Nested Input-Constrained Codes

Josh Hogan* Ron M. Roth† Gitit Ruckenstein‡

February 10, 2000

Abstract

An input-constrained channel, or simply a constraint, is a set S of words that is generated by a finite labeled directed graph. An encoder for S maps in a lossless manner sequences of unconstrained input blocks into sequences of channel blocks, the latter sequences being words of S. In most applications, the encoders are finite-state machines and, thus, presented by state diagrams. In the special case where the state diagram of the encoder is (output) deterministic, only the current encoder state and the current channel block are needed for the decoding of the current input block.

In this work, the problem of designing coding schemes 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.

Keywords: Constrained systems; Deterministic encoders; Finite-state encoders; Input-constrained channels; Nested encoders.

*Hewlett-Packard Laboratories, 1501 Page Mill Road, Palo Alto, CA 94304, USA.

†Computer Science Department, Technion, Haifa 32000, Israel. Work done in part while on sabbatical leave from Technion at Hewlett-Packard Laboratories, Palo Alto, CA 94304, USA, and in part while at Hewlett-Packard Laboratories—Israel, Haifa, Israel.

‡Computer Science Department, Technion, Haifa 32000, Israel. Work done as a summer student of Hewlett-Packard Laboratories—Israel and Hewlett-Packard Laboratories, Palo Alto, CA 94304, USA.

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.

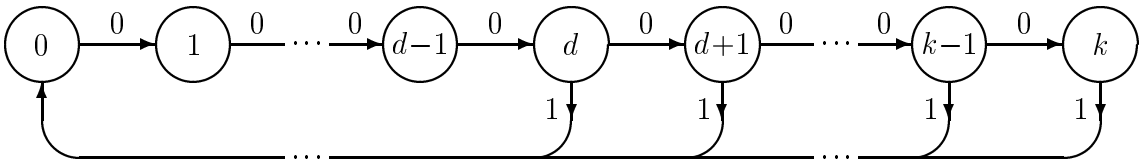

Figure 1: Graph presentation of $S_{(d,k)}$.

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. For example, 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 [4], [5].)

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 non-overlapping 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 $\mathcal{E}_2$.

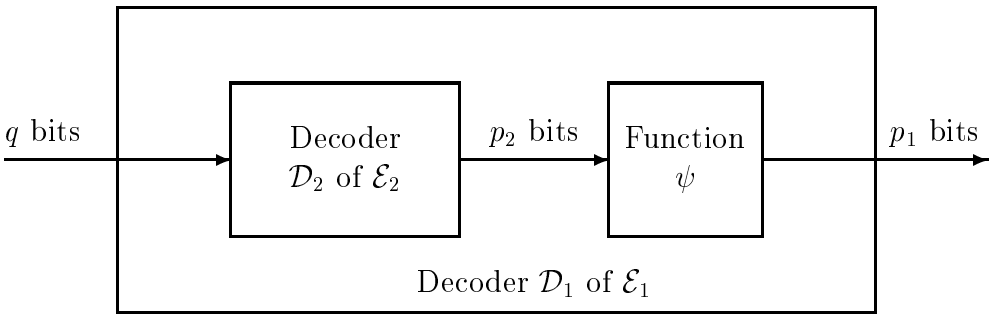

Figure 2: Encoder $\mathcal{E}_1$ observable from encoder $\mathcal{E}_2$.

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; yet, as we shall see, the special case of deterministic encoders is already elaborate as is. In addition, deterministic encoders are desirable in practice, especially if they belong to the sub-class of *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. The advantage of using block decodable (deterministic) encoders in practice is limiting error propagation and having simple decoding structure. We will also assume that our encoders are *irreducible*, namely, every state is reachable from every other state in the state diagram. 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 (which applies to irreducible deterministic encoders) 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 2.5 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 [7].

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 $e_1 e_2 \ldots e_\ell$ such that $\sigma_G(e_{j+1}) = \tau_G(e_j)$ for $i = 1, 2, \ldots, \ell - 1$. 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., [7, 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., [12, 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. If G is a lossless (for instance, deterministic) presentation of S, then

$$\mathsf{cap}(S) \;=\; \log \lambda(A_G) \tag{1}$$

(see [7, Section 3.2.2], [9]). Table 5.4 in [4, 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 (1) that

$$\mathsf{cap}(S^q) \;=\; q \cdot \mathsf{cap}(S) \;. \tag{2}$$

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. 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 [12].

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. We will make use of the following result from [8]:

Lemma 2.1 *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. A graph G is called *reduced* if no two states in G are follower-set equivalent.

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 [7, Section 2.6.4]).

Theorem 2.2 *Let S be an irreducible constraint.*

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

(b) The follower sets of the states in any irreducible deterministic presentation of S coincide with the follower sets of the states of the Shannon cover.

We will also make use of the following lemma.

Lemma 2.3 *[8] 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 and is denoted by $\mu(G)$. Given an integer $m \geq \mu(G)$, each state u in G can be associated with a set $W_G^{[m]}(u)$ 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 2.4 *Let G be an irreducible deterministic graph with finite memory and let m be an integer not smaller than $\mu(G)$. There is an edge $u \xrightarrow{a} u'$ in G if and only if there are words $\mathbf{w} \in W_G^{[m]}(u)$ and $\mathbf{w}' \in W_G^{[m]}(u')$ 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* [6]. 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 $\lambda(A_{\mathcal{E}}) = n$ [12, Ch. 1] and, so, by (1) we have $\mathsf{cap}(S(\mathcal{E})) = \log n$. The inequality $\mathsf{cap}(S(\mathcal{E})) \geq \mathsf{cap}(S)$ thus implies

$$\mathsf{cap}(S) \;\geq\; \log n \;. \tag{3}$$

Conversely, it was shown by Adler, Coppersmith and Hassner in [1] that (3) is also a *sufficient* condition for having an (S, n)-encoder with finite anticipation.

