Codes on Graphs, Normal Realizations, and Partition Functions

G. David Forney, Jr.[1]

Workshop on Counting, Inference and Optimization
Princeton, NJ

November 3, 2011

[1] Joint work with: Heide Gluesing-Luerssen, Yongyi Mao, Pascal Vontobel

Codes on graphs

Origins: linear **convolutional codes** and encoders

- Linear time-invariant systems over finite fields
- Finite-state
  - Trellis: finite-state transition diagrams spread out in time
  - Underlying graphical model is a simple chain graph, **cycle-free**

Flowering: **capacity-approaching codes** and realizations

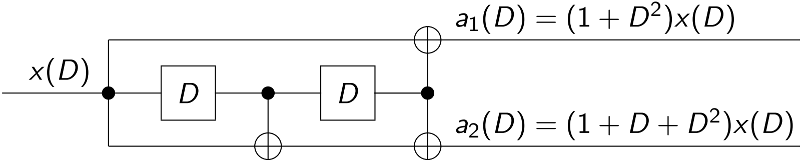
- Turbo codes, low-density parity-check (LDPC) codes
  - Usually described by graphical representations
  - For capacity-approaching codes, graphs must have **cycles**

Linear convolutional codes

Linear convolutional encoders:

linear systems (filters) over finite fields

Example (4-state rate-$\frac{1}{2}$ binary linear convolutional encoder):

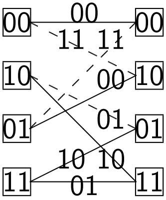


Convolutional codes and local constraints

Convolutional encoder: specified by

- Symbol alphabets $\mathcal{A}_k$ (k = time index)
- State spaces $\mathcal{S}_k$
- Local constraint codes $\mathcal{C}_k \subseteq \mathcal{S}_k \times \mathcal{A}_k \times \mathcal{S}_{k+1}$ **(trellis sections)**

Example ($\mathcal{A}_k = \mathcal{S}_k = (\mathbb{F}_2)^2$):

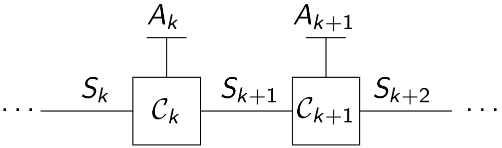


Linear $(6,3)$ constraint code $\mathcal{C}_k$, generated by $\begin{matrix} 00 & 11 & 10 \\ 10 & 01 & 01 \\ 01 & 11 & 00 \end{matrix}$

Normal graph of a convolutional encoder

Normal graph:

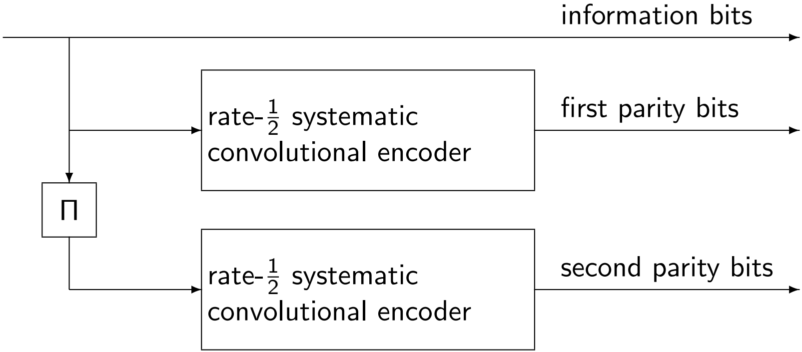
- Constraint codes $\mathcal{C}_k$ are represented by vertices
- Each symbol alphabet $\mathcal{A}_k$ is involved in precisely one constraint code; represented by a "dangling edge" (half-edge)
- Each state space $\mathcal{S}_k$ is involved in precisely two constraint codes; represented by an ordinary edge



Trellis realization $\Leftrightarrow$ simple chain graph (cycle-free)

