

CNS 188a Overview

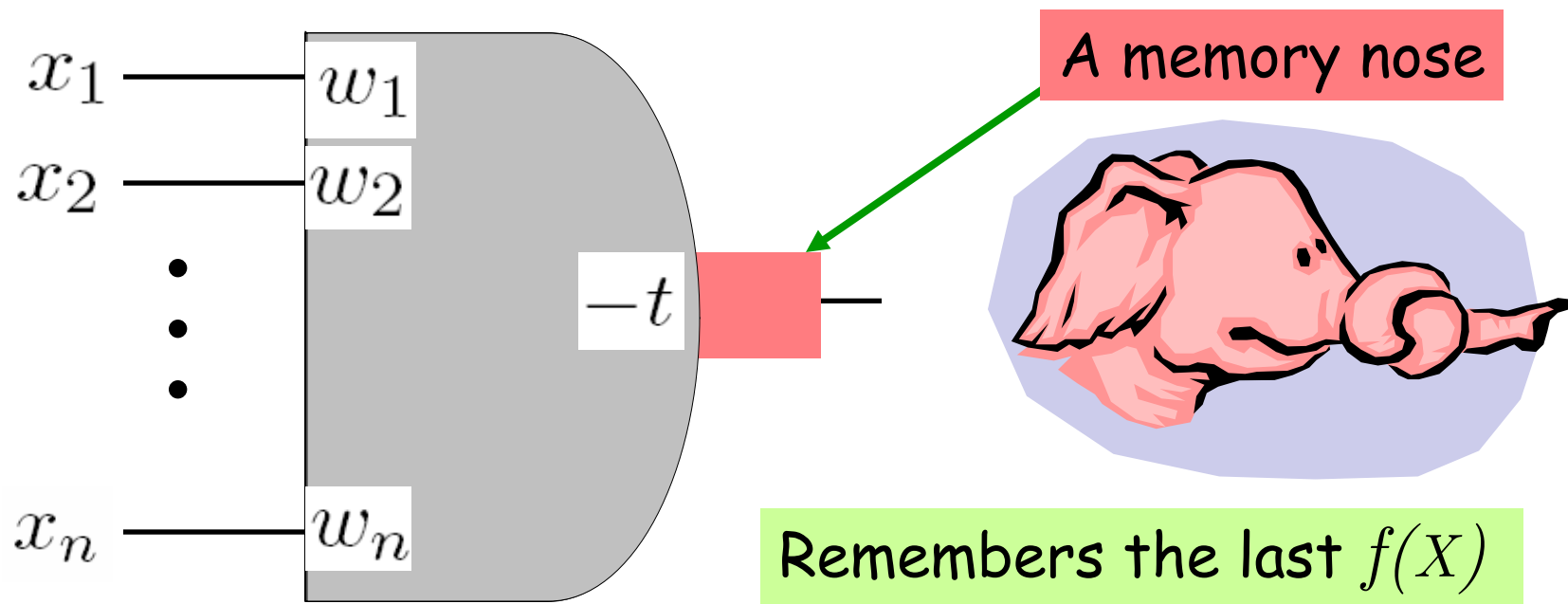
- Boolean algebra as an axiomatic system
- Boolean functions and their representations using Boolean formulas and spectral methods
- Implementing Boolean functions with relay circuits, circuits of AON (AND, OR, NOT) gates and LT (Linear Threshold) gates
- Analyzing the complexity (size and depth) of circuits
- Relations (as opposed to functions) and their implementation in circuits
- Feedback and convergence in LT circuits