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 $\mathbf{s} = s_1 s_2 \ldots s_\ell$ defines a path π of length ℓ in $\mathcal{E}$ starting at u_0, and the image of $\mathbf{s}$ is the word $\mathbf{w} = w_1 w_2 \ldots w_\ell$ 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.

A *rate $p : q$ finite-state encoder for S* is a tagged $(S^q, 2^p)$-encoder, where we assume that the input tags are binary p-blocks. By (2) and (3) it follows that a rate $p : q$ finite-state encoder for S exists if and only if $p/q \leq \mathsf{cap}(S)$.

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 (into the future) 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 [7, Section 3.5].

Proposition 2.5 *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 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. A block decodable encoder is necessarily deterministic.

2.6 Observable 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 more complex as it might need to depend on more information beside the current tag reconstructed by $\mathcal{D}_2$ (but see Example 3.1).

We will assume in our discussion that $\mathcal{E}_1$ and $\mathcal{E}_2$ are irreducible. Otherwise, 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 (3) 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 $\mathcal{E}_1$).

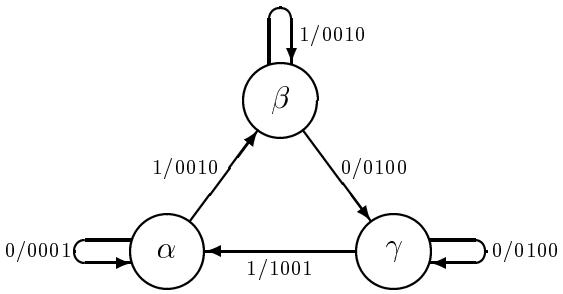

Figure 3: Rate $1:4$ finite-state encoder for $S_{(2,3)}$.

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

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

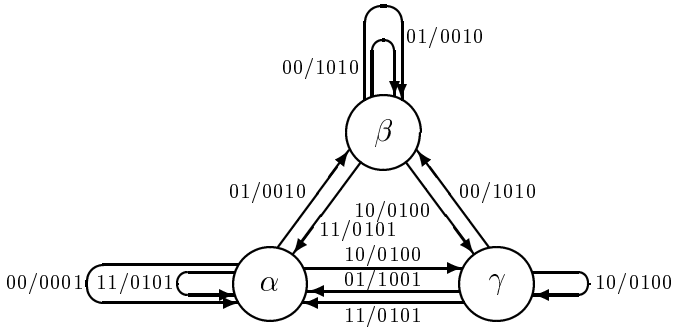

Figure 4: Rate $2:4$ finite-state encoder for $S_{(1,3)}$.

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

$$g_1(a) = g_1(c) = 0 \quad \text{and} \quad g_1(b) = g_1(d) = 1 \,. \tag{4}$$

By Theorem 2.2, 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).

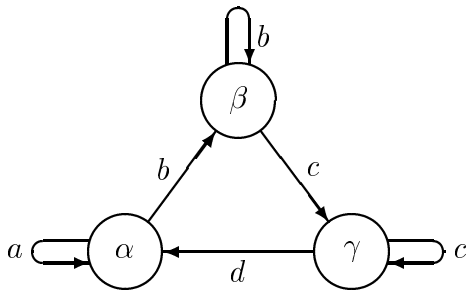

Figure 5: Graph $\mathcal{E}_1$ for Example 3.2.

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

$$g_2(a) = g_2(d) = 0\ , \quad g_2(b) = 1\ , \quad \text{and} \quad g_2(c) = 2 \tag{5}$$

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

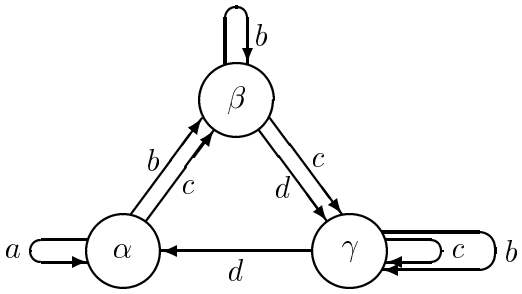

Figure 6: Graph $\mathcal{E}_2$ for Example 3.2.

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 (4), and let $\mathcal{E}_2'$ be a block decodable $(S_2,3)$-encoder with the block decoder g_2 defined by (5). 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 Let $S_1 = S_{(3,7)}$ and let G_1 be the Shannon cover of S_1. We number the states of G_1 from 0 to 7, as shown in Figure 1. We construct a graph H_1 from G_1^{14} by first redirecting into state 6 all the incoming edges to states 3, 4, and 5, and then deleting states 3–5 and 7, along with their incident edges. Clearly, H_1 is deterministic. Furthermore, since $\mathcal{F}_{G_1^{14}}(6) \subset \mathcal{F}_{G_1^{14}}(i)$ for $i \in \{3,4,5\}$, all words generated by H_1 are still in S_1^{14}. We rename the states 0, 1, 2, and 6 in H_1 as u_0, u_1, u_2, and u_3. The adjacency

matrix of H_1 is given by

$$A_{H_1} = \begin{bmatrix} 9 & 7 & 6 & 11 \\ 12 & 9 & 7 & 15 \\ 17 & 12 & 9 & 20 \\ 13 & 10 & 7 & 14 \end{bmatrix}$$

and the minimum degree of H_1 is 33.