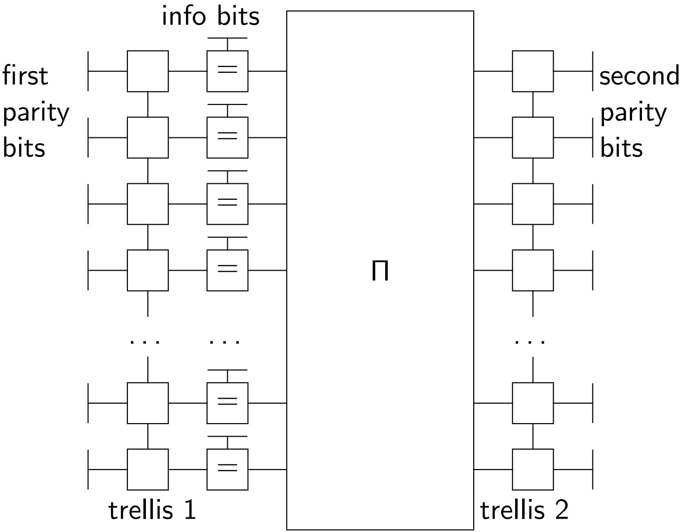
Turbo codes (Berrou *et al.*, 1993)

Rate-$\frac{1}{3}$ Berrou-type turbo code:



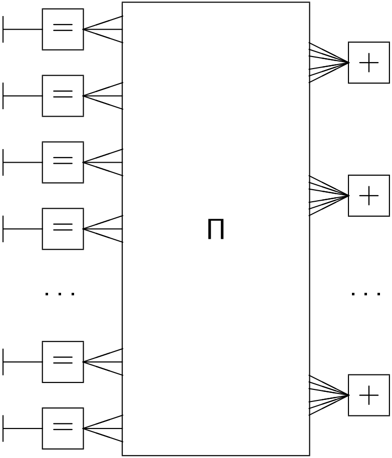
Approaches Shannon Limit to within 1 dB!

Normal graph of a Berrou-type turbo code



LDPC codes (Gallager, 1961; rediscovered 1994)

Normal graph of a regular $(3, 6)$ low-density parity-check code:



Approaches Shannon Limit to within 1 dB!

Behavioral realizations of linear codes and systems

Linear behavioral realization of a linear code $\mathcal{C}$:

- **External variables** $\mathcal{A}_i$ (code symbols)
- **Internal variables** $\mathcal{S}_j$ ("state variables")
- **Constraint codes** $\mathcal{C}_k$, each involving a subset of the variables
- **Linear**: variable alphabets are vector spaces;
  constraint codes are linear (*i.e.*, vector subspaces)

Behavior $\mathfrak{B}$: set of all $(\mathbf{a}, \mathbf{s})$ that satisfy all constraints

Code $\mathcal{C}$: set of all $\mathbf{a}$ that appear in some $(\mathbf{a}, \mathbf{s}) \in \mathfrak{B}$

- <u>One-to-one</u>: the projection map $\mathfrak{B} \to \mathcal{C}$ is one-to-one

Normal realizations

Normal realization = realization with **degree restrictions**:

- Each external variable $\mathcal{A}_i$ is involved in one constraint code
- Each internal variable $\mathcal{S}_j$ is involved in two constraint codes
- Note: a trellis realization is inherently normal

Lemma: any realization can be "normalized" as follows:

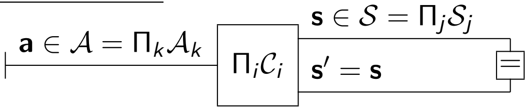
- For each appearance of each variable, introduce a dummy variable ("**replica**")
- For each variable, introduce a **repetition constraint code** that constrains all replicas to be equal

Normal graph: natural graphical model of a normal realization

- Constraint codes $\mathcal{C}_k$: vertices
- External variables $\mathcal{A}_i$: half-edges (dangling edges, dongles)
- Internal variables $\mathcal{S}_j$: ordinary edges

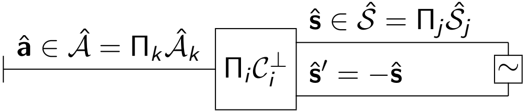
Normal graph duality theorem (NGDT)

Generic normal graph:



Dual normal graph:

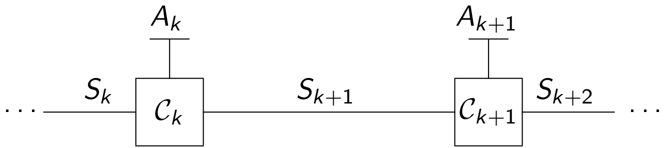
- replace alphabets $\mathcal{A}_k, \mathcal{S}_j$ by dual groups/spaces $\hat{\mathcal{A}}_k, \hat{\mathcal{S}}_j$
- replace constraint codes $\mathcal{C}_i$ by orthogonal codes $\mathcal{C}_i^\perp$
- insert a sign inverter $\sim$ into every internal variable edge