Linear Threshold

New Ingredient

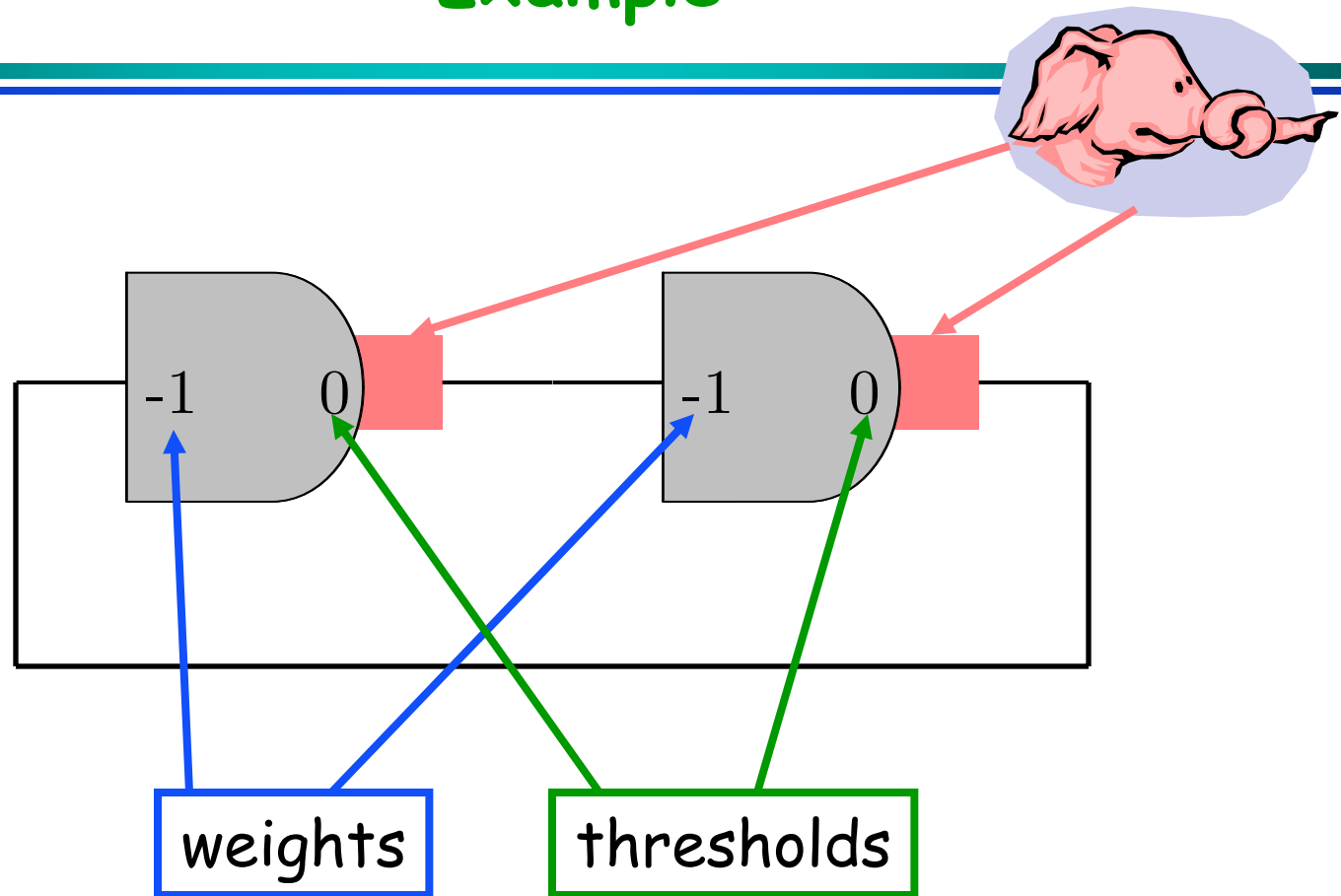
$$f(X) = \text{sgn}(F(X))$$

$$F(X) = -t + \sum_{i=1}^n w_i x_i$$



Feedback Networks

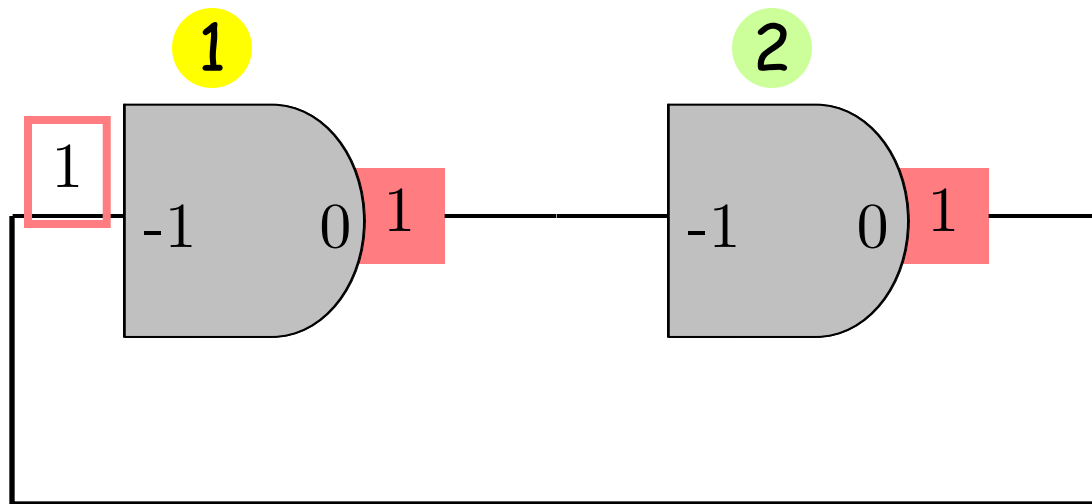
Example



The state of the network: the vector that corresponds to the states (noses...) of the gates

