (G, n)-subgraph.

The (G, n)-components can be found as follows. Let $H_{\max}$ be the maximal (G, n)-subgraph, if any. Then every (G, n)-component is a subgraph of $H_{\max}$. Furthermore, since every (G, n)-component is irreducible, each is a subgraph of some irreducible component of $H_{\max}$. Let $H^{(1)}, H^{(2)}, \ldots, H^{(t)}$ denote the irreducible components of $H_{\max}$, sorted by their minimum degrees in decreasing order, and let s be the last index s for which $H^{(s)}$ has minimum degree at least n. Note that if $H_{\max}$ exists, then all its irreducible sinks will have minimum degree at least n; so, s is well-defined (it is at least 1). The graphs $H^{(1)}, H^{(2)}, \ldots, H^{(s)}$ are (G, n)-components, and the remaining (G, n)-components, if any, will be obtained by finding recursively the $(H^{(i)}, n)$-components for $s < i \leq t$.

5.2 Conditions for having deterministic nested encoders

In this section we prove the following result.

Theorem 5.1 *Let S_1 and S_2 be two irreducible constraints where $S_1 \subseteq S_2$, and let n_1 and n_2 be positive integers. Denote by G_1 and G_2 the Shannon covers of S_1 and S_2, respectively. Then there exists an irreducible deterministic (S_1, n_1)-encoder that is nested in an irreducible deterministic (S_2, n_2)-encoder if and only if there is a (G_2, n_2)-component H_2 for which there exists a $(G_1 * H_2, n_1)$-component.*

The proof of Theorem 5.1 is constructive in the sense that it implies an algorithm for finding the nested encoders. We prove the theorem using the following two lemmas. The first lemma is a stronger version of Proposition 2.7.

Lemma 5.2 *Let G be a deterministic graph which presents a constraint S and let $\mathcal{E}$ be an irreducible deterministic (S, n)-encoder. Then for some (G, n)-component G' of G, $S(\mathcal{E}) \subseteq S(G')$.*

Proof. We construct an irreducible (G, n)-subgraph H such that $S(\mathcal{E}) \subseteq S(H)$. The graph H, in turn, must be a subgraph of some (G, n)-component.

For a state $u \in V_G$, denote by $Z(u)$ the set of all states $v \in V_{\mathcal{E}}$ such that $\mathcal{F}_{\mathcal{E}}(v) \subseteq \mathcal{F}_G(u)$. By Lemmas 2.2 and 2.4(a), for every $v \in V_{\mathcal{E}}$ there is at least one state $u \in V_G$ such that $v \in Z(u)$. In particular, at least one of the sets $Z(u)$ is nonempty.

The graph H is defined as an irreducible sink of the following graph H'. The states of H' are all the nonempty subsets $Z(u)$, $u \in V_G$. We endow H' with an edge $Z(u) \xrightarrow{a} Z(u')$ if and only if $u \xrightarrow{a} u'$ is an edge in G and there is $v \in Z(u)$ and $v' \in V_{\mathcal{E}}$ such that $v \xrightarrow{a} v'$ is an edge in $\mathcal{E}$ (note that in this case, $\mathcal{F}_{\mathcal{E}}(v') \subseteq \mathcal{F}_G(u')$ and, so, v' must be in $Z(u')$).

By construction, H is isomorphic to some irreducible subgraph in G (with state $Z(u)$ in H corresponding to state u in G). Furthermore, the out-degree of every state $Z(u)$ in H is at least the out-degree, in $\mathcal{E}$, of any $\mathcal{E}$-state $v \in Z(u)$. Hence, H is isomorphic to some irreducible (G, n)-subgraph.

It remains to show that $S(\mathcal{E}) \subseteq S(H)$. Let $Z(u)$ be a state in H and let v be a particular $\mathcal{E}$-state in $Z(u)$. By Lemma 2.4(b), it suffices to show that every word that can be generated in $\mathcal{E}$ by a path starting at state v, can also be generated in H' by a path starting at state $Z(u)$. Let

$$v_0 \xrightarrow{a_1} v_1 \xrightarrow{a_2} \ldots \xrightarrow{a_\ell} v_\ell$$

be a path in $\mathcal{E}$ that starts in $v_0 = v$ and generates a word $a_1 a_2 \ldots a_\ell$. We construct a path

$$\pi \ : \ Z(u_0) \xrightarrow{a_1} Z(u_1) \xrightarrow{a_2} \ldots \xrightarrow{a_\ell} Z(u_\ell)$$

in H' with $v_i \in Z(u_i)$ inductively as follows. We let $Z(u_0)$ be the state $Z(u)$ in H such that $v \in Z(u)$. Next, suppose there is a path in H that consists of the first $k < \ell$ edges in π with $v_i \in Z(u_i)$ for $i = 0, 1, \ldots, k$. Since v_k is in $Z(u_k)$, we have $\mathcal{F}_{\mathcal{E}}(v_k) \subseteq \mathcal{F}_G(u_k)$. Hence, there must be an outgoing edge from u_k in G labeled a_{k+1}. Define u_{k+1} to be the terminal state of that edge. By construction, the edge $Z(u_k) \xrightarrow{a_k} Z(u_{k+1})$ is in H'; and, since $Z(u_k)$ belongs to the irreducible sink H, then this edge is also in H. Furthermore, the inclusion $\mathcal{F}_{\mathcal{E}}(v_k) \subseteq \mathcal{F}_G(u_k)$ implies the inclusion $\mathcal{F}_{\mathcal{E}}(v_{k+1}) \subseteq \mathcal{F}_G(u_{k+1})$; hence v_{k+1} is in $Z(u_{k+1})$. □

Lemma 5.3 *Let $\mathcal{E}_1$ be an n_1-regular irreducible subgraph of an irreducible graph G_2 where the minimum degree in G_2 is $n_2 \geq n_1$. Then there exists an n_2-regular irreducible subgraph $\mathcal{E}_2$ of G_2 such that $\mathcal{E}_1 \subseteq \mathcal{E}_2$.*

Proof. Let H be a subgraph of G_2 that satisfies the following conditions:

(H1) The minimum degree of H is at least n_2 and —

(H2) H contains $\mathcal{E}_1$ as a subgraph and the states of $\mathcal{E}_1$ are accessible from every state in H.

(In particular, conditions (H1) and (H2) hold for $H \leftarrow G_2$.) We show that if H contains a state with out-degree greater than n_2, then we can always delete an edge from H to obtain a new graph H' that satisfies (H1) and (H2).

Let u be a state in H whose out-degree is greater than n_2. For every $e \in E_H(u)$, define the *distance* of e (from $V_{\mathcal{E}_1}$) as the length of the shortest path in H from $\tau_H(e)$ to $V_{\mathcal{E}_1}$ (the distance is zero if $\tau_H(e)$ is in $V_{\mathcal{E}_1}$). Let $e_{\max}$ be a particular edge in $E_H(u)$ whose distance is the largest among all the distances of the edges in $E_H(u)$. Note that $e_{\max}$ is not in $\mathcal{E}_1$, even when u is in $\mathcal{E}_1$. The graph H' is obtained from H by deleting $e_{\max}$.

We next show that from every state $v \in V_{H'}$ there is a path in H' from v to $V_{\mathcal{E}_1}$. Let π denote a particular shortest path in H from v to $V_{\mathcal{E}_1}$. If π does not pass through $e_{\max}$, then π is also a path in H' and we are done. Otherwise, there is a prefix of π (possibly of length zero) that is a path from v to u which does not pass through $e_{\max}$; as such, this prefix is entirely contained in H'. Hence, it suffices to show that there is path in H' from u to $V_{\mathcal{E}_1}$. Let e' be an edge in $E_H(u) \setminus \{e_{\max}\}$. Since the distance of e' is not greater than the distance of the deleted edge $e_{\max}$, there must be a path π' in H from $\tau_H(e')$ to $V_{\mathcal{E}_1}$ that does not pass through $e_{\max}$. It follows that the path $e'\pi'$ is entirely contained in H'.