Table 1 lists the number of words of length 14 in $S_{(3,7)}$ that actually appear as labels of edges in H_1. The words are classified according to the lengths of their first and last runs of 0's. For example, there are $1 + 7 + 4 = 12$ words of length 14 in $S_{(3,7)}$ that start

	Last runlength			
First runlength	0	1	2	3–6
7	1	1	1	1
6	1	1	1	1
3–5	7	5	4	9
2	4	3	2	5
1	6	4	3	6
0	7	6	4	8

Table 1: Number of words of length 14 in $S_{(3,7)}$ with given first and last runlengths.

with a run of length between 2 and 6 and end with a run of length 0. These words label the edges from u_1 to u_0 in H_1 and, indeed, the respective entry in A_{H_1} equals 12.

Let $S_2 = S_{(2,13)}$ and let G_2 be the Shannon cover of S_2, with state numbering as in Figure 1. We construct a graph H_2 from G_2^{14} by redirecting into state 11 all the incoming edges to states 3 through 10, and then deleting states 3–10, 12, and 13, along with their incident edges. Again, H_2 is a deterministic graph and $S(H_2) \subseteq S_2^{14}$. We rename the states 0, 1, 2, and 11 in H_2 as v_0, v_1, v_2, and v_3. The adjacency matrix of H_2 is given by

$$A_{H_2} = \begin{bmatrix} 41 & 28 & 19 & 40 \\ 59 & 41 & 28 & 58 \\ 87 & 59 & 41 & 85 \\ 60 & 41 & 28 & 58 \end{bmatrix}$$

and the minimum degree of H_2 is 128. Table 2 lists the number of words of length 14 in $S_{(2,13)}$ that actually appear as labels of edges in H_2.

Edges in H_1 (resp. H_2) whose labels end with a run of length $i \in \{0, 1, 2\}$ terminate in state u_i (resp. v_i), and the remaining edges terminate in state u_3 (resp. v_3). This,

First runlength	Last runlength			
	0	1	2	3–11
13	1	0	0	0
12	0	1	0	0
3–11	27	18	13	27
2	13	9	6	13
1	19	13	9	18
0	28	19	13	27

Table 2: Number of words of length 14 in $S_{(2,13)}$ with given first and last runlengths.

combined with the containment $L_{H_1}(E_{H_1}(u_i) \subseteq L_{H_2}(E_{H_2}(v_i)$ for $i \in \{0,1,2,3\}$, implies that H_1 is nested in H_2, with state u_i coinciding with state v_i.

By deleting edges from H_1 and H_2, we can obtain an irreducible deterministic $(S_1^{14}, 2^5)$-encoder $\mathcal{E}_1$ which is nested in an irreducible deterministic $(S_2^{14}, 2^7)$-encoder $\mathcal{E}_2$. The rates of those encoders are $5/14 \approx 0.3571$ and $7/14 = 0.5$, respectively; for comparison, the respective capacity values of S_1 and S_2 are approximately 0.4057 and 0.5485. As we show in Appendix A, we can tag $\mathcal{E}_1$ and $\mathcal{E}_2$ so that they are block decodable and $\mathcal{E}_1$ is observable from $\mathcal{E}_2$. □

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 [5].

The capacity of the $(3, 10)$-RLL constraint is approximately 0.4460. By (3) there exist rate 7 : 16 encoders for $S_{(3,10)}$. However, by Proposition 2.5 it can be verified that 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 [5]; 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 [4, 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 $\mathcal{E}_1$. □

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

$$\mathcal{F}_{G*H}(\langle u, u' \rangle) \;=\; \mathcal{F}_G(u) \cap \mathcal{F}_H(u')\;. \tag{6}$$

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 2.1, there is an irreducible component H_1 in $G_1 * G_2$ such that $S_1 = S(H_1)$.

We now follow [10] and [11] 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

$$\langle u_0, v_0 \rangle \xrightarrow{a_1} \langle u_1, v_1 \rangle \xrightarrow{a_2} \ldots \xrightarrow{a_\ell} \langle u_\ell, v_\ell \rangle \tag{7}$$

in H only if there is a path

$$v_0 \xrightarrow{a_1} v_1 \xrightarrow{a_2} \ldots \xrightarrow{a_\ell} v_\ell \tag{8}$$

in G_2. Conversely, for every path (8) in G_2 there are $u_0, u_1, \ldots, u_\ell$ such that (7) 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 $\mathcal{E}_2' \leftarrow H_2$. By Theorem 2.2(b), the out-degree of every state in $\mathcal{E}_1'$ is equal to the out-degree of some state in the Shannon cover $\mathcal{E}_1$ and is therefore n_1. The graph $\mathcal{E}_1'$ is thus an (S_1, n_1)-encoder. In a similar way, $\mathcal{E}_2'$ is an (S_2, n_2)-encoder. □

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

$$\mathcal{F}_{H_1}(\langle u, v\rangle) \ = \ \mathcal{F}_{G_1}(u) \ . \tag{9}$$

By the construction of H_1 (in particular, since H_1 is irreducible and deterministic), it suffices to prove (9) 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 2.2(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

$$\mathcal{F}_{H_2}(\langle u, v\rangle) \;=\; \mathcal{F}_{G_2}(v)\ . \tag{10}$$

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 (9) 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 (10) 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 (6) the equality $\mathcal{F}_{\mathcal{E}''_1}(\langle u'_\psi, v'_\psi\rangle) = \mathcal{F}_{\mathcal{E}_1}(u'_\psi)$; so, by Lemma 2.3(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 emphasize 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 property 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 $\mathcal{E}'_2$.