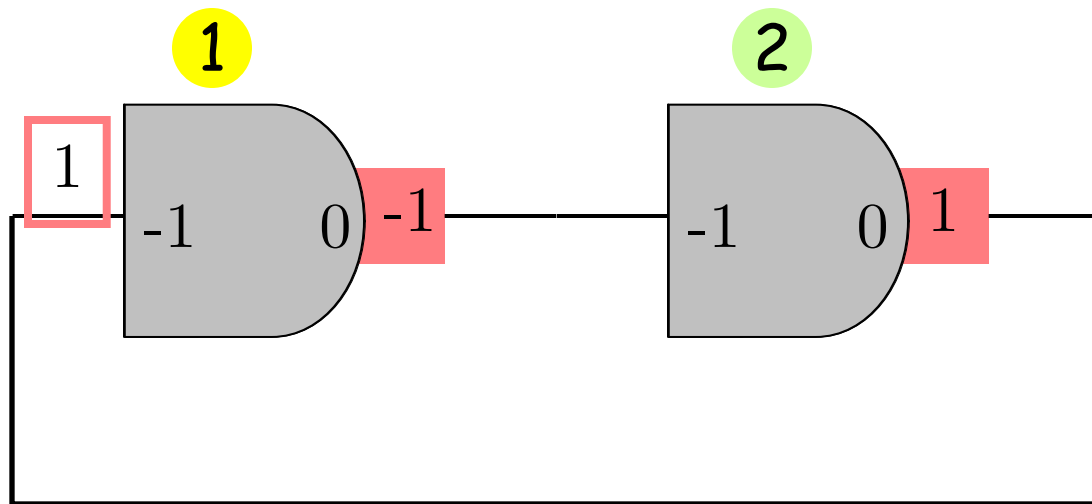
Feedback Networks

Example



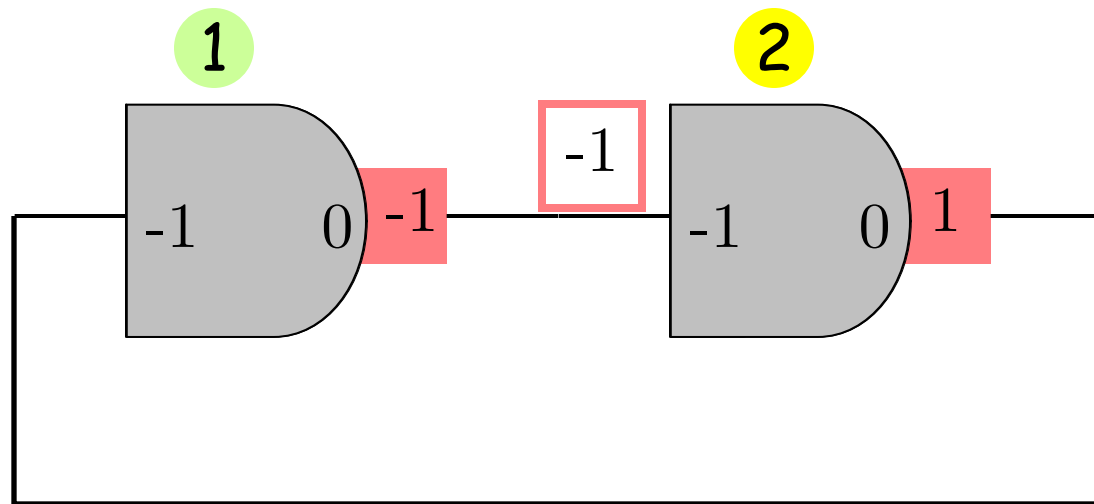
Feedback Networks

Example



Feedback Networks

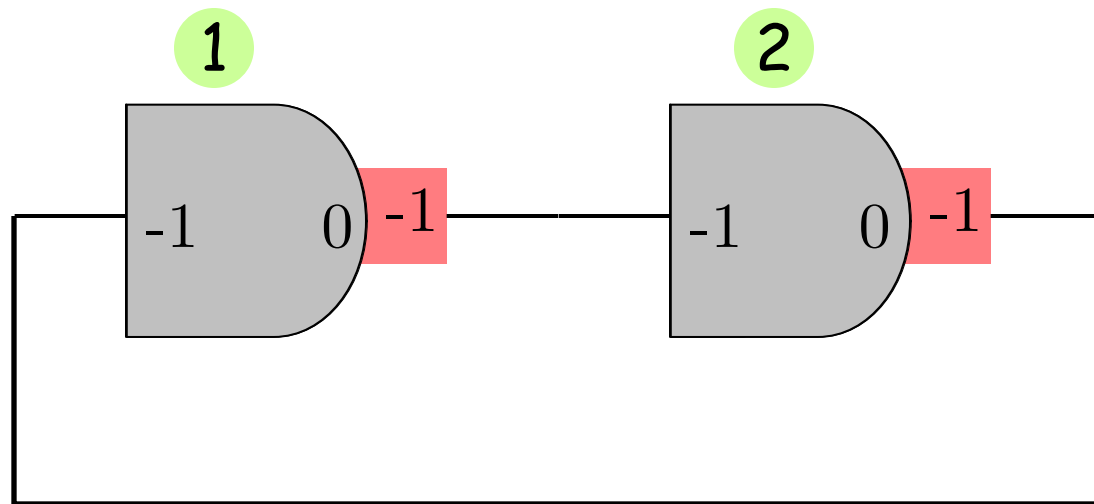