In order to obtain the graph $\mathcal{E}_2$, we proceed as follows. We start with $H \leftarrow G_2$, and then successively delete edges from H while satisfying conditions (H1) and (H2), until we reach a graph $\hat{\mathcal{E}}_2$ that is n_2-regular. Finally, we let $\mathcal{E}_2$ be an irreducible sink of $\hat{\mathcal{E}}_2$. Since $V_{\mathcal{E}_1}$ is accessible from every state in $\hat{\mathcal{E}}_2$, the graph $\mathcal{E}_1$ must be a subgraph of that sink. □

Proof of Theorem 5.1. We start with the 'only if' part. Suppose there exist irreducible deterministic (S_i, n_i)-encoders $\mathcal{E}_i$ such that $\mathcal{E}_1 \subseteq \mathcal{E}_2$. By Lemma 5.2, there is a (G_2, n_2)-component H_2 such that $S(\mathcal{E}_2) \subseteq S(H_2)$. Hence,

$$S(\mathcal{E}_1) \;\subseteq\; S_1 \cap S(\mathcal{E}_2) \;\subseteq\; S_1 \cap S(H_2) \;=\; S(G_1 * H_2)\;.$$

Again, by Lemma 5.2 there is a $(G_1 * H_2, n_1)$-component H_1 such that $S(\mathcal{E}_1) \subseteq S(H_1)$.

We next turn to the 'if' part. Let H_1 be a $(G_1 * H_2, n_1)$-component where H_2 is a (G_2, n_2)-component. Clearly, $S(H_1)$ and $S(H_2)$ are irreducible and $S(H_1) \subseteq S(H_2)$. Applying Lemma 4.2 to $S(H_1)$ and $S(H_2)$, there exist irreducible deterministic graphs H_1' and H_2' such that $S(H_1) = S(H_1')$, $S(H_2) = S(H_2')$, and H_1' is a subgraph of H_2'. Furthermore, by Theorem 2.3(b), every state in H_i' is follower-set equivalent to some state in H_i for $i = 1, 2$; so, the minimum degree of H_i' is at least n_i.

Next, we take $\mathcal{E}_1$ to be any n_1-regular irreducible subgraph of H_1' (e.g., $\mathcal{E}_1$ is an irreducible sink of some n_1-regular subgraph of H_1'); as such, $\mathcal{E}_1$ is also a subgraph of H_2'. Finally, we invoke Lemma 5.3 to obtain $\mathcal{E}_2$ as an n_2-regular irreducible subgraph of H_2' that contains $\mathcal{E}_1$ as a subgraph. □

Re-iterating our remark towards the end of Section 4, the nested encoders in Theorem 5.1 are not necessarily reduced. Therefore, for reduced encoders, the conditions of the theorem are only necessary.

6 The diamond condition set

Let S_1 and S_2 be irreducible constraints such that $S_1 \subseteq S_2$. We say that a quadruple of deterministic graphs $(G_1, G_2, \mathcal{E}_1, \mathcal{E}_2)$ satisfies the *diamond condition set with respect to* (S_1, S_2, n_1, n_2) if and only if the following four conditions hold:

(A) G_1 and G_2 are irreducible deterministic presentations of S_1 and S_2, respectively.

(B) G_1 is a subgraph of G_2.

(C) $\mathcal{E}_2$ is an irreducible (S_2, n_2)-encoder and is a subgraph of G_2.

(D) $\mathcal{E}_1$ is an irreducible (S_1, n_1)-encoder and is a subgraph of G_1 and $\mathcal{E}_2$ (both being subgraphs of G_2).

The diamond condition set is illustrated in Figure 8.

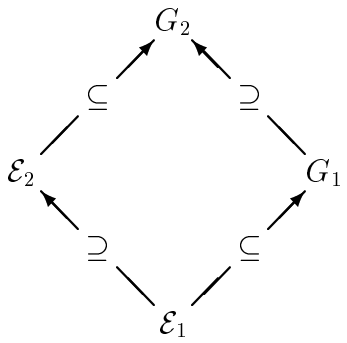

Figure 8: Diamond condition set.

The diamond condition set seems to be the appropriate generalization of the containment condition in (the irreducible version of) Proposition 2.7 to nested encoders. Namely, we

would like to claim that if there exists an irreducible deterministic (S_1, n_1)-encoder which is nested in an irreducible deterministic (S_2, n_2)-encoder, then there is a quadruple of deterministic graphs $(G_1, G_2, \mathcal{E}_1, \mathcal{E}_2)$ that satisfies the diamond condition set with respect to (S_1, S_2, n_1, n_2). Indeed, we will prove that this holds when S_2 has finite memory. Yet, Example 6.1 below shows that the claim is false when S_2 has infinite memory.

Example 6.1 Let S_2 be the constraint presented by the graph H_2 in Figure 9, and let S_1 be the constraint presented by the subgraph in Figure 9 that is marked by the dotted box; namely, S_1 is presented by the subgraph H_1 of H_2 that is induced by states γ and δ. We choose $n_1 = 2$ and $n_2 = 3$ and verify that the subgraph induced by states α

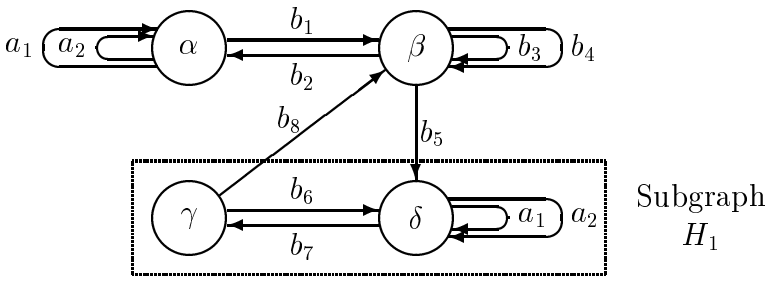

Figure 9: Graph H_2 for Example 6.1.

and β is an (S_2, n_2)-encoder, and the subgraph induced by state α (or state δ) is an (S_1, n_1)-encoder.

We next claim that there is no quadruple of deterministic graphs $(G_1, G_2, \mathcal{E}_1, \mathcal{E}_2)$ that satisfies the diamond condition set with respect to $(S_1, S_2, 2, 3)$. Indeed, suppose to the contrary that there are such graphs, and let u be a state that belongs to all those graphs. By Theorem 2.3(b), the states of G_i are follower-set equivalent to states in H_i for $i = 1, 2$. Since u has three outgoing edges in $\mathcal{E}_2$, then, as a state of G_2, state u cannot be follower-set equivalent to state γ in H_2. Now, γ is the only state in H_2 that has outgoing edges labeled b_6 or b_8. Therefore, no outgoing edge from state u in G_2 can be labeled by those symbols. It follows that state u in G_1 cannot be follower-set equivalent to state γ in H_1, which means that u, as a state of G_1, must be follower-set equivalent to state δ in H_1. In particular, u has in G_1 an outgoing edge labeled b_7. Such a symbol can be generated in H_2 only from state δ, thus implying that u, as a state of G_2, is follower-set equivalent to state δ in H_2. So, the three outgoing edges from state u in both G_2 and $\mathcal{E}_2$ are labeled a_1, a_2, and b_7. Let v denote the terminal state in $\mathcal{E}_2$ (and G_2) of the edge labeled b_7 outgoing from state u. State v in G_2 is follower-set equivalent to state γ in H_2, which means that state v in $\mathcal{E}_2$ has at most two outgoing edges, thus reaching a contradiction. □

6.1 Nested encoders in the finite-memory case

Theorem 6.1 *Let S_1 and S_2 be two irreducible constraints where $S_1 \subseteq S_2$ and S_2 has finite memory. Then there exists an irreducible deterministic (S_1, n_1)-encoder that is nested in an irreducible deterministic (S_2, n_2)-encoder if and only if there exists a quadruple of deterministic graphs $(G_1, G_2, \mathcal{E}_1, \mathcal{E}_2)$ that satisfies the diamond condition set with respect to (S_1, S_2, n_1, n_2).*

Theorem 6.1 is based on the following result.