The topic of generalizing Proposition 4.3 to encoders with finite anticipation is an

ongoing research work with Brian Marcus. It turns out that (unlike the deterministic case) nesting of encoders that are not necessarily deterministic may force an increase of the anticipation compared to non-nested encoders. Specifically, we show in Appendix B 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; yet, 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.

5 Construction of deterministic nested encoders

In this section, we present an algorithm (contained in the proof of Theorem 5.1 below) that decides for any two given irreducible constraints $S_1 \subseteq S_2$ and positive integers n_1 and n_2 whether there exists an irreducible deterministic (S_1, n_1)-encoder that is nested in an irreducible deterministic (S_2, n_2)-encoder. Furthermore, the algorithm provides such encoders whenever they exist. Our discussion makes use of the term (G, n)-subgraph of a graph G, as defined below.

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 out-degree at least n. Clearly, for a given graph G, there are values of n for which (G, n)-subgraphs do not exist. 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 $n \leq \lambda(A_G)$ [7, 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 [7, 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 [7, Section 3.1.4], when given as input the matrix A_G, the integer n, and the all-one vector.

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 as it implies an algorithm for finding the nested encoders. We prove the theorem using the following two lemmas. The first lemma is a somewhat stronger version of Proposition 2.5.

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 2.1 and 2.3(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 $Z(u) \stackrel{a}{\longrightarrow} Z(u')$ if and only if $u \stackrel{a}{\longrightarrow} u'$ is an edge in G and there is $v \in Z(u)$ and $v' \in V_\mathcal{E}$ such that $v \stackrel{a}{\longrightarrow} v'$ 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 2.3(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

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

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

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

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 $Z(u_k) \stackrel{a_k}{\longrightarrow} Z(u_{k+1})$ 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 the sink $\mathcal{E}_2$. □

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 $S(\mathcal{E}_1) \subseteq S(H_1)$.

The 'if' part follows by the next algorithm which effectively finds an irreducible deterministic (S_1, n_1)-encoder $\mathcal{E}_1$ that is nested in an irreducible deterministic (S_2, n_2)-encoder $\mathcal{E}_2$.

1. Find graphs H_1 and H_2 such that H_2 is a (G_2, n_2)-component and H_1 is a $(G_1 * H_2, n_1)$-component.

[Such components can be found by using the method described at the end of Section 5.1. If no such graphs H_1 and H_2 exist then, by the 'only if' part, the encoders $\mathcal{E}_1$ and $\mathcal{E}_2$ do not exist either.]

2. Find irreducible deterministic graphs H_1' and H_2' such that $S(H_1') = S(H_1)$, $S(H_2') = S(H_2)$, and $H_1' \subseteq H_2'$.

   [The existence of the graphs H_1' and H_2' follows from an application of Lemma 4.2 to $S(H_1)$ and $S(H_2)$, and the proof of that lemma implies an algorithm to find those graphs. Note that by Theorem 2.2(b), every state in H_i' is follower-set equivalent to some state in H_i, $i = 1, 2$; so, the minimum degree of H_i' is at least n_i.]

3. Let $\mathcal{E}_1$ 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'.

   [Note that $\mathcal{E}_1$ is also a subgraph of H_2'.]

4. Using the method described in the proof of Lemma 5.3, obtain $\mathcal{E}_2$ as an n_2-regular irreducible subgraph of H_2' that contains $\mathcal{E}_1$ as a subgraph.

It is easily seen that each $\mathcal{E}_i$ is an irreducible deterministic (S_i, n_i)-encoder, and $\mathcal{E}_1$ is nested in $\mathcal{E}_2$. □

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 Theorem 5.1 are only necessary.

6 Nested encoders and 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 7.

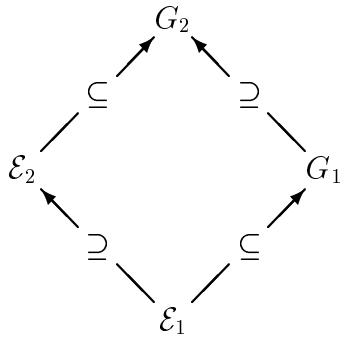

Figure 7: Diamond condition set.

6.1 Nested encoders within the Shannon covers

Proposition 2.5 implies that once there exists a deterministic (S, n) encoder for a given constraint S, then there must also exist a deterministic (S, n) encoder which is a subgraph of the Shannon cover G of S. The next theorem generalizes this result through the diamond condition set to the case of nested deterministic encoders, under the assumption of finite memory.

Theorem 6.1 *Let S_1 and S_2 be two irreducible constraints such that $S_1 \subseteq S_2$ and S_2 has finite memory. 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).*

We point out that powers of (d, k)-RLL constraints fall into the category studied here: first, $S_{(d,k)}^q$ has finite memory at most k; secondly, if $S_1 = S_{(d_1,k_1)}^q$ is contained in $S_2 = S_{(d_2,k_2)}^q$, then $d_1 \geq d_2$, $k_1 \leq k_2$, and the Shannon cover of S_1 is a subgraph of the Shannon cover of S_2. The proof of Theorem 6.1 makes use of the following definition and lemma.

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. Given an integer $m \geq \mu(G)$, we construct next a subgraph $H_{\mathcal{E}}^{[m]}$ of G which we call the *mth order super-graph of $\mathcal{E}$ (with respect to G).*

Recall the notation $W_G^{[m]}(u)$ of the set of all words of length m that can be generated by paths in G that terminate in a state $u \in V_G$. Note that since $m \geq \mu(G)$, the sets $W_G^{[m]}(u)$ are disjoint for distinct states u.