Example



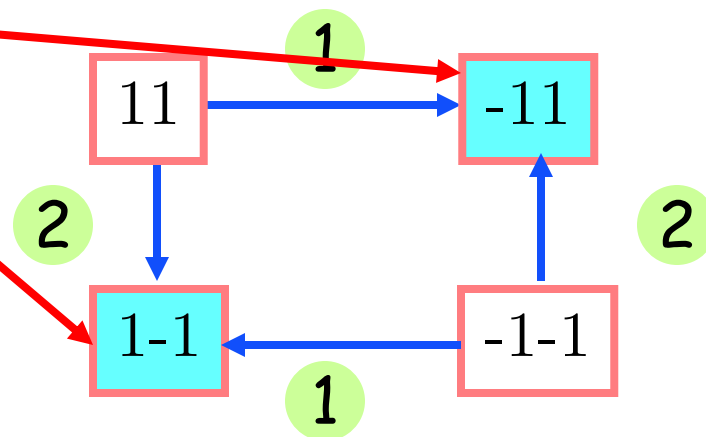
-11 is a stable state

Feedback Networks

Example



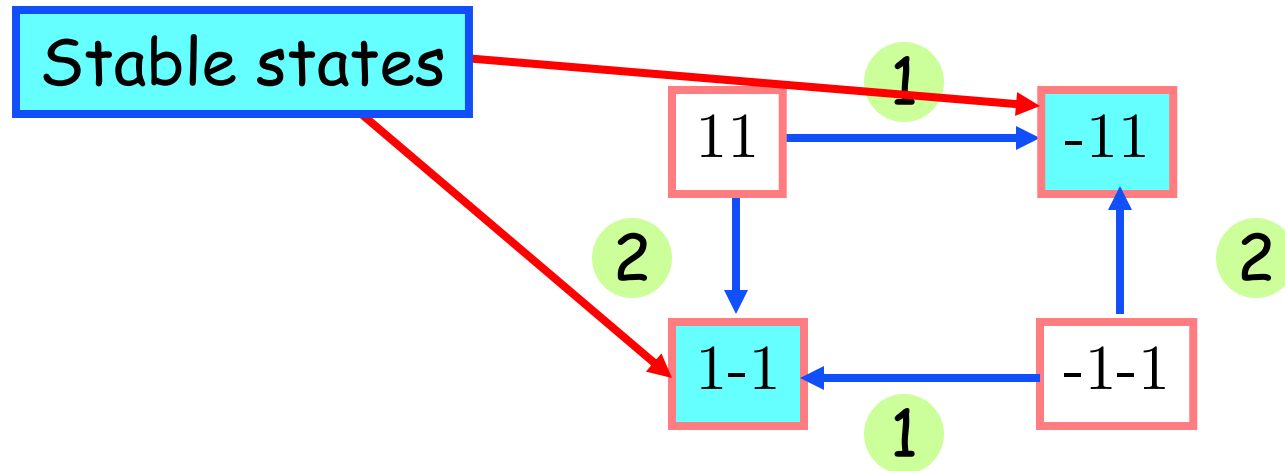
Stable states



Q:
why care about
stable states?