Proposition 6.2 *Let S_A, S_B, and $S_{\max}$ be irreducible constraints where $|S_A \cap S_B| = \infty$, $S_A \cup S_B \subseteq S_{\max}$, and $S_{\max}$ has finite memory. Then there exists a finite set of irreducible deterministic graphs $\{G^{(1)}, G^{(2)}, \ldots, G^{(t)}\}$ that satisfies the following conditions:*

- *For every irreducible constraint $S_{\min} \subseteq S_A \cap S_B$, there is at least one deterministic graph $G^{(k)}$ such that $S_{\min} \subseteq S(G^{(k)})$.*
- *For every deterministic graph $G^{(k)}$ there exist irreducible deterministic graphs $G_A = G_A^{(k)}$, $G_B = G_B^{(k)}$, and $G_{\max} = G_{\max}^{(k)}$, such that*
  1. *G_A is a subgraph of $G_{\max}$ and $S_A = S(G_A)$;*
  2. *G_B is a subgraph of $G_{\max}$ and $S_B = S(G_B)$; and —*
  3. *$G^{(k)}$ is a subgraph of G_A and G_B within $G_{\max}$.*

The nesting relationships among the constraints in Proposition 6.2 are shown in Figure 10. We defer the proof of Proposition 6.2 to Section 6.3, and present here a proof of Theorem 6.1, based on Proposition 6.2.

Proof of Theorem 6.1. The 'if' part is obvious. We prove next the 'only if' part. For $i = 1, 2$, let $\hat{\mathcal{E}}_i$ be the given (S_i, n_i)-encoders. Apply Proposition 6.2 with $S_A \leftarrow S_1$, $S_B \leftarrow S(\hat{\mathcal{E}}_2)$, $S_{\max} \leftarrow S_2$, and $S_{\min} \leftarrow S(\hat{\mathcal{E}}_1)$, to obtain graphs $G^{(k)}$, G_A, G_B and $G_{\max}$. Note that the condition $|S_A \cap S_B| = \infty$ is satisfied since $S(\hat{\mathcal{E}}_1) \subseteq S_1 \cap S(\hat{\mathcal{E}}_2)$ and $|S(\hat{\mathcal{E}}_1)| = \infty$. We identify the graphs G_1, $\mathcal{E}_2$, and G_2 as G_A, G_B, and $G_{\max}$, respectively. Note that $G_B \equiv \mathcal{E}_2$ is n_2-regular since, by Theorem 2.3(b), the follower sets of the states in G_B and $\hat{\mathcal{E}}_2$ are the same.

Since $S(\hat{\mathcal{E}}_1) \subseteq S(G^{(k)})$, the graph $\hat{\mathcal{E}}_1$ is a deterministic $(S(G^{(k)}), n_1)$-encoder. Therefore, by Proposition 2.7, there is a subgraph of $G^{(k)}$ that is an $(S(G^{(k)}), n_1)$-encoder. Taking an irreducible sink of that subgraph, we obtain the desired irreducible deterministic (S_1, n_1)-encoder $\mathcal{E}_1$. □

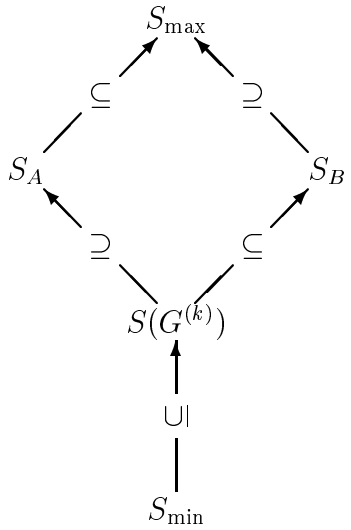

Figure 10: Constraints in Proposition 6.2.

6.2 Nested encoders within the Shannon covers

In this section, we discuss a special case of the diamond condition set where there is an irreducible presentation G_1 of S_1 that is a subgraph of the Shannon cover G_2 of S_2. As we show, in such a case, G_1 and G_2 can be taken as the graphs guaranteed by Theorem 6.1. Note that the important family of (d, k)-RLL constraints falls into the category studied here.

Let G be an irreducible deterministic graph of finite memory that presents a constraint S and let $\mathcal{E}$ be an irreducible deterministic (S, n)-encoder. We construct next a subgraph $H_{\mathcal{E}}$ of G which we call the *super-graph of $\mathcal{E}$ (with respect to G)*.

Let m be the memory of G and, for a state $u \in V_G$, let $W_G(u)$ be the set of all words of length m that can be generated by paths in G that terminate in state u. Note that since G has memory m, the sets $W_G(u)$ are disjoint for distinct states u.

For a word $\mathbf{w}$ of length m over $\Sigma(S)$, we define $V_{\mathcal{E}}(\mathbf{w})$ to be the set of terminal states of the paths in $\mathcal{E}$ that generate the word $\mathbf{w}$.

The graph $H_{\mathcal{E}}$ is defined as follows. For a state $u \in V_G$, let $Z_{G,\mathcal{E}}(u) = Z(u)$ denote the union $\cup_{\mathbf{w} \in W_G(u)} V_{\mathcal{E}}(\mathbf{w})$. The states of $H_{\mathcal{E}}$ are all the nonempty sets $Z(u)$, $u \in V_G$. We endow $H_{\mathcal{E}}$ with an edge $Z(u) \stackrel{a}{\longrightarrow} Z(u')$ if and only if there are words $\mathbf{w} \in W_G(u)$, $\mathbf{w}' \in W_G(u')$ such that $\mathbf{w}a = b\mathbf{w}'$, and states $v \in V_{\mathcal{E}}(\mathbf{w})$, $v' \in V_{\mathcal{E}}(\mathbf{w}')$ such that $v \stackrel{a}{\longrightarrow} v'$ is an edge in $\mathcal{E}$.

Lemma 6.3 *Let G be an irreducible deterministic graph of finite memory that presents a constraint S and let $H_{\mathcal{E}}$ be the super-graph of an irreducible deterministic (S,n)-encoder $\mathcal{E}$ with respect to G. Then*

(a) $H_{\mathcal{E}}$ is isomorphic to some subgraph of G.

(b) $H_{\mathcal{E}}$ is irreducible.

(c) $S(\mathcal{E}) \subseteq S(H_{\mathcal{E}})$.

(d) The minimum degree of $H_{\mathcal{E}}$ is at least n.

Proof. (a) We show that if $Z(u) \stackrel{a}{\longrightarrow} Z(u')$ is an edge in $H_{\mathcal{E}}$, then $u \stackrel{a}{\longrightarrow} u'$ is an edge in G. Indeed, if $Z(u) \stackrel{a}{\longrightarrow} Z(u')$ is in $H_{\mathcal{E}}$, then there are words $\mathbf{w} \in W_G(u)$, $\mathbf{w}' \in W_G(u')$ such that $\mathbf{w}a = b\mathbf{w}'$, and states $v \in V_{\mathcal{E}}(\mathbf{w})$, $v' \in V_{\mathcal{E}}(\mathbf{w}')$ such that $v \stackrel{a}{\longrightarrow} v'$ is an edge in $\mathcal{E}$. This implies that $\mathbf{w}a$ is a word in $S(\mathcal{E})$, and, as such, it is also a word in S. It follows by Lemma 2.5 that there is an edge $u \stackrel{a}{\longrightarrow} u'$ in G.