For a word $\mathbf{w}$ 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 $H_{\mathcal{E}}^{[m]}$ is defined as follows. For a state $u \in V_G$, let $T_{G,\mathcal{E}}^{[m]}(u) = T^{[m]}(u)$ denote the union $\cup_{\mathbf{w} \in W_G^{[m]}(u)} V_{\mathcal{E}}(\mathbf{w})$. The states of $H_{\mathcal{E}}^{[m]}$ are all the nonempty sets $T^{[m]}(u)$, $u \in V_G$. We endow $H_{\mathcal{E}}^{[m]}$ with an edge $T^{[m]}(u) \stackrel{a}{\longrightarrow} T^{[m]}(u')$ if and only if there are words $\mathbf{w} \in W_G^{[m]}(u)$, $\mathbf{w}' \in W_G^{[m]}(u')$ 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 6.2 *Let G be an irreducible deterministic graph of finite memory that presents a constraint S and, for an integer $m \geq \mu(G)$, let $H = H_{\mathcal{E}}^{[m]}$ be the mth order super-graph of an irreducible deterministic (S,n)-encoder $\mathcal{E}$ with respect to G. Then*

(a) H is isomorphic to some subgraph of G, with each state $T^{[m]}(u)$ in H identified with state u in G.

(b) H is irreducible.

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

(d) The minimum degree of H is at least n.

Proof. (a) We show that if $T^{[m]}(u) \stackrel{a}{\longrightarrow} T^{[m]}(u')$ is an edge in H, then $u \stackrel{a}{\longrightarrow} u'$ is an edge in G. Indeed, if $T^{[m]}(u) \stackrel{a}{\longrightarrow} T^{[m]}(u')$ is in H, then there are words $\mathbf{w} \in W_G^{[m]}(u)$, $\mathbf{w}' \in W_G^{[m]}(u')$ 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 2.4 that there is an edge $u \stackrel{a}{\longrightarrow} u'$ in G.

(b) Let $T^{[m]}(u)$ and $T^{[m]}(u')$ be two states in H and let $\mathbf{w}$ and $\mathbf{w}'$ be words in $W_G^{[m]}(u)$ and $W_G^{[m]}(u')$, 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

$$v_0 \stackrel{a_1}{\longrightarrow} v_1 \stackrel{a_2}{\longrightarrow} \ldots \stackrel{a_\ell}{\longrightarrow} v_\ell \tag{11}$$

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 $T^{[m]}(u_{j-1}) \stackrel{a_j}{\longrightarrow} T^{[m]}(u_j)$

in H, where each u_j is the unique state in G such that $\mathbf{w}_j \in W_G^{[m]}(u_j)$. In particular, $u_0 = u$ and $u_\ell = u'$. We conclude that there is a path

$$T^{[m]}(u) \equiv T^{[m]}(u_0) \xrightarrow{a_1} T^{[m]}(u_1) \xrightarrow{a_2} \ldots \xrightarrow{a_\ell} T^{[m]}(u_\ell) \equiv T^{[m]}(u') \tag{12}$$

in H.

(c) Let (11) be a path in $\mathcal{E}$ and let $u_0 \in V_G$ and $\mathbf{w} \in W_G^{[m]}(u_0)$ be such that $v_0 \in V_{\mathcal{E}}(\mathbf{w})$. The proof of (b) implies that there is also a path (12) in H. Hence, $S(\mathcal{E}) \subseteq S(H)$.

(d) Let v be a state in $T^{[m]}(u)$; that is, $v \in V_{\mathcal{E}}(\mathbf{w})$ for some $\mathbf{w} \in W_G^{[m]}(u)$. Suppose there is an edge $v \xrightarrow{a} v'$ in $\mathcal{E}$. Then $\mathbf{w}a \in S$ and, so, there is an outgoing edge labeled a from u in G, with a terminal state u' (say). Furthermore, if $\mathbf{w}'$ is the suffix of length m of $\mathbf{w}a$, then $v' \in V_{\mathcal{E}}(\mathbf{w}')$ and $\mathbf{w}' \in W_G^{[m]}(u')$. Therefore, by construction, there is an edge $T^{[m]}(u) \xrightarrow{a} T^{[m]}(u')$ in H. Hence, the out-degree of $T^{[m]}(u)$ in H is at least the out-degree of v in $\mathcal{E}$. □

Proof of Theorem 6.1. Fix an integer $m \geq \mu(G_2) \geq \mu(G_1)$. For $i = 1, 2$, let $H_{\mathcal{E}_i'}^{[m]}$ be the mth order super-graph of $\mathcal{E}_i'$ with respect to G_i. By Lemma 6.2(a), we can regard $H_{\mathcal{E}_i'}$ as a subgraph of G_i, with state $T_{G_i,\mathcal{E}_i'}^{[m]}(u)$ in $H_{\mathcal{E}_i'}^{[m]}$ identified with state u in G_i. We show that $H_{\mathcal{E}_1'}^{[m]}$ is a subgraph of $H_{\mathcal{E}_2'}^{[m]}$. For every state $u \in V_{G_1}$ we have $W_{G_1}^{[m]}(u) \subseteq W_{G_2}^{[m]}(u)$, and for every $\mathbf{w} \in W_{G_1}^{[m]}(u)$ we have $V_{\mathcal{E}_1'}(\mathbf{w}) \subseteq V_{\mathcal{E}_2'}(\mathbf{w})$. Hence, $T_{G_1,\mathcal{E}_1'}^{[m]}(u) \subseteq T_{G_2,\mathcal{E}_2'}^{[m]}(u)$, and for every edge $T_{G_1,\mathcal{E}_1'}^{[m]}(u) \xrightarrow{a} T_{G_1,\mathcal{E}_1'}^{[m]}(u')$ in $H_{\mathcal{E}_1'}^{[m]}$ we also have an edge $T_{G_2,\mathcal{E}_2'}^{[m]}(u) \xrightarrow{a} T_{G_2,\mathcal{E}_2'}^{[m]}(u')$ in $H_{\mathcal{E}_2'}$.