Feedback Networks

Computing with Dynamics



Input:
initial state

11

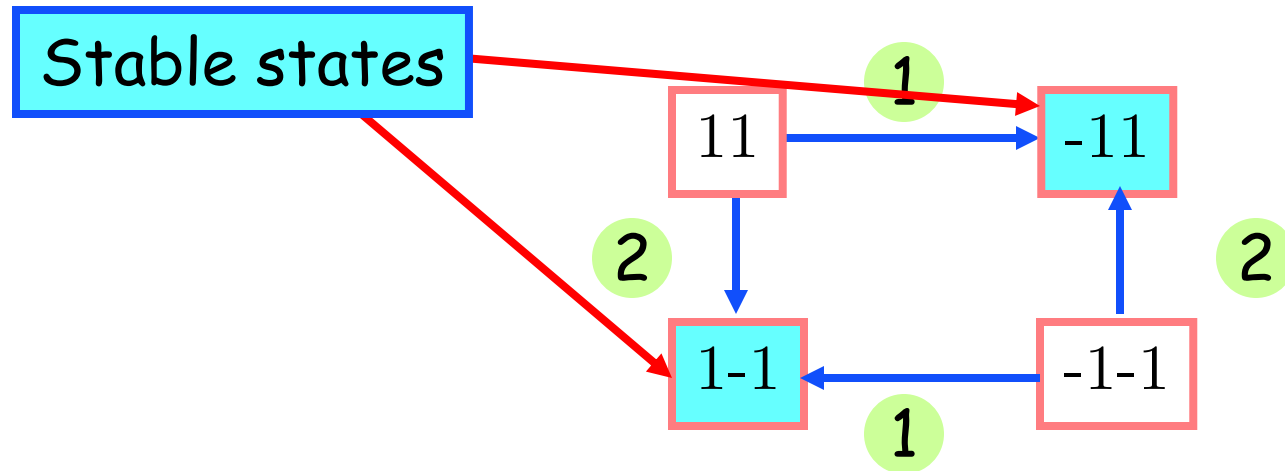
Feedback
Network

Output:
stable state

-11

Feedback Networks

Computing with Dynamics



Input:
initial state

11

Feedback
Network

Output:
stable state

1-1

Feedback Networks

Computing with Dynamics

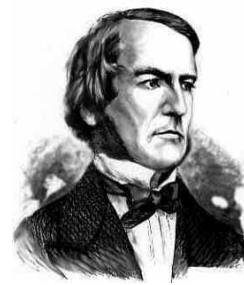
Associative Memory
"The Leibniz-Boole Machine"

Input:
initial state

Output:
stable state



Feedback
Network



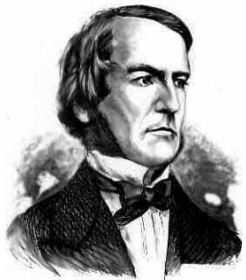
Feedback Networks

Computing with Dynamics

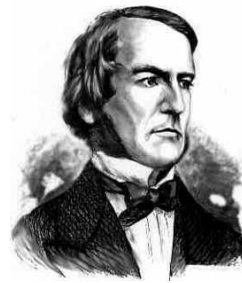
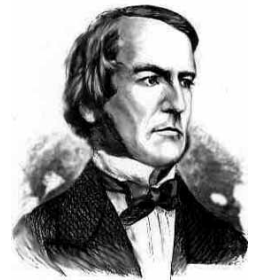
Associative Memory
"The Leibniz-Boole Machine"

Input:
initial state

Output:
stable state



Feedback
Network



Feedback Networks

Computing with Dynamics

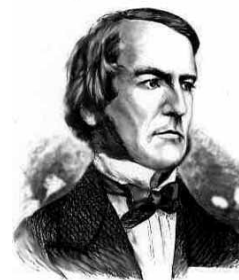
Associative Memory
"The Leibniz-Boole Machine"

Input:
initial state

Output:
stable state



Feedback
Network



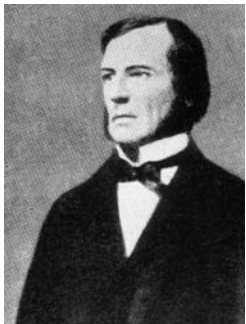
Feedback Networks

Computing with Dynamics

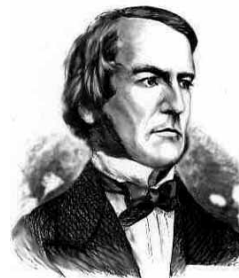
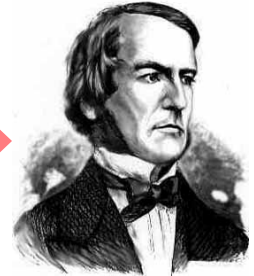
Associative Memory
"The Leibniz-Boole Machine"

Input:
initial state

Output:
stable state



Feedback
Network



Feedback Networks

Computing with Dynamics

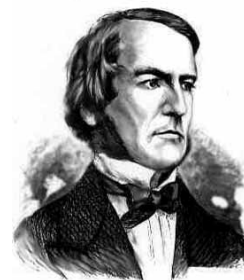
Associative Memory
"The Leibniz-Boole Machine"