(b) Let $Z(u)$ and $Z(u')$ be two states in $H_{\mathcal{E}}$ and let $\mathbf{w}$ and $\mathbf{w}'$ be words in $W_G(u)$ and $W_G(u')$, respectively, such that $V_{\mathcal{E}}(\mathbf{w})$ and $V_{\mathcal{E}}(\mathbf{w}')$ are nonempty. Let v_0 be a state in $V_{\mathcal{E}}(\mathbf{w})$. Since $\mathcal{E}$ is irreducible, there is a path

$$v_0 \stackrel{a_1}{\longrightarrow} v_1 \stackrel{a_2}{\longrightarrow} \ldots \stackrel{a_\ell}{\longrightarrow} v_\ell \tag{9}$$

in $\mathcal{E}$ that generates a word $a_1 a_2 \ldots a_\ell$ whose suffix is $\mathbf{w}'$. Write $\mathbf{w} = a_{-m+1}a_{-m+2}\ldots a_0$, and denote by $\mathbf{w}_j$ the word $a_{j-m+1}a_{j-m+2}\ldots a_j$. In particular, $\mathbf{w}_0 = \mathbf{w}$ and $\mathbf{w}_\ell = \mathbf{w}'$. We have $v_j \in V_{\mathcal{E}}(\mathbf{w}_j)$ and $\mathbf{w}_{j-1}a_j = a_{j-m}\mathbf{w}_j$. Hence, there are edges $Z(u_{j-1}) \stackrel{a_j}{\longrightarrow} Z(u_j)$ in $H_{\mathcal{E}}$, where each u_j is the unique state in G such that $\mathbf{w}_j \in W_G(u_j)$. In particular, $u_0 = u$ and $u_\ell = u'$. We conclude that there is a path

$$Z(u) \equiv Z(u_0) \stackrel{a_1}{\longrightarrow} Z(u_1) \stackrel{a_2}{\longrightarrow} \ldots \stackrel{a_\ell}{\longrightarrow} Z(u_\ell) \equiv Z(u') \tag{10}$$

in $H_{\mathcal{E}}$.

(c) Let (9) be a path in $\mathcal{E}$ of length $\ell \geq m$, and let $u_0 \in V_G$ and $\mathbf{w} \in W_G(u_0)$ be such that $v_0 \in V_{\mathcal{E}}(\mathbf{w})$. The proof of (b) implies that there is also a path (10) in $H_{\mathcal{E}}$. Hence, $S(\mathcal{E}) \subseteq S(H_{\mathcal{E}})$.

(d) Let v be a state in $Z(u)$. If there is an outgoing edge labeled a from v in $\mathcal{E}$, then there also exists an outgoing edge labeled a from $Z(u)$ in $H_{\mathcal{E}}$. Hence, the minimum degree of $H_{\mathcal{E}}$ is at least n. □

Theorem 6.4 *Let S_1 and S_2 be two irreducible constraints such that $S_1 \subseteq S_2$ and S_2 has finite memory m. Suppose that there is an irreducible presentation G_1 of S_1 that*

is a subgraph of the Shannon cover G_2 of S_2. Further, suppose that there exist two irreducible deterministic encoders $\mathcal{E}_1' \subseteq \mathcal{E}_2'$ where $\mathcal{E}_1'$ is an (S_1, n_1)-encoder and $\mathcal{E}_2'$ is an (S_2, n_2)-encoder. Then there exist two irreducible deterministic encoders $\mathcal{E}_1$ and $\mathcal{E}_2$ such that the quadruple $(G_1, G_2, \mathcal{E}_1, \mathcal{E}_2)$ satisfies the diamond condition set with respect to (S_1, S_2, n_1, n_2).

Proof. For $i = 1, 2$, let $H_{\mathcal{E}_i'}$ be the super-graph of $\mathcal{E}_i'$ with respect to G_i. By Lemma 6.3(a), we can regard $H_{\mathcal{E}_i'}$ as a subgraph of G_i, with state $Z_{G_i,\mathcal{E}_i'}(u)$ in $H_{\mathcal{E}_i'}$ identified with state u in G_i. We show that $H_{\mathcal{E}_1'}$ is a subgraph of $H_{\mathcal{E}_2'}$. For every state $u \in V_{G_1}$ we have $W_{G_1}(u) \subseteq W_{G_2}(u)$, and for every $\mathbf{w} \in W_{G_1}(u)$ we have $V_{\mathcal{E}_1'}(\mathbf{w}) \subseteq V_{\mathcal{E}_2'}(\mathbf{w})$. Hence, $Z_{G_1,\mathcal{E}_1'}(u) \subseteq Z_{G_2,\mathcal{E}_2'}(u)$, and for every edge $Z_{G_1,\mathcal{E}_1'}(u) \xrightarrow{a} Z_{G_1,\mathcal{E}_1'}(u')$ in $H_{\mathcal{E}_1'}$ we also have an edge $Z_{G_2,\mathcal{E}_2'}(u) \xrightarrow{a} Z_{G_2,\mathcal{E}_2'}(u')$ in $H_{\mathcal{E}_2'}$.

Next, we define $\mathcal{E}_1$ to be an n_1-regular irreducible subgraph of $H_{\mathcal{E}_1'}$; clearly, $\mathcal{E}_1$ is an irreducible deterministic (S_1, n_1)-encoder which is a subgraph of $H_{\mathcal{E}_2'}$. By Lemmas 5.3 and 6.3(d), there is an irreducible n_2-regular subgraph $\mathcal{E}_2$ of $H_{\mathcal{E}_2'}$ such that $\mathcal{E}_1 \subseteq \mathcal{E}_2$. The graph $\mathcal{E}_2$ is the desired (S_2, n_2)-encoder. □

We point out that Theorem 6.4 is false if we remove the requirement that G_1 is a subgraph of G_2. For example, let Σ be an alphabet of size n_2, and let S_2 be the constraint consisting of all words over Σ. The Shannon cover G_2 of S_2 consists of a single state and n_2 self-loops, and G_2 is an (S_2, n_2)-encoder. Let S_1 be an irreducible constraint over Σ whose Shannon cover is an (S_1, n_1)-encoder with at least two states. Then no (S_1, n_1)-encoder is contained in G_2.

6.3 Proof of Proposition 6.2

The proof of Proposition 6.2 is technical but rather long and will be carried out by constructing the deterministic graphs $G^{(k)}$, $G_A^{(k)}$, $G_B^{(k)}$, and $G_{\max}^{(k)}$, and proving several lemmas. We denote by H_A, H_B, and $H_{\max}$ the Shannon covers of S_A and S_B, and $S_{\max}$, respectively, and by m the memory of $H_{\max}$.

Graph $G^{(k)}$

Let $\{H^{(1)}, H^{(2)}, \ldots, H^{(t)}\}$ be a maximal set of nontrivial irreducible components of $H_A * H_B$ such that $S(H^{(i)}) \not\subseteq S(H^{(j)})$ for $i \neq j$. Since $|S_A \cap S_B| = \infty$, such a set of components is nonempty. By Lemma 2.2, for every irreducible constraint $S_{\min} \subseteq S_A \cap S_B$ there is an irreducible component $H^{(k)}$ such that $S_{\min} \subseteq S(H^{(k)})$. For every component $H^{(k)}$, we define the graph $G^{(k)} = (V_{G^{(k)}}, E_{G^{(k)}}, L_{G^{(k)}})$ as follows. The set $V_{G^{(k)}}$ consists of all triples $(\mathbf{w}; u, v)$, where $\langle u, v\rangle \in V_{H^{(k)}}$ and $\mathbf{w}$ is a word of length m that is generated by a path in $H^{(k)}$ that terminates in state $\langle u, v\rangle$. We endow $G^{(k)}$ with an edge $(\mathbf{w}; u, v) \xrightarrow{a}$

$(\mathbf{w}'; u', v')$ if and only if $\langle u, v\rangle \xrightarrow{a} \langle u', v'\rangle$ is an edge in $H^{(k)}$ and $\mathbf{w}a = b\mathbf{w}'$ for some $b \in \Sigma(S_A) \cap \Sigma(S_B)$.

The following can be easily verified.