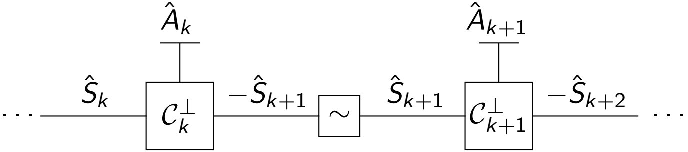
NGDT: the dual graph realizes the orthogonal code $\mathcal{C}^\perp$.
Elementary group-theoretic proof: projection-subcode duality.

Example: Orthogonal convolutional codes

Normal graph of a trellis realization, with constraint codes $\mathcal{C}_k$:



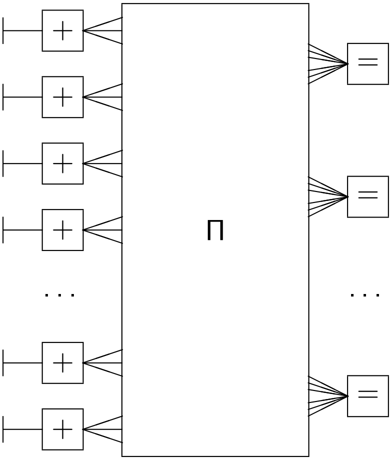
Normal graph of dual realization, with orthogonal constraint codes $\mathcal{C}_k^\perp$ and sign inverters (omit for binary):



NGDT: This dual trellis realization generates $\mathcal{C}^\perp$.

Example: Low-density generator matrix codes

An LDGM code is the dual of an LDPC code. Dual realization:



And now for something holographic...

Normal factor graph duality theorem (NFGDT)

- A **normal partition function** is represented by a **normal factor graph**
  - Partition function: a function in "sum of products" form
  - "Normal": same degree restrictions as in normal realizations
  - Any partition function can be "normalized"
- NFGDT: the dual normal factor graph represents the dual partition function
  - Dual partition function: Fourier transform
- Corollary: Normal graph duality theorem

(Generalized) partition functions

Partition function: a function expressed in sum-of-products form

$$Z(\mathbf{x}) = \sum_{\mathbf{y}\in\mathcal{Y}} \prod_{k\in\mathcal{K}} f_k(\mathbf{x}_k, \mathbf{y}_k), \quad \mathbf{x} \in \mathcal{X},$$

- **External variables** X_i taking values x_i in alphabets $\mathcal{X}_i$; external configurations $\mathbf{x} \in \mathcal{X} = \prod_{i=1}^{m} \mathcal{X}_i$
- **Internal variables** Y_j taking values y_j in alphabets $\mathcal{Y}_j$; internal configurations $\mathbf{y} \in \mathcal{Y} = \prod_{j=1}^{n} \mathcal{Y}_j$
- **Factors** $f_k(\mathbf{x}_k, \mathbf{y}_k)$, where $\mathbf{x}_k \subseteq \mathbf{x}$, $\mathbf{y}_k \subseteq \mathbf{y}$

Normal partition functions

Normal partition function:
a partition function with normal degree restrictions

- All external variables are involved in precisely one factor
- All internal variables are involved in precisely two factors

Normalization: any partition function may be straightforwardly converted to an equivalent normal partition function by the following **replication procedure**:

- Replace each variable in each factor by a **replica** variable;
- Constrain all replicas of each variable to be equal by introducing an **equality indicator function** factor $\Phi_=$, which equals 1 when they are all the same, and 0 otherwise.