Input:
initial state

Output:
stable state



Feedback
Network



Feedback Networks

Computing with Dynamics

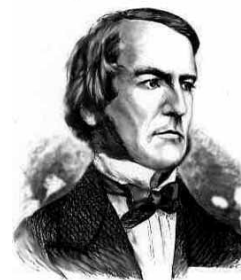
Associative Memory
"The Leibniz-Boole Machine"

Input:
initial state

Output:
stable state



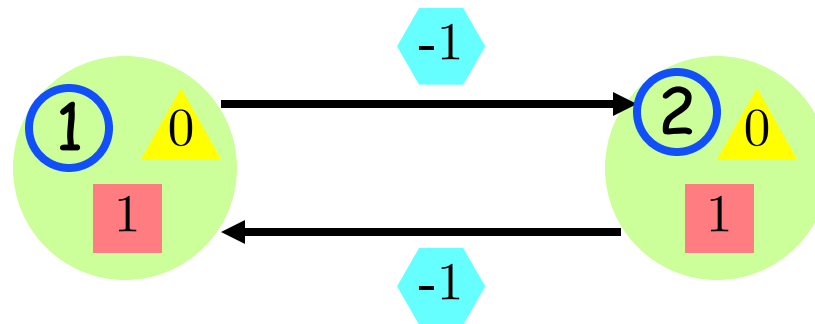
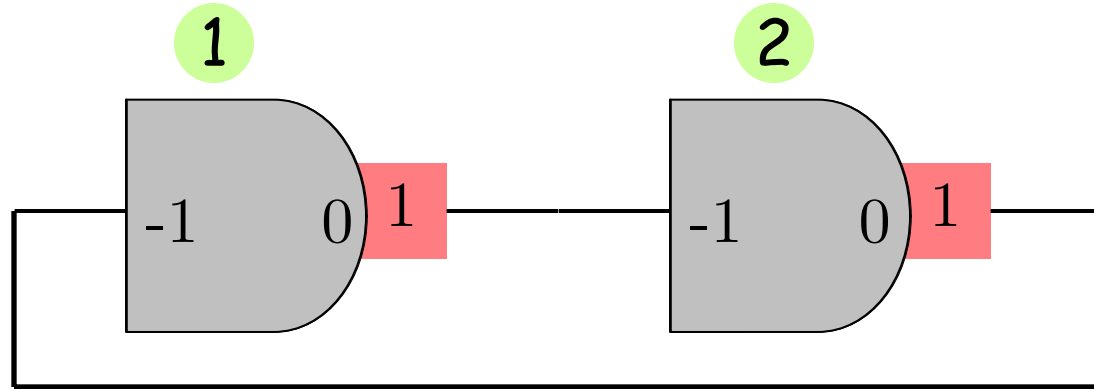
Feedback
Network





Feedback Networks

Hopfield Model (Caltech 1982)



$\textcircled{i}$ = node i

$\triangle 0$ = threshold t_i

$\textcolor{red}{1}$ = state v_i

$\textcolor{cyan}{-1}$ = weight of edge (i,j)

Feedback Networks

The Vector/Matrix Description

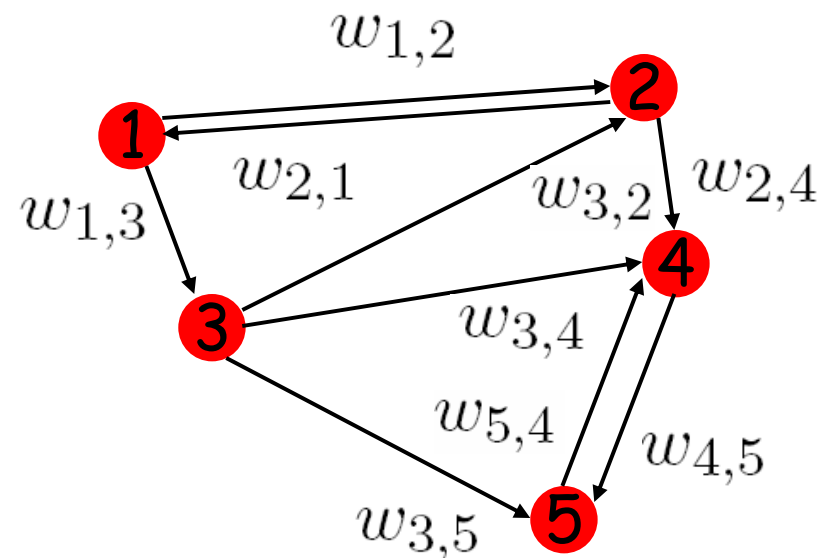
An n node feedback network can be specified by:

- W an $n \times n$ matrix of weights
- T an n vector of thresholds
- V an n vector of states

$$N = (W, T)$$

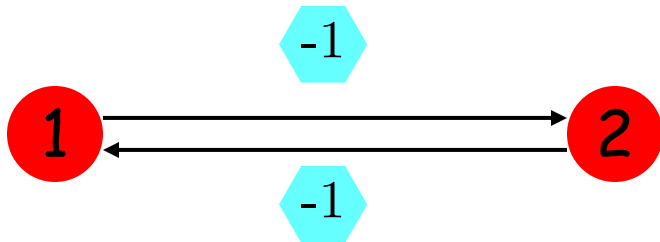
$$V = \begin{bmatrix} v_1 \\ v_2 \\ \cdot \\ \cdot \\ v_n \end{bmatrix}$$

$$T = \begin{bmatrix} t_1 \\ t_2 \\ \cdot \\ \cdot \\ t_n \end{bmatrix}$$



The Matrix Description

Example



$$W = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$

$$T = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Computation in $N = (W, T)$

$$F(t) = W^T \cdot V - T$$

$$\begin{bmatrix} F_1 \\ F_2 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

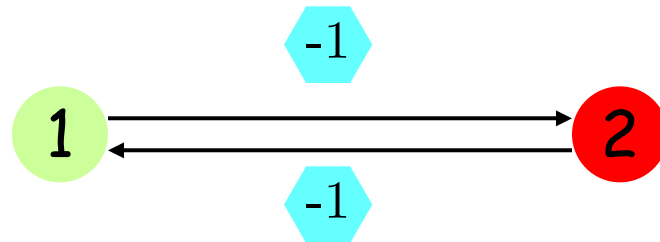
$$= \begin{bmatrix} -v_2 \\ -v_1 \end{bmatrix}$$

Q: when do the nodes compute?

Modes of Operation

Q: when do the nodes compute?

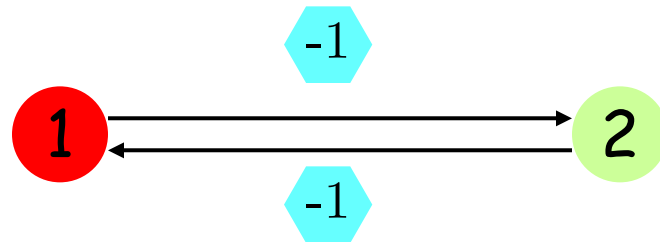
Serial mode: one node at a time (arbitrary order)



Modes of Operation

Q: when do the nodes compute?

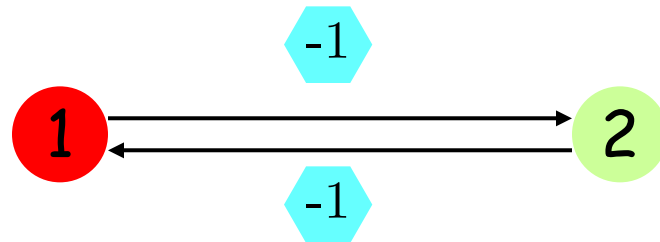
Serial mode: one node at a time (arbitrary order)



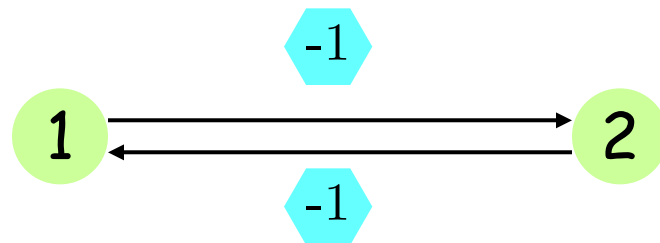
Modes of Operation

Q: when do the nodes compute?

Serial mode: one node at a time (arbitrary order)