Lemma 6.5 *The graph $G^{(k)}$ is an irreducible deterministic graph that presents $S(H^{(k)})$.*

It follows from Lemma 2.2 and Lemma 6.5 that for every irreducible $S_{\min} \subseteq S_A \cap S_B$ there is a graph $G^{(k)}$ such that $S_{\min} \subseteq S(G^{(k)})$. Hereafter, we fix a particular graph $G^{(k)}$ and define the respective graphs $G_A = G_A^{(k)}$, $G_B = G_B^{(k)}$, and $G_{\max} = G_{\max}^{(k)}$ (the construction of which depends on k).

Graph $G'_{\max}$

We first introduce a graph $G'_{\max} = (V_{G'_{\max}}, E_{G'_{\max}}, L_{G'_{\max}})$, where $V_{G'_{\max}}$ consists of all triples $(\mathbf{w}; u, v)$, where $u \in V_{H_A} \cup \{\phi_A\}$, $v \in V_{H_B} \cup \{\phi_B\}$, and $\mathbf{w}$ is a word of length m in $S_{\max}$, with the additional requirement that —

1. if $u \in V_{H_A}$, then $\mathbf{w}$ is generated by a path in H_A that terminates in state u, and —
2. if $v \in V_{H_B}$, then $\mathbf{w}$ is generated by a path in H_B that terminates in state v.

The set of edges $E_{G'_{\max}}$ and their labels are defined in Table 1. The principles of constructing the edges in $G'_{\max}$ are as follows. First, an edge $(\mathbf{w}; u, v) \xrightarrow{a} (\mathbf{w}'; u', v')$ exists in $G'_{\max}$ only if $\mathbf{w}'$ is a suffix of $\mathbf{w}a$ and $\mathbf{w}a \in S_{\max}$. Now, if $u \in V_{H_A}$, then u' is the terminal state of the edge labeled a outgoing from state u in H_A, provided that such an edge exists. If no such edge exists, then $u' = \phi_A$. A similar rule applies also to the case where $v \in V_{H_B}$. If $\langle u, v\rangle = \langle \phi_A, \phi_B\rangle$, then $\langle u', v'\rangle = \langle \phi_A, \phi_B\rangle$, unless there is a state of the form $(\mathbf{w}'; u'', v'')$ in $G^{(k)}$ for the given word $\mathbf{w}'$, in which case $\langle u', v'\rangle$ will be the first pair $\langle u'', v''\rangle$ (according to some lexicographic ordering) for which such a state exists in $G^{(k)}$. Finally, if $u \in V_{H_A}$ and $v = \phi_B$, we first determine u' according to the previous rules, and then set $v' = \phi_B$, unless there is a state of the form $(\mathbf{w}'; u', v'')$ in $G^{(k)}$, in which case v' is set so that $\langle u', v'\rangle$ is the first pair $\langle u', v''\rangle$ for which such a state exists in $G^{(k)}$. A similar rule applies in case $u = \phi_A$ and $v \in V_{H_B}$.

Clearly, $G'_{\max}$ is deterministic. States $(\mathbf{w}; u, v)$ in $G'_{\max}$ in which either $u = \phi_A$ or $v = \phi_B$ will be called ϕ-states of $G'_{\max}$. Note that there may be states in $V_{G'_{\max}}$ which are neither states of $V_{G^{(k)}}$ nor ϕ-states. Nevertheless, an outgoing edge from a ϕ-state must terminate either in $V_{G^{(k)}}$ or in a ϕ-state. Furthermore, in the latter case, the ϕ-coordinate cannot change its position, i.e., there are no edges connecting ϕ-states of the type $(\mathbf{w}; u, \phi_B)$ with ϕ-states of the type $(\mathbf{w}'; \phi_A, v')$ unless $v' = \phi_B$.

$u \in$	$v \in$	$u' \in$	$v' \in$	$(\mathbf{w}; u, v) \xrightarrow{a} (\mathbf{w}'; u', v') \in E_{G'_{\max}}$ if $\mathbf{w}a = b\mathbf{w}' \in S_{\max}$ and
V_{H_A}	V_{H_B}	V_{H_A}	V_{H_B}	$u \xrightarrow{a} u' \in E_{H_A}$, $v \xrightarrow{a} v' \in E_{H_B}$
V_{H_A}	V_{H_B}	V_{H_A}	$\{\phi_B\}$	$u \xrightarrow{a} u' \in E_{H_A}$, $\forall v'' \in V_{H_B} : v \xrightarrow{a} v'' \notin E_{H_B}$
V_{H_A}	V_{H_B}	$\{\phi_A\}$	V_{H_B}	$v \xrightarrow{a} v' \in E_{H_B}$, $\forall u'' \in V_{H_A} : u \xrightarrow{a} u'' \notin E_{H_A}$
V_{H_A}	V_{H_B}	$\{\phi_A\}$	$\{\phi_B\}$	$\forall \langle u'', v'' \rangle \in V_{H_A * H_B} : u \xrightarrow{a} u'' \notin E_{H_A}$, $v \xrightarrow{a} v'' \notin E_{H_B}$
$\{\phi_A\}$	$\{\phi_B\}$	$\{\phi_A\}$	$\{\phi_B\}$	$\forall \langle u'', v'' \rangle \in V_{H_A * H_B} : (\mathbf{w}'; u'', v'') \notin V_{G^{(k)}}$
$\{\phi_A\}$	$\{\phi_B\}$	V_{H_A}	V_{H_B}	$\langle u', v' \rangle = \text{first } \langle u'', v'' \rangle \in V_{H_A * H_B} : (\mathbf{w}'; u'', v'') \in V_{G^{(k)}}$
V_{H_A}	$\{\phi_B\}$	V_{H_A}	$\{\phi_B\}$	$u \xrightarrow{a} u' \in E_{H_A}$, $\forall v'' \in V_{H_B} : (\mathbf{w}'; u', v'') \notin V_{G^{(k)}}$
V_{H_A}	$\{\phi_B\}$	V_{H_A}	V_{H_B}	$u \xrightarrow{a} u' \in E_{H_A}$, $v' = \text{first } v'' \in V_{H_B} : (\mathbf{w}'; u', v'') \in V_{G^{(k)}}$
V_{H_A}	$\{\phi_B\}$	$\{\phi_A\}$	$\{\phi_B\}$	$\forall u'' \in V_{H_A} : u \xrightarrow{a} u'' \notin E_{H_A}$
$\{\phi_A\}$	V_{H_B}	$\{\phi_A\}$	V_{H_B}	$v \xrightarrow{a} v' \in E_{H_B}$, $\forall u'' \in V_{H_A} : (\mathbf{w}'; u'', v') \notin V_{G^{(k)}}$
$\{\phi_A\}$	V_{H_B}	V_{H_A}	V_{H_B}	$v \xrightarrow{a} v' \in E_{H_B}$, $u' = \text{first } u'' \in V_{H_A} : (\mathbf{w}'; u'', v') \in V_{G^{(k)}}$
$\{\phi_A\}$	V_{H_B}	$\{\phi_A\}$	$\{\phi_B\}$	$\forall v'' \in V_{H_B} : v \xrightarrow{a} v'' \notin E_{H_B}$

Table 1: Edges of $G'_{\max}$.

It is easy to see that $G^{(k)}$ is a subgraph of $G'_{\max}$. Indeed, let $(\mathbf{w}; u, v) \xrightarrow{a} (\mathbf{w}'; u', v')$ be an edge in $G^{(k)}$. Then $\mathbf{w}a$ must be a word in $S_A \cap S_B$ and, as such, it is also a word in $S_{\max}$. Hence $(\mathbf{w}; u, v) \xrightarrow{a} (\mathbf{w}'; u', v')$ is an edge in $G'_{\max}$.

Graph G_A

Let G'_A be the subgraph of $G'_{\max}$ induced by all states $(\mathbf{w}; u, v) \in V_{G'_{\max}}$ with $u \in V_{H_A}$. Our previous argument for showing that $G^{(k)}$ is a subgraph of $G'_{\max}$ implies in fact that $G^{(k)}$ is a subgraph of G'_A within $G'_{\max}$. So, there is a unique irreducible component in G'_A that contains $G^{(k)}$ as a subgraph. We let G_A be that unique irreducible component, and we have the following.