By Lemma 6.2(d), there is an n_1-regular subgraph $\mathcal{E}_1$ of $H_{\mathcal{E}_1'}^{[m]}$, and by shifting to an irreducible sink we can assume that $\mathcal{E}_1$ is irreducible. As such, $\mathcal{E}_1$ is an irreducible deterministic (S_1, n_1)-encoder that is a subgraph of $H_{\mathcal{E}_2'}^{[m]}$. By Lemmas 5.3 and 6.2(d), there is an irreducible n_2-regular subgraph $\mathcal{E}_2$ of $H_{\mathcal{E}_2'}^{[m]}$ such that $\mathcal{E}_1 \subseteq \mathcal{E}_2$. The graph $\mathcal{E}_2$ is the desired (S_2, n_2)-encoder. □

The next example shows that Theorem 6.1 does not necessarily hold when S_2 has infinite memory.

Example 6.1 Let S_2 be the constraint presented by the graph H_2 in Figure 8, and let S_1 be the constraint presented by the subgraph H_1 of H_2 that is induced by states γ and δ. Note that H_1 and H_2 are the Shannon covers of S_1 and S_2, respectively, and that S_2 has infinite memory (the word $a_1 a_1 \ldots$ of arbitrary length can be generated by a path that ends in state α, and also by a path that ends in state δ). We choose $n_1 = 2$ and $n_2 = 3$ and verify that the subgraph induced by states α and β is an $(S_2, 3)$-encoder,

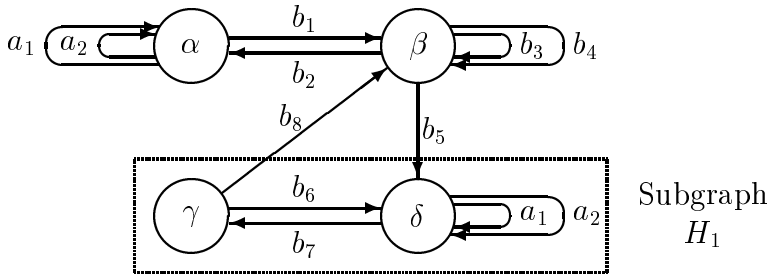

Figure 8: Graph H_2 for Example 6.1.

and the subgraph induced by state α (or state δ) is an $(S_1, 2)$-encoder. The subgraph of H_1 induced by state δ is the only $(S_1, 2)$-encoder nested in H_1, and that subgraph is not a subgraph of any $(S_2, 3)$-encoder within H_2. Hence, there are no encoders $\mathcal{E}_1$ and $\mathcal{E}_2$ such that the quadruple $(H_1, H_2, \mathcal{E}_1, \mathcal{E}_2)$ satisfies the diamond condition set with respect to $(S_1, S_2, 2, 3)$.

In fact, for this example, a stronger claim can be made: we show next that there is no quadruple $(G_1, G_2, \mathcal{E}_1, \mathcal{E}_2)$ that satisfies the diamond condition set with respect to $(S_1, S_2, 2, 3)$, for any irreducible deterministic presentation G_2 of S_2 (i.e., G_2 is not necessarily the Shannon cover of S_2).

Indeed, suppose to the contrary that there is such a quadruple, and let u be a state that belongs to all four graphs G_1, G_2, $\mathcal{E}_1$, and $\mathcal{E}_2$. By Theorem 2.2(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. □

6.2 Nested encoders in the general finite-memory case

Recall that Theorem 6.1 assumes irreducible constraints $S_1 \subseteq S_2$ such that S_2 has finite memory and an irreducible presentation of S_1 is nested in the Shannon cover of S_2. The next theorem does not assume the latter nesting.

Theorem 6.3 *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 6.3 is based on the following result.

Proposition 6.4 *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 nonempty 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 $G^{(k)}$ there exist irreducible deterministic graphs $G_A^{(k)}$, $G_B^{(k)}$, and $G_{\max}^{(k)}$, such that*
  1. *$S(G_{\max}^{(k)}) = S_{\max}$;*
  2. *$G_A^{(k)}$ is a subgraph of $G_{\max}^{(k)}$, and $S_A = S(G_A^{(k)})$;*
  3. *$G_B^{(k)}$ is a subgraph of $G_{\max}^{(k)}$, and $S_B = S(G_B^{(k)})$; and —*
  4. *$G^{(k)}$ is a subgraph of $G_A^{(k)}$ and $G_B^{(k)}$ within $G_{\max}^{(k)}$.*

The containment relationships among the constraints in Proposition 6.4 are shown in Figure 9. We point out that Proposition 6.4 does not necessarily hold if we remove the requirement that $|S_A \cap S_B| = \infty$. Indeed, suppose that $S_{\max}$ is presented by the graph $H_{\max}$ given in Figure 10. Let S_A be presented by the subgraph H_A of $H_{\max}$ induced by states α and β, and let S_B be presented by the subgraph H_B of $H_{\max}$ induced by states γ and δ. We have $S_A \cap S_B = \{b\}$, and $S_{\min}$ can be taken as the irreducible constraint that contains the empty word (this constraint is generated by a graph with one state and