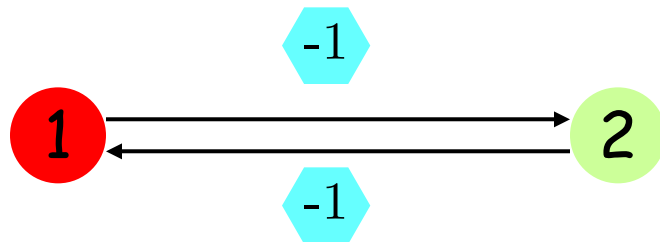


Fully-Parallel mode: all nodes at the same time



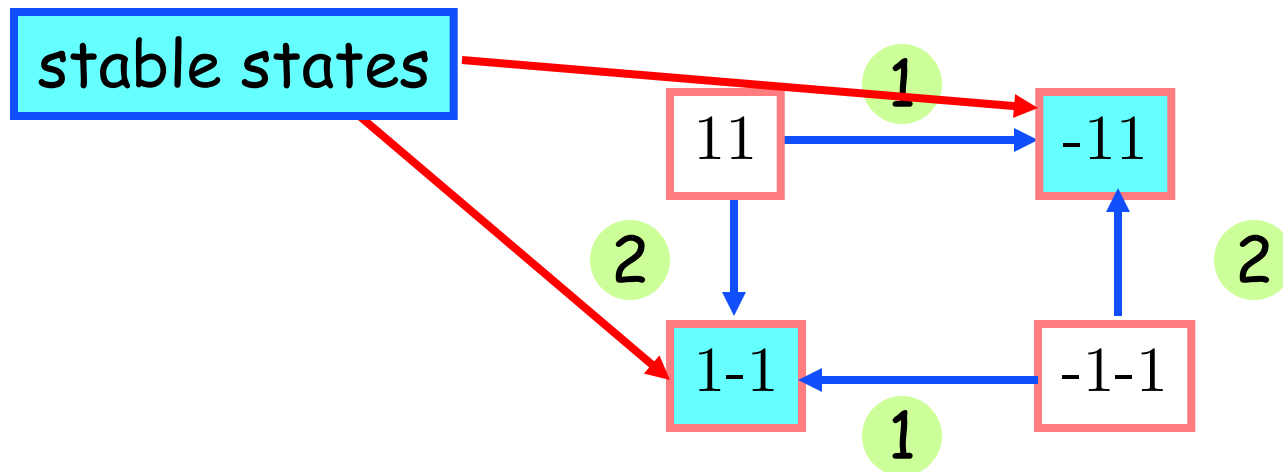
Example 1

Serial Mode - Symmetric Weight Matrix



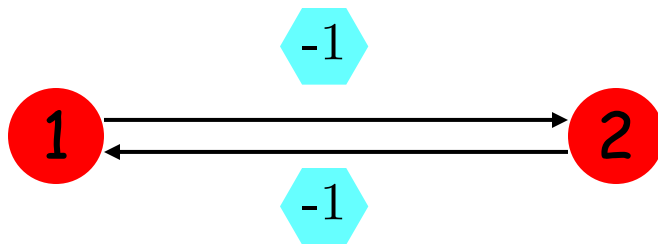
$$W = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$
$$T = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

The state space:



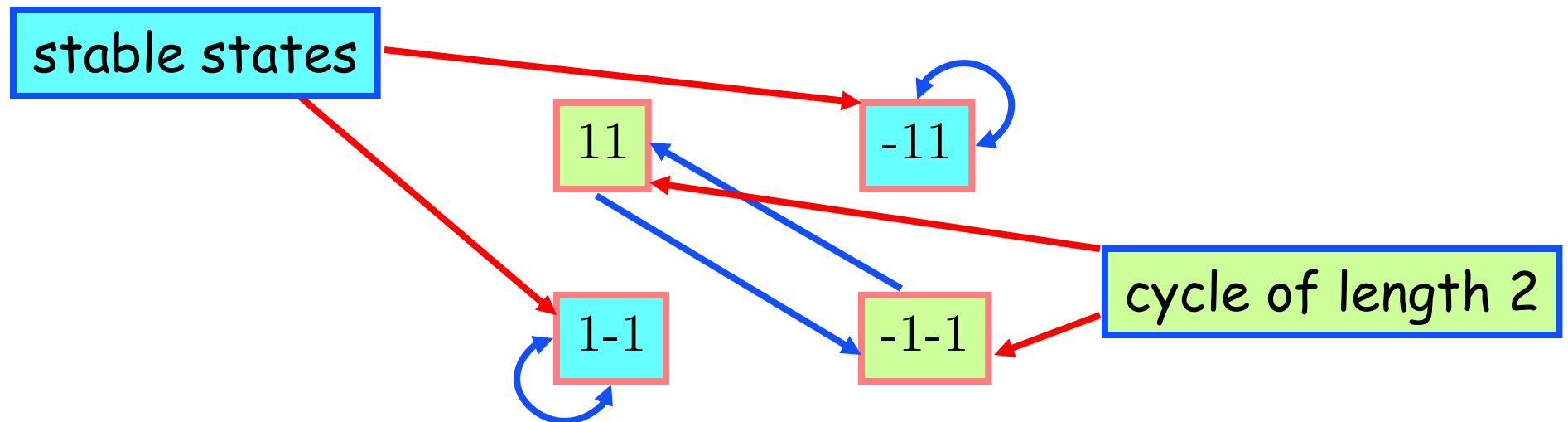
Example 2

Fully-Parallel Mode - Symmetric Weight Matrix