Lemma 6.6 $S(G_A) = S_A$.

Proof. By construction of $G'_{\max}$, there is a path

$$\pi : (\mathbf{w}_0; u_0, v_0) \xrightarrow{a_1} (\mathbf{w}_1; u_1, v_1) \xrightarrow{a_2} \ldots \xrightarrow{a_\ell} (\mathbf{w}_\ell; u_\ell, v_\ell) \tag{11}$$

in $G'_{\max}$ with $u_j \in V_{H_A}$ only if

$$u_0 \xrightarrow{a_1} u_1 \xrightarrow{a_2} \ldots \xrightarrow{a_\ell} u_\ell \tag{12}$$

is a path in H_A. Conversely, for every path (12) in H_A there are words $\mathbf{w}_j$ and states $v_j \in V_{H_B} \cup \{\phi_B\}$ such that (11) is a path in $G'_{\max}$. Hence, $S(G'_A) = S_A$ and, so, $S(G_A) \subseteq S_A$. The rest of the proof is devoted to showing that $S_A \subseteq S(G_A)$.

Let u_0 be a state in V_{H_A} such that $(\mathbf{w}_0; u_0, v_0) \in V_{G^{(k)}}$ for some $\mathbf{w}_0$ and v_0. Further, assume that v_0 is the first state v' in V_{H_B} such that $(\mathbf{w}_0; u_0, v') \in V_{G^{(k)}}$. We obtain the containment $S_A \subseteq S(G_A)$ by showing that every word in S_A that can be generated in H_A by a path starting at state u_0, can also be generated in G_A. Let $a_1 a_2 \ldots, a_\ell$ be such a word in S_A and let (12) be a path that generates that word in H_A. There is also a path π as in (11) that generates that word in G'_A. In order to show that π is in G_A, it suffices to show that there is a path in G'_A from $(\mathbf{w}_\ell; u_\ell, v_\ell)$ to $V_{G^{(k)}}$. We distinguish between the following three cases.

Case 1: $v_\ell = \phi_B$. Let

$$u'_0 \xrightarrow{b_1} u'_1 \xrightarrow{b_2} \ldots \xrightarrow{b_r} u'_r \tag{13}$$

be a path in H_A from $u'_0 = u_\ell$ to $u'_r = u_0$ that generates a word $b_1 b_2 \ldots b_r$ whose suffix equals $\mathbf{w}_0$. Then there is also a path

$$(\mathbf{w}_\ell; u_\ell, v_\ell) \equiv (\mathbf{w}'_0; u'_0, v'_0) \xrightarrow{b_1} (\mathbf{w}'_1; u'_1, v'_1) \xrightarrow{b_2} \ldots \xrightarrow{b_r} (\mathbf{w}'_r; u'_r, v'_r) \equiv (\mathbf{w}_0; u_0, v'_r) \tag{14}$$

in G'_A that generates the same word $b_1 b_2 \ldots b_r$. Note that if $v'_{r-1} = \phi_B$ then $v'_r = v_0$; hence, there must be an index $j \leq r$ such that $v'_{j-1} = \phi_B$ and $v'_j \in V_{H_B}$. Furthermore, by construction of $G'_{\max}$, the state $(\mathbf{w}'_j; u'_j, v'_j)$ is in $V_{G^{(k)}}$. It follows that there is a path in G'_A from $(\mathbf{w}_\ell; u_\ell, \phi_B)$ to $V_{G^{(k)}}$.

Case 2: $v_\ell \neq \phi_B$ *and* $\mathcal{F}_{H_A}(u_\ell) \not\subseteq \mathcal{F}_{H_B}(v_\ell)$. Let $\mathbf{z}$ be a word in $\mathcal{F}_{H_A}(u_\ell)$ that is not contained in $\mathcal{F}_{H_B}(v_\ell)$. Consider the paths (13) and (14) as in Case 1, and assume further that $\mathbf{z}$ is a prefix of $b_1 b_2 \ldots b_r$. Since $\mathbf{z} \notin \mathcal{F}_{H_B}(v_\ell)$, there must be at least one index $j \leq r$ such that $v'_j = \phi_B$. Now, from Case 1 it follows that there is a path in $V_{G'_A}$ from $(\mathbf{w}'_j; u'_j, \phi_B)$ to $V_{G^{(k)}}$. Hence, there is a path from $(\mathbf{w}_\ell; u_\ell, v_\ell)$ to $V_{G^{(k)}}$.

Case 3: $v_\ell \neq \phi_B$ *and* $\mathcal{F}_{H_A}(u_\ell) \subseteq \mathcal{F}_{H_B}(v_\ell)$. By Lemma 2.4(b) we have $S_A \subseteq S_B$ which implies $S(H_A * H_B) = S_A$; so, by Lemma 2.2, there is an irreducible component $H^{(i)}$ such that $S(H^{(i)}) = S_A$. However, we also assume that $S(H^{(k)}) \not\subseteq S(H^{(i)})$ unless $i = k$. Hence, $S(H^{(k)}) = S_A$, and by Lemma 6.5 we thus have $S(G^{(k)}) = S_A$. This, in turn, yields the degenerate case where the inclusions in the chain

$$S_A \;=\; S(H^{(k)}) \;=\; S(G^{(k)}) \;\subseteq\; S(G_A) \;\subseteq\; S_A$$

become equalities. □

Graph G_B

In analogy to G_A, the graph G'_B is defined as the subgraph of $G'_{\max}$ induced by all states $(\mathbf{w}; u, v)$ with $v \in V_{H_B}$. The graph G_B is defined as the irreducible component of G'_B that contains $G^{(k)}$ as a subgraph. The following lemma is proved similarly to Lemma 6.6.

Lemma 6.7 $S(G_B) = S_B$.

Graph $G_{\max}$

The graph $G_{\max}$ is defined as the unique irreducible component of $G'_{\max}$ that contains $G^{(k)}$ as a subgraph. Since $G^{(k)}$ is a subgraph of the irreducible subgraphs G_A and G_B within $G'_{\max}$, both G_A and G_B will be subgraphs of $G_{\max}$.

We next prove the following.

Lemma 6.8 $S(G_{\max}) = S_{\max}$.

Proof. The proof is similar to that of Lemma 6.6. By Lemma 2.5, the inclusion $S(G_{\max}) \subseteq S_{\max}$ follows from having any edge $(\mathbf{w}; u, v) \xrightarrow{a} (\mathbf{w}'; u', v')$ in $G'_{\max}$ only when $\mathbf{w}a \in S_{\max}$. In the remaining part of the proof we show the inclusion $S_{\max} \subseteq S(G_{\max})$.

Following the lines of the proof of Lemma 6.6, we let $\mathbf{w}_0$ be a word such that $(\mathbf{w}_0; u_0, v_0) \in V_{G^{(k)}}$ for some u_0 and v_0, and assume that $\langle u_0, v_0 \rangle$ is the first state $\langle u', v' \rangle \in V_{H_A * H_B}$ to satisfy $(\mathbf{w}_0; u', v') \in V_{G^{(k)}}$. The inclusion $S_{\max} \subseteq S(G_{\max})$ will follow by showing that every word generated in $H_{\max}$ by a path starting at state $\mathbf{w}_0$, can also be generated in $G_{\max}$. Let $a_1 a_2 \ldots, a_\ell$ be a word that is generated by the path

$$\mathbf{w}_0 \xrightarrow{a_1} \mathbf{w}_1 \xrightarrow{a_2} \ldots \xrightarrow{a_\ell} \mathbf{w}_\ell$$

in $H_{\max}$, where we identify a state in $H_{\max}$ with any word of length m that is generated by a path that terminates in that state. There is also a path

$$\pi : (\mathbf{w}_0; u_0, v_0) \xrightarrow{a_1} (\mathbf{w}_1; u_1, v_1) \xrightarrow{a_2} \ldots \xrightarrow{a_\ell} (\mathbf{w}_\ell; u_\ell, v_\ell)$$

that generates that word in $G'_{\max}$. To show that π is in $G_{\max}$, we first find a path in $G'_{\max}$ from $(\mathbf{w}_\ell; u_\ell, v_\ell)$ to a ϕ-state in $G'_{\max}$. Indeed, by construction of $G'_{\max}$, ϕ-states are inaccessible from $(\mathbf{w}_\ell; u_\ell, v_\ell)$ only when $\mathcal{F}_{H_{\max}}(\mathbf{w}_\ell) \subseteq \mathcal{F}_{H_A}(u_\ell) \cap \mathcal{F}_{H_B}(v_\ell)$, i.e., $S_{\max} \subseteq S_A \cap S_B$. This corresponds to the degenerate case where $H_A = H_B = H_{\max}$ (thus $S_{\max} = S_A = S(G_A) \subseteq S(G_{\max})$).