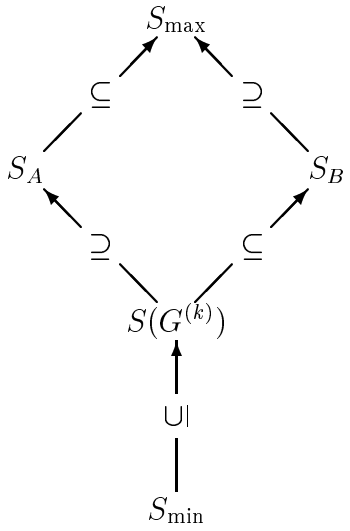

Figure 9: Constraints in Proposition 6.4.

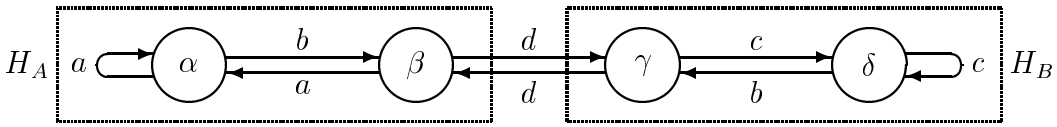

Figure 10: Graph $H_{\max}$.

no edges). Now, if Proposition 6.4 were to hold in this case, then the graphs $G_A^{(k)}$ and $G_B^{(k)}$ had to intersect in at least one state u in $G_{\max}^{(k)}$. As a state of $G_A^{(k)}$, there had to be outgoing edges from u labeled a and, as a state of $G_B^{(k)}$, there had to be outgoing edges from u labeled c. However, no state in $H_{\max}^{(k)}$—and hence in $G_{\max}^{(k)}$—generates both a and c.

The proof of Proposition 6.4 is rather long and technical and we therefore omit it (the proof can be found in [3]). Next we present a proof of Theorem 6.3, based on Proposition 6.4.

Proof of Theorem 6.3. 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 6.4 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 $G^{(k)}$, $G_A^{(k)}$, $G_B^{(k)}$ and $G_{\max}^{(k)}$. 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 $G_A^{(k)}$, $G_B^{(k)}$, and $G_{\max}^{(k)}$, respectively.

Note that $G_B^{(k)} \equiv \mathcal{E}_2$ is n_2-regular since, by Theorem 2.2(b), the follower sets of the states in $G_B^{(k)}$ 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 2.5, 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 $\mathcal{E}_1$. □

Appendix A

Referring to Example 3.3, we show here how to obtain tagged encoders $\mathcal{E}_1$ and $\mathcal{E}_2$ that are block decodable and $\mathcal{E}_1$ is observable from $\mathcal{E}_2$.

Recall that the labels of $\mathcal{E}_1$ are words of length 14 in $S_1 = S_{(3,7)}$ and the labels of $\mathcal{E}_2$ are words of length 14 in $S_2 = S_{(2,13)}$. To guarantee block decodability, edges with the same label must have the same tag assignment. Table 3 suggests a tag assignment to words that yields block decodability and observability. The tags are seven bits long, ranging

	Words from $S_{(3,7)}$		Other words from $S_{(2,13)}$	
First runlength	Number	Tags (hex.)	Number	Tags (hex.)
13	0		1	24
12	0		1	25
8–11	0		53	26–5a
7	4	00–03		
6	4	04–07		
3–5	24	08–1f		
2	4	20–23	37	5b–7f
1	7	24–2a	0	
0	25	2b–43	55	00–1f; 44–5a

Table 3: Tag assignment for Example 3.3.

(in hexadecimal notation) between 00 and 7f. The second column in Table 3 contains the number of words of length 14 in $S_{(3,7)}$ that are used as labels of edges of $\mathcal{E}_1$; those words are classified according to their first runlength (compare with Table 1). Similarly, the fourth column contains the number of *additional* words in $S_{(2,13)}$ that are used as labels of edges of $\mathcal{E}_2$ (compare with Table 2). The third and fifth columns contain the tag assignment of all labels chosen. The function ψ just truncates the two most significant bits to obtain tags in the range 00–1f.

As an example, the labels of the outgoing edges from state u_3 in $\mathcal{E}_1$ are words of length 14 in $S_{(3,7)}$ whose first run has length at most 1. From the last two rows in Table 1 it follows that there exist 44 such words, out of which $7+25=32$ words appear in Table 3. These 32 words label the outgoing edges from state u_3 in $\mathcal{E}_1$, and the tag assignment of these edges in $\mathcal{E}_2$ ranges between 24 and 43 (hex.); after truncation, those tags exhaust all the values between 00 and 1f. The outgoing edges from state v_3 in $\mathcal{E}_2$ are words of length 14 in $S_{(2,13)}$ whose first run has length at most 2. The last three rows in Table 3 contain $4+7+25+37+55=128$ such words (including the $7+25=32$ words that are generated from state u_3 in $\mathcal{E}_1$). The tag assignment of those words ranges over all values between 00 and 7f.