Graphical model of a normal partition function

Normal factor graph:

- Vertices: factors $f_k(\mathbf{x}_k, \mathbf{y}_k)$
- Ordinary edges: internal variables Y_j (involved in two factors)
- Half-edges: external variables X_i (involved in one factor)

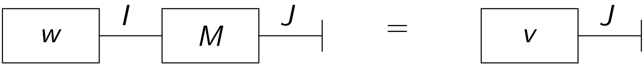
Example: Vector-matrix multiplication

Let $\mathbf{v} = \mathbf{w}M$; *i.e.*, $v_j = \sum_i w_i M_{ij}$

A normal partition function with

- Factors: w_i, M_{ij}
- External variable: J
- Internal variable: I

Normal factor graph:

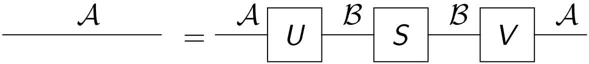


Einstein summation convention: sum over all variables that appear twice

Generalized holographic transformations

General approach: Let $U(a,b), S(b,b')$ and $V(b',a')$ be three factors whose concatenation USV is equivalent to the identity; *i.e.*,

$$\delta(a,a') = \sum_{b\in\mathcal{B},b'\in\mathcal{B}} U(a,b)S(b,b')V(b',a')$$



Then in any NFG, any edge may be replaced by such a concatenation USV without changing the partition function.

External variables may be transformed as well.

$\Rightarrow$ "Generalized Holant theorem" —Al-Bashabsheh and Mao

Special case: "Holant theorem" —Valiant

Example: normal factor graph duality theorem

Theorem (normal factor graph duality theorem):
Let $Z(\mathbf{x})$ be the partition function of a normal factor graph $\mathcal{G}$.
Define the **dual normal factor graph** $\hat{\mathcal{G}}$ as follows:

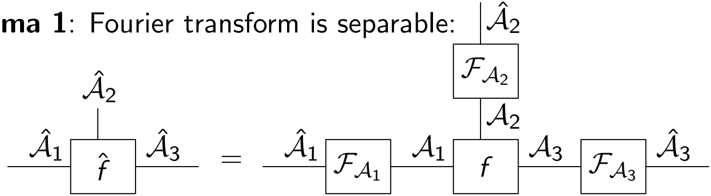
- Replace each variable alphabet in $\mathcal{G}$ by its dual alphabet;
- Replace each factor in $\mathcal{G}$ by its Fourier transform;
- Insert a sign inverter factor $\Phi_\sim$ into each ordinary edge.

Then the partition function of $\hat{\mathcal{G}}$ is the Fourier transform $\hat{Z}(\hat{\mathbf{x}})$.

—Al-Bashabsheh and Mao [IT, Feb. 2011]; Forney [IT, Mar. 2011]

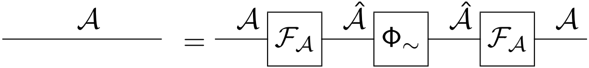
Lemmas for proof of NFGDT

Lemma 1: Fourier transform is separable:



$\hat{f}$: Fourier transform of complex-valued function f
$\mathcal{F}_{\mathcal{A}}$: Fourier kernel $\omega^{\langle a, \hat{a} \rangle}$ (from vector space $\mathcal{A}$ to dual space $\hat{\mathcal{A}}$)

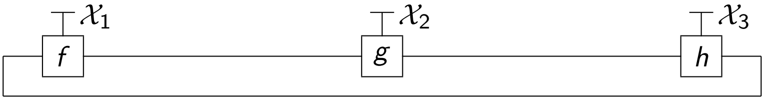
Lemma 2: $\mathcal{F}_{\mathcal{A}} \Phi_{\sim} \mathcal{F}_{\mathcal{A}} =$ identity $\Rightarrow$ "holographic" transformation:



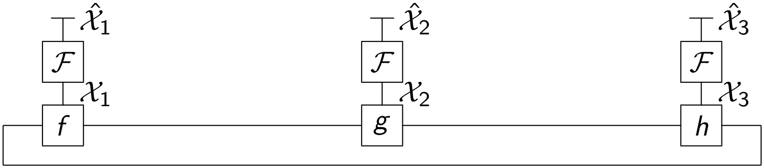
$\Phi_{\sim}$: sign inverter indicator function over $\hat{\mathcal{A}}$

NFGDT proof

Proof: Given a normal factor graph $\mathcal{G}$, partition function $Z(\mathbf{x})$:

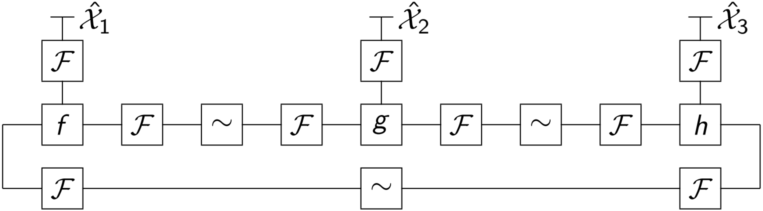


Step 1: apply Lemma 1 globally $\Rightarrow$ partition function $\hat{Z}(\hat{\mathbf{x}})$;