Next we show that from every ϕ-state $(\mathbf{w}; u, v) \in V_{G'_{\max}}$ there is a path in $G'_{\max}$ that terminates in $V_{G^{(k)}}$. We already showed in the proof of Lemma 6.6 (Case 1) that when $u \neq \phi_A$ there is a path from $(\mathbf{w}, u, \phi_B)$ (which is a state of G'_A) to $V_{G^{(k)}}$, and a similar proof applies to the case where $v \neq \phi_B$. Hence, it remains to consider the case where $\langle u, v \rangle = \langle \phi_A, \phi_B \rangle$. Since $S_{\max}$ is irreducible, there is a path in $H_{\max}$ that generates the word $\mathbf{wzw}_0$ for some word $\mathbf{z}$. Let this path be

$$\mathbf{w}'_0 \xrightarrow{b_1} \mathbf{w}'_1 \xrightarrow{b_2} \ldots \xrightarrow{b_r} \mathbf{w}'_r$$

with $\mathbf{w}'_0 = \mathbf{w}$ and $\mathbf{w}'_r = \mathbf{w}_0$. The word $\mathbf{wzw}_0$ is also generated by a path

$$(\mathbf{w}; \phi_A, \phi_B) \equiv (\mathbf{w}'_0; u'_0, v'_0) \xrightarrow{b_1} (\mathbf{w}'_1; u'_1, v'_1) \xrightarrow{b_2} \ldots \xrightarrow{b_r} (\mathbf{w}'_r; u'_r, v'_r) \equiv (\mathbf{w}_0; u'_r, v'_r)$$

in $G'_{\max}$. Note that if $\langle u'_{r-1}, v'_{r-1} \rangle = \langle \phi_A, \phi_B \rangle$, then $u'_r = u_0$ and $v'_r = v_0$. Hence, there must be an index $j \leq r$ such that $\langle u'_{j-1}, v'_{j-1} \rangle = \langle \phi_A, \phi_B \rangle$ and $\langle u'_j, v'_j \rangle \in V_{H_A * H_B}$; by construction, $(\mathbf{w}'_j, u'_j, v'_j) \in V_{G^{(k)}}$. □

Proof of Proposition 6.2. The proof follows from Lemmas 6.5, 6.6, 6.7, and 6.8. □

We point out that Proposition 6.2 does not necessarily hold if we remove the requirement that $|S_A \cap S_B| = \infty$. Indeed, suppose that $H_{\max}$ is given by the graph in Figure 11, H_A is the subgraph of $H_{\max}$ induced by states α and β, and H_B is the subgraph of $H_{\max}$ induced by states γ and δ. We have $S_A \cap S_B = \{b\}$; so, the only possible choice for

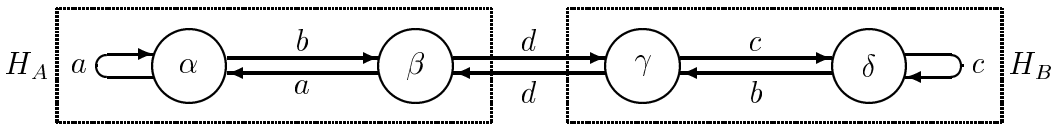

Figure 11: Example showing that we must have $|S_A \cap S_B| = \infty$.

$S_{\min}$ is the empty word, and $G^{(k)}$ must be the trivial graph with one state and no edges. However, if $G^{(k)}$ is a subgraph of both G_A and G_B within $G_{\max}$, then this means that G_A and G_B intersect in $G_{\max}$. In particular, we can generate words $\mathbf{w}_A \in S_A$ and $\mathbf{w}_B \in S_B$, each of length 2, such that $\mathbf{w}_A \mathbf{w}_B \in S_{\max}$. The word $\mathbf{w}_A$ contains at least one a and no d's, and the word $\mathbf{w}_B$ contains at least one c and no d's. However, no word in $S_{\max}$ can contain both a's and c's without having any d's.

Appendix

We show here an example of an irreducible deterministic (S_1, n_1)-encoder $\mathcal{E}_1$ that is observable from an irreducible (S_2, n_2)-encoder $\mathcal{E}_2$ with anticipation 1. On the other

hand, we show that if $\mathcal{E}_1'$ is an (S_1, n_1)-encoder nested in an (S_2, n_2)-encoder $\mathcal{E}_2'$, then $\mathcal{E}_2'$ must have anticipation at least 2. We also provide a pair of nested encoders $\mathcal{E}_1' \subseteq \mathcal{E}_2'$ where $\mathcal{E}_2'$ has infinite anticipation. (We do not know if there is a nested pair where $\mathcal{E}_2'$ has finite anticipation.)

Let S_1 be the constraint presented by the graph G_1 in Figure 12. The out-degree of each

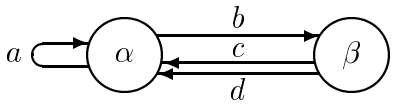

Figure 12: Shannon cover G_1 of S_1.

state in G_1 is 2; so, $\mathsf{cap}(S_1) = \log 2$ and G_1 is also a deterministic $(S_1, 2)$-encoder. We assign input tags over $\Upsilon_2 = \{0, 1\}$ to the edges of G_1 to obtain the encoder $\mathcal{E}_1$ that is shown in Figure 13.

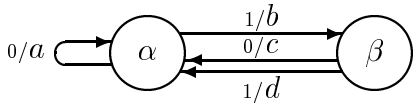

Figure 13: Tagged $(S_1, 2)$-encoder $\mathcal{E}_1$.

Let S_2 be the constraint presented by the graph G_2 in Figure 14. It is easy to check

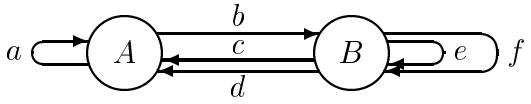

Figure 14: Shannon cover G_2 of S_2.

that $\mathsf{cap}(S_2) = \log \lambda(A_{G_2}) = \log 3$. By Proposition 2.7 there are no deterministic $(S_2, 3)$-encoders; however, we do have $(S_2, 3)$-encoders with anticipation 1. Such a tagged encoder, $\mathcal{E}_2$, with an assignment of input tags over $\Upsilon_3 = \{0, 1, 2\}$, is shown in Figure 15.

The encoder $\mathcal{E}_2$ can be decoded with one-symbol look-ahead using the state-independent decoder $\mathcal{D}_2$ whose decoding rules are given in Figure 16.

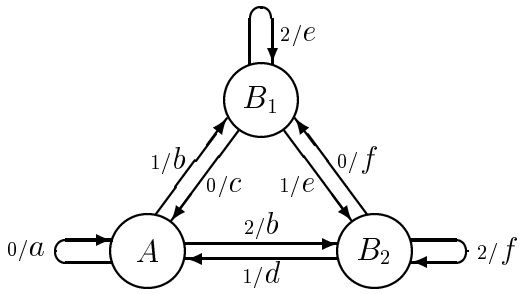

Figure 15: Tagged $(S_2, 3)$-encoder $\mathcal{E}_2$.