There are states for which Table 3 allows freedom in choosing the labels. For example, there are $24+4+7=35$ words in the table that can label the outgoing edges from state u_2 in $\mathcal{E}_1$; we take 32 of those words such that their assigned tags, when truncated by ψ, are distinct. Similarly, there are 213 words that can label the outgoing edges from state v_2 in $\mathcal{E}_2$, of which we need to take only 128 words (including the 32 words that were selected for u_2). In fact, the 128 words can be selected so that state v_2 becomes follower-set equivalent to state v_3. This allows us to merge those two states, in which case $\mathcal{E}_1$ will no longer be nested in $\mathcal{E}_2$, although it will still be observable from it (see the remark at the end of Example 3.1). So, nesting can be used as a tool to obtain observability, although ultimately the nesting property is not necessarily maintained.

Appendix B

We present 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.

Let S_1 be the constraint presented by the graph G_1 in Figure 11. The out-degree of

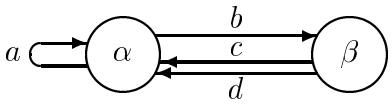

Figure 11: Shannon cover G_1 of S_1.

each 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 12.

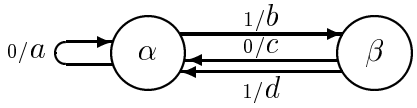

Figure 12: Tagged $(S_1, 2)$-encoder $\mathcal{E}_1$.

Let S_2 be the constraint presented by the graph G_2 in Figure 13. It is easy to

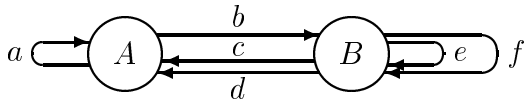

Figure 13: Shannon cover G_2 of S_2.

check that $\mathsf{cap}(S_2) = \log \lambda(A_{G_2}) = \log 3$. By Proposition 2.5 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 14.

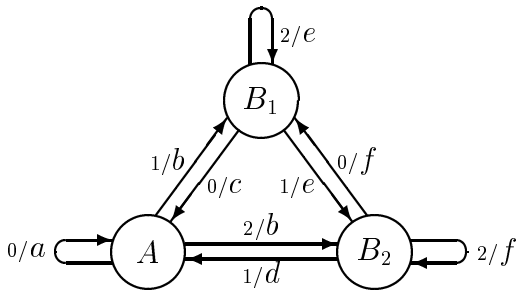

Figure 14: Tagged $(S_2, 3)$-encoder $\mathcal{E}_2$.

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 Table 4.

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 $\psi(1) = \psi(2) = 1$.

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

Table 4: Decoder $\mathcal{D}_2$ for $\mathcal{E}_2$.

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 the anticipation of $\mathcal{E}_2'$ must be 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 13 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 11 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.

The lower bound 2 on the anticipation is tight in this example, as demonstrated in Figure 15, where $\mathcal{E}_1$ is nested in a five-state $(S_2, 3)$-encoder with anticipation 2. In fact,

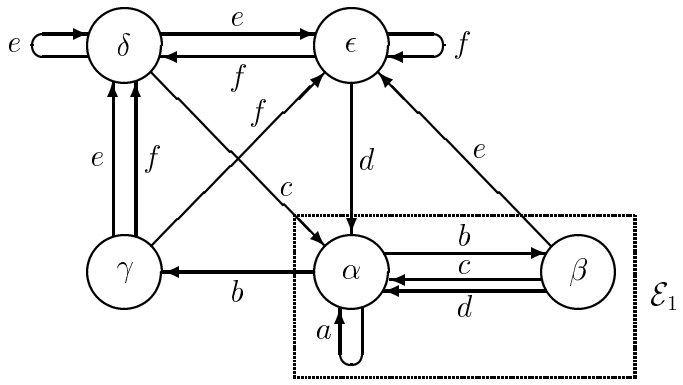

Figure 15: $\mathcal{E}_1$ nested in an $(S_2, 3)$-encoder with anticipation 2.

$\mathcal{E}_1$ is nested also in the three-state $(S_2, 3)$-encoder shown in Figure 16; however, the latter encoder has *infinite* anticipation (though it is still a lossless graph).

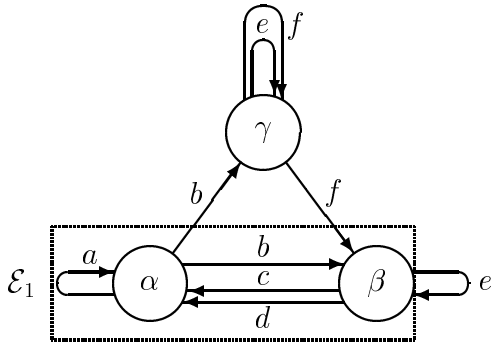

Figure 16: $\mathcal{E}_1$ nested in an $(S_2, 3)$-encoder with infinite anticipation.

Acknowledgment

The authors thank Brian Marcus for his thoughtful reading and helpful comments.

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.

[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 and Trademark Office.

[3] J. Hogan, R.M. Roth, G. Ruckenstein, *Nested input-constrained codes,* Hewlett-Packard Laboratories TR 98-169 (1998).
`http://www.hpl.hp.com/techreports/98/HPL-98-165.html`.

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

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

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

[7] B.H. Marcus, R.M. Roth, P.H. Siegel, *Constrained Systems and Coding for Recording Channels, Handbook of Coding Theory,* V.S. Pless and W.C. Huffman (Editors), Elsevier, Amsterdam, 1998, 1635–1764.

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

[9] 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.

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

[11] G. Ruckenstein, R.M. Roth, *Lower bounds on the anticipation of encoders for input-constrained channels,* submitted to *IEEE Trans. Inform. Theory.* See also *IEEE Int'l Symp. on Information Theory,* Ulm, Germany (June 1997), p. 140.

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