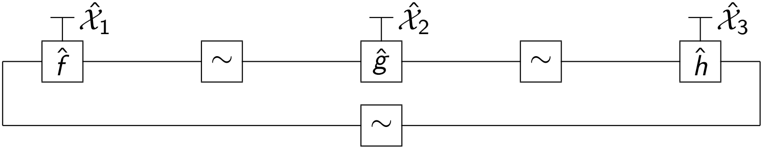


NFGDT proof (cont.)

Step 2: apply Lemma 2 to each edge; partition function still $\hat{Z}(\hat{\mathbf{x}})$;



Step 3: apply Lemma 1 locally; partition function still $\hat{Z}(\hat{\mathbf{x}})$. $\square$



Application of NFGDT: NGDT

The **normal graph duality theorem for linear codes** is a corollary of the normal factor graph duality theorem.

Conversion of a normal graph realizing $\mathcal{C}$ to a normal factor graph whose partition function is the indicator function $\Phi_{\mathcal{C}}$ of $\mathcal{C}$:

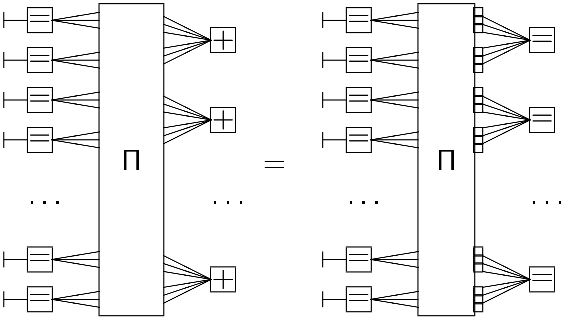
- Replace each constraint code $\mathcal{C}_i$ by its indicator function $\Phi_{\mathcal{C}_i}$
- Partition function (assuming realization is one-to-one):

$$\sum_{\mathbf{s} \in \mathcal{S}} \prod_i \Phi_{\mathcal{C}_i}(\mathbf{a}_i, \mathbf{s}_i) = \Phi_{\mathcal{C}}(\mathbf{a}), \mathbf{a} \in \mathcal{A}$$

Fact: The Fourier transform of the indicator function $\Phi_{\mathcal{C}}$ of a linear code $\mathcal{C}$ is $\Phi_{\mathcal{C}^\perp}$, up to scale.

Application of NFGDT: LDPC codes

Normal factor graph of indicator function $\Phi_{\mathcal{C}}$ of LDPC code $\mathcal{C}$:



$+, =$ represent indicator functions of parity-check, repetition codes
Each little box represents a binary Fourier transform.
Decoding via the right graph: the "tanh rule" of LDPC decoding.

Other applications of generalized holographic transforms

- Generating functions of linear codes on graphs: Forney
- "Holographic" algorithms: Valiant, Cai *et al.*
- Tree reparameterization: Wainwright *et al.*
- Loop calculus: Chertkov and Chernyak
- Lagrange duality, Legendre transforms: Vontobel and Loeliger
- . . .

—Forney and Vontobel, ITA 2011 (arXiv: 1102.0316)

Generating functions of linear codes on graphs

Given a **code indicator function** $\Phi_{\mathcal{C}}(\mathbf{a})$

- **variables** A_k taking values a_k in alphabets $\mathcal{A}_k$
- $\mathbf{a} \in \mathcal{A} = \prod_k \mathcal{A}_k$
- $\Phi_{\mathcal{C}}(\mathbf{a}) = 1$ if $\mathbf{a} \in \mathcal{C}$, otherwise 0

For each A_k, define a set of **indeterminates** $\{z_k(a_k), a_k \in \mathcal{A}_k\}$

- $\mathbf{z}(\mathbf{a}) = \prod_k z_k(a_k)$

(Exact) **generating function**:

$$g_{\mathcal{C}}(\mathbf{z}) = \sum_{\mathbf{a}\in\mathcal{A}} \Phi_{\mathcal{C}}(\mathbf{a})\mathbf{z}(\mathbf{a}) = \sum_{\mathbf{a}\in\mathcal{C}} \mathbf{z}(\mathbf{a})$$

Generating functions of normal partition functions

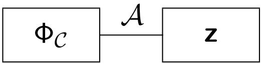
Let $\Phi_{\mathcal{C}}(\mathbf{a})$ be given as a normal partition function:

$$\Phi_{\mathcal{C}}(\mathbf{a}) = \sum_{\mathbf{s}\in\mathcal{S}} \prod_i \Phi_{\mathcal{C}_i}(\mathbf{a}_i, \mathbf{s}_i), \mathbf{a} \in \mathcal{A}$$

Then its generating function is the partition function

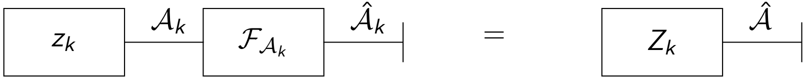
$$g_{\mathcal{C}}(\mathbf{z}) = \sum_{\mathbf{a}\in\mathcal{A}} \mathbf{z}(\mathbf{a}) \sum_{\mathbf{s}\in\mathcal{S}} \prod_i \Phi_{\mathcal{C}_i}(\mathbf{a}_i, \mathbf{s}_i)$$

of the following normal factor graph:

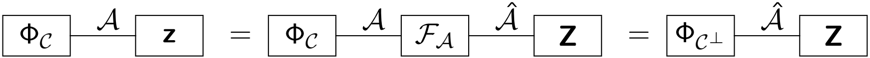


MacWilliams identities

For each A_k, define a dual set of indeterminates $\{Z_k(\hat{a}_k), \hat{a}_k \in \hat{\mathcal{A}}_k\}$ as the Fourier transform of $\{z_k(a_k), a_k \in \mathcal{A}_k\}$



Then the generating function $\Phi_{\mathcal{C}^\perp}(\mathbf{Z})$ of the dual code $\mathcal{C}^\perp$ is the Fourier transform of the generating function $\Phi_{\mathcal{C}}(\mathbf{z})$ (up to scale):



Corollaries: many MacWilliams identities [Forney, 2011]

System theory of normal linear realizations

Normal linear realization:

- Constraints $\mathcal{C}_k$, external variables $\mathcal{A}_i$, internal variables $\mathcal{S}_j$
- Normal degree constraints $\Rightarrow$ normal graph representation
- Generates a linear code $\mathcal{C}$

Normal graph duality theorem:
The dual normal realization generates the orthogonal code $\mathcal{C}^\perp$.

Minimal realization theorem:
A normal realization on a finite connected **cycle-free** graph $\mathcal{G}$ is minimal if and only if every constraint code $\mathcal{C}_i$ is **trim** and **proper**.

Unobservable/uncontrollable $\Rightarrow$ locally reducible:
An unobservable normal realization or its dual uncontrollable realization is locally reducible.

Trimness and properness

A constraint code $\mathcal{C}_i$ is

- **trim** if the projection of $\mathcal{C}_i$ onto any state space $\mathcal{S}_j$ that is involved in $\mathcal{C}_i$ is $\mathcal{S}_j$
- **proper** if $\mathcal{C}_i$ has no nonzero codewords that are supported on a single state variable $\mathcal{S}_j$

Theorem [Gluesing-Luerssen and Williams, 2011]:
A constraint code $\mathcal{C}_i$ is trim if and only if $\mathcal{C}_i^\perp$ is proper.

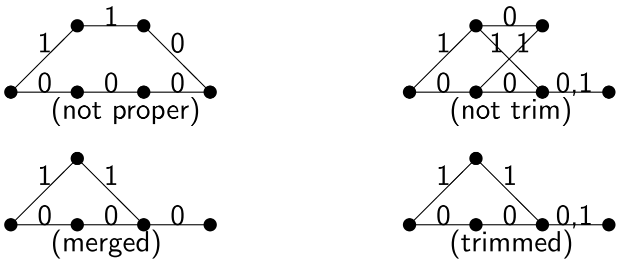
Proof: projection-subcode duality.

Local reducibility

- Obviously if $\mathcal{C}_i$ is not trim, then the state space $\mathcal{S}_j$ can be "trimmed" without changing the code $\mathcal{C}$ that is realized.
- Correspondingly, if $\mathcal{C}_i$ is not proper, then the state space $\mathcal{S}_j$ can be "merged" without changing the code $\mathcal{C}$ that is realized. *Proof*: normal graph duality theorem.

Trimness and properness (cont.)

Example (dual conventional trellis realizations)
$\mathcal{C} = \{000, 110\}$; $\quad \mathcal{C}^{\perp} = \{000, 110, 001, 111\}$



Minimal cycle-free graph realizations

Cycle-free graph $\mathcal{G}$:

- any edge cut partitions $\mathcal{G}$ into two disconnected graphs $\mathcal{P}, \mathcal{F}$

State space theorem [Willems, generalized]:

- In any cycle-free linear realization, minimal state spaces are uniquely determined up to isomorphism ($\Sigma_{\mathcal{P}} \cong \mathcal{C}_{|\mathcal{P}}/\mathcal{C}_{\mathcal{P}}$)

Trim-proper minimal realization theorem:

- A normal realization of a linear code $\mathcal{C}$ on a finite connected cycle-free graph $\mathcal{G}$ is minimal if and only if every constraint code $\mathcal{C}_i$ is trim and proper.

Proof: Necessity is obvious. For sufficiency, prove by induction:
Trim $\Rightarrow$ every state in $\Sigma_{\mathcal{P}}$ is reached by some $\mathbf{a}_{\mathcal{P}} \in \mathcal{C}_{|\mathcal{P}}$;
Proper $\Rightarrow$ every $\mathbf{a}_{\mathcal{P}} \in \mathcal{C}_{|\mathcal{P}}$ reaches a unique state in $\Sigma_{\mathcal{P}}$.
Corollary: Straightforward minimal realization algorithm.

Cycle-free vs. cyclic representations

Cycle-free realizations (*e.g.*, trellis realizations):

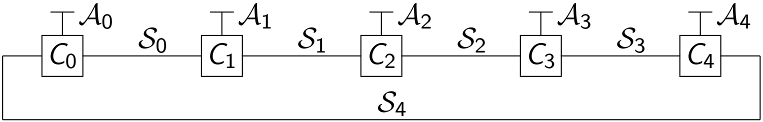
- Given $\mathcal{G}$, minimal realization of a linear code $\mathcal{C}$ is unique and easily computed
- Straightforward exact decoding algorithms (*e.g.*, belief propagation)
- However, the state complexity of any cycle-free realization of a capacity-approaching code is necessarily high

Realizations with cycles (*e.g.*, capacity-approaching codes):

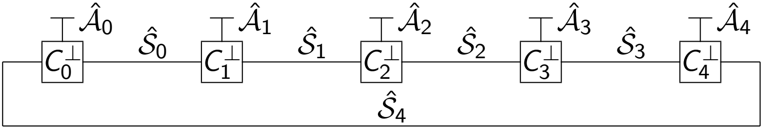
- No unique minimal realizations
- Iterative, approximate decoding algorithms (*e.g.*, belief propagation)
- Feasible decoding complexity (*e.g.*, LDPC codes, turbo codes)

Single-cycle ("tail-biting") trellis realizations

Tail-biting convolutional codes defined on a cyclical time axis:



The dual realization is also a tail-biting trellis realization:



(Sign inverters omitted)

State complexity can be much less than that of conventional trellis

Example: $(24, 12, 8)$ binary Golay code

- Conventional trellis realization: 256 states
- Tail-biting trellis realization: 16 states

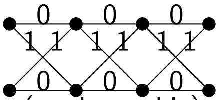
Controllability and observability

A linear realization is:

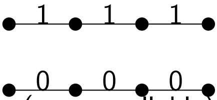
- **observable** if it is *one-to-one*—
  i.e., if the codeword **a** determines the state sequence **s**
- **controllable** if its *constraints are independent*

Theorem: A linear realization is observable if and only if its dual realization is controllable.

Example (dual tail-biting trellis realizations):
($\mathcal{C} = \{000, 011, 101, 110\}$; $\quad \mathcal{C}^{\perp} = \{000, 111\}$)



(unobservable)



(uncontrollable)

Controllability test

Theorem: A linear realization is controllable if and only if

$$\dim \mathfrak{B} = \sum_i \dim \mathcal{C}_i - \sum_j \dim \mathcal{S}_j.$$

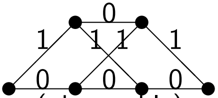
Examples:
(TBT 1): $3 = 6 - 3 \Rightarrow$ controllable.
(TBT 2): $1 \neq 3 - 3 \Rightarrow$ uncontrollable.

Corollary: A linear parity-check realization (*e.g.*, an LDPC code) is controllable if and only if its parity checks are independent.
Proof: count dimensions.

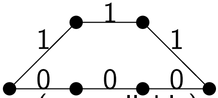
Unobservable/uncontrollable $\Rightarrow$ locally reducible

Theorem: An unobservable linear realization on a finite graph $\mathcal{G}$ with a nonzero trajectory $(\mathbf{0}, \mathbf{s}) \in \mathfrak{B}$ may be locally reduced by trimming any single state space in the support of $\mathbf{s}$.
The dual uncontrollable realization may be correspondingly locally reduced by the dual merging ("pinching") operation.
Proof: construction.

Example (dual tail-biting trellis realizations, cont.):
$(\mathcal{C} = \{000, 011, 101, 110\}; \quad \mathcal{C}^{\perp} = \{000, 111\})$



(observable)



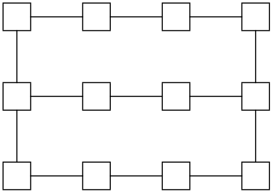
(controllable)

Unobservability and cycles

Theorem: Given an unobservable trim and proper linear realization on a finite graph $\mathcal{G}$ with a nonzero trajectory $(\mathbf{0}, \mathbf{s})$, the support of $(\mathbf{0}, \mathbf{s})$ must be a cycle or generalized cycle $\mathcal{G}' \subseteq \mathcal{G}$.

Cycle: a finite connected graph with vertex degrees $= 2$.
Generalized cycle: same, with vertex degrees ≥ 2 (a "2-core").

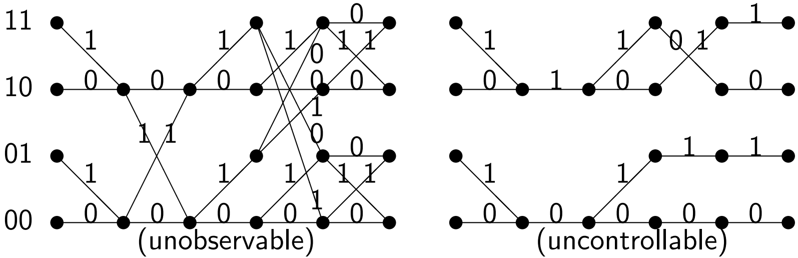
Example of a generalized cycle:



Unobservability and cycles (cont.)

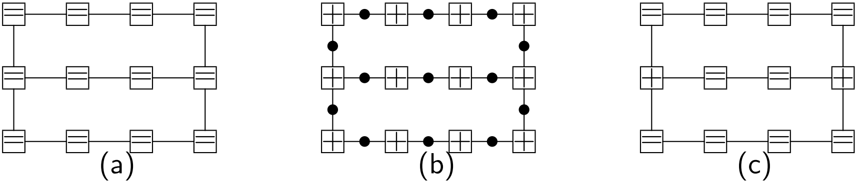
Theorem (cont.): On an unobservable trajectory $(\mathbf{0}, \mathbf{s})$ with support $\mathcal{G}'$, all first state coordinates may be taken to be equal. In the dual uncontrollable realization, the corresponding global constraint on $\mathcal{G}'$ partitions $\mathfrak{B}$ into disconnected cosets.

Example (dual tail-biting trellis realizations):
$\mathcal{C} = \langle 01110, 10010, 01101 \rangle; \quad \mathcal{C}^{\perp} = \langle 10111, 01100 \rangle$



Unobservability and cycles (cont.)

Theorem (cont.): A primal repetition realization on $\mathcal{G}'$ determines the possible values of the first state coordinates in the subspace of $\mathfrak{B}$ generated by $(\mathbf{0}, \mathbf{s})$. The dual zero-sum realization on $\mathcal{G}'$ determines the constraints on all possible trajectories of the first dual state coordinates in the dual uncontrollable realization.



(a) repetition realization defined on a generalized cycle $\mathcal{G}'$;
(b) dual zero-sum realization ($\bullet$ = inverter); (c) equivalent dual.