Current symbol	Next symbol	Decode into
a	$-$	0
b	c, e	1
b	d, f	2
c	$-$	0
d	$-$	1
e	c, e	2
e	d, f	1
f	c, e	0
f	d, f	2

Figure 16: Decoder $\mathcal{D}_2$ for $\mathcal{E}_2$.

Now, it is easy to see that $\mathcal{E}_1$ is observable from $\mathcal{E}_2$ with the function $\psi : \Upsilon_3 \to \Upsilon_2$ defined by $\psi(0) = 0$ and $\psi(1) = \psi(2) = 1$.

We next show that if $\mathcal{E}_1'$ is an $(S_1, 2)$-encoder nested in an $(S_2, 3)$-encoder $\mathcal{E}_2'$, then $\mathcal{E}_2'$ must have anticipation at least 2.

Suppose to the contrary that $\mathcal{E}_2'$ has anticipation 1. We first claim that no state in $\mathcal{E}_2'$ has three outgoing edges labeled a. Suppose that there were such a state, and let u_1, u_2, and u_3 be the terminal states of its outgoing edges. Since $\mathcal{E}_2'$ has anticipation 1, the sets $L_{\mathcal{E}_2'}(E_{\mathcal{E}_2'}(u_i))$ for distinct u_i's are nonempty and disjoint. However, from Figure 14 we see that the symbol a in a word of S_2 can be followed either by a or by b. Therefore, each set $L_{\mathcal{E}_2'}(E_{\mathcal{E}_2'}(u_i))$ must be contained in $\{a, b\}$; hence a contradiction.

Let r denote a symbol in the set $\{a, c, d\}$. We next verify that no state in $\mathcal{E}_2'$ has two outgoing edges labeled by the same symbol r. Suppose that there were such a state, and let u_1 and u_2 be the terminal states of its outgoing edges labeled r. Again, the

sets $L_{\mathcal{E}_2'}(E_{\mathcal{E}_2'}(u_1))$ and $L_{\mathcal{E}_2'}(E_{\mathcal{E}_2'}(u_2))$ are nonempty, disjoint, and contained in $\{a, b\}$. This means that the outgoing edges from u_1 (say) must all be labeled a. However, we have already shown that this is not possible.

Let t denote a symbol in the set $\{b, e, f\}$. As our next step, we show that no state in $\mathcal{E}_2'$ has three outgoing edges labeled by the same symbol t. If there were such a state, then the terminal states u_1, u_2, and u_3 of its outgoing edges could generate only symbols from the set $\{c, d, e, f\}$. However, from what we have shown it follows that there can be at most one edge labeled c outgoing from at most one of the u_i's, and the same applies to the label d. Hence, there are at least seven edges in $E_{\mathcal{E}_2'}(u_1) \cup E_{\mathcal{E}_2'}(u_2) \cup E_{\mathcal{E}_2'}(u_3)$ that are labeled e or f. Therefore, at least four of those edges must carry the same label (say, e), which implies that the symbol e can be generated from at least two of the u_i's, contradicting our assumption that $\mathcal{A}(\mathcal{E}_2') = 1$.

Let v be a state in $\mathcal{E}_1'$ (and $\mathcal{E}_2'$) from which we can generate the label b (clearly, there is always such a state, or else the words in $S(\mathcal{E}_1')$ could be generated by G_1 in Figure 12 without passing through the edge labeled b; this, however, would imply that $\mathsf{cap}(S(\mathcal{E}_1')) = 0$). The incoming edges to v in $\mathcal{E}_2'$ can be labeled a, c, or d; so the outgoing edges from v in $\mathcal{E}_2'$ can be labeled a or b. Furthermore, from what we have previously shown, it follows that there must be exactly one edge labeled a and two edges labeled b outgoing from v in $\mathcal{E}_2'$. Let u_1 and u_2 be the terminal states in $\mathcal{E}_2'$ of the edges labeled b outgoing from state v. At least one of those states, say u_1, is also a state in $\mathcal{E}_1'$. Since u_1 has an incoming edge labeled b, the two outgoing edges from u_1 in $\mathcal{E}_1'$ can be labeled either c or d. Furthermore, the labels of those two edges must be distinct, or else there would be two outgoing edges from u_1 in $\mathcal{E}_2'$ that would have the same label from the set $\{c, d\}$, contradicting our previous conclusions. Hence, one outgoing edge from u_1 in $\mathcal{E}_2'$ is labeled c, a second outgoing edge is labeled d, and the remaining third edge is labeled either e or f (no other symbol can follow the symbol b which labels an incoming edge to u_1); assume without loss of generality that the third edge is labeled e. This means that all the three outgoing edges from state u_2 in $\mathcal{E}_2'$ must be labeled f. However, no state in $\mathcal{E}_2'$ can have that. This establishes the contradiction that $\mathcal{E}_1'$ is nested in $\mathcal{E}_2'$ while $\mathcal{E}_2'$ has anticipation 1.

On the other hand, $\mathcal{E}_1$ can be nested in an $(S_2, 3)$-encoder with *infinite* anticipation. Such an encoder (which is still a lossless graph) is shown in Figure 17.

7 References

[1] R.L. Adler, D. Coppersmith, M. Hassner, *Algorithms for sliding block codes — an application of symbolic dynamics to information theory, IEEE Trans. Inform. Theory,* 29 (1983), 5–22.

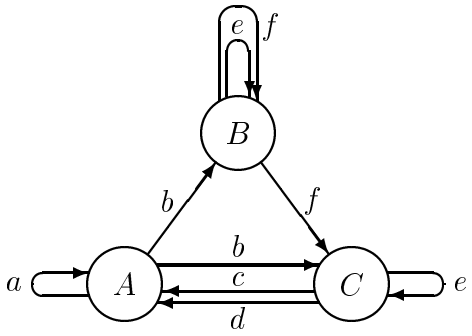

Figure 17: $(S_2, 3)$-encoder with infinite anticipation.

[2] J. Hogan, R.M. Roth, G. Ruckenstein, *Method and apparatus having cascaded decoding for multiple runlength-limited channel codes,* patent application filed with the US Patent Office.

[3] K.A.S. Immink, *Block-decodable runlength-limited codes via look-ahead technique, Philips J. Research,* 46 (1992), 293-310.

[4] K.A.S. Immink, *EFMPlus: The coding format of the multimedia compact disc, IEEE Trans. Consum. Electron.,* 41 (1995), 491–497.

[5] D. Lind and B. Marcus, *An Introduction to Symbolic Dynamics and Coding,* Cambridge University Press, 1995.

[6] B.H. Marcus, R.M. Roth, P.H. Siegel, *Constrained Systems and Coding for Recording Channels, Handbook of Coding Theory,* V. Pless, W.C. Huffman, R.A. Brualdi (Eds.), Elsevier Science Publishers, to appear. See also Technical Report No. 0929, Computer Science Department, Technion, Haifa, Israel (1988).

[7] B.H. Marcus, R.M. Roth, *Bounds on the number of states in encoders graphs for input-constrained channels, IEEE Trans. Inform. Theory,* IT-37 (1991), 742–758.

[8] B.H. Marcus, P.H. Siegel, J.K. Wolf, *Finite-state modulation codes for data storage, IEEE J. Sel. Areas Comm.,* 10 (1992), 5–37.

[9] G. Ruckenstein, *Encoding for Input-Constrained Channels,* M.Sc. Thesis, Computer Science Department, Technion, 1996.

[10] G. Ruckenstein, R.M. Roth, *Lower bounds on the anticipation of encoders for input-constrained channels,* in preparation. See also *IEEE Int'l Symp. on Information Theory,* Ulm, Germany (June 1997).

[11] E. Seneta, *Non-negative Matrices and Markov Chains,* Second Edition, Springer, New York, 1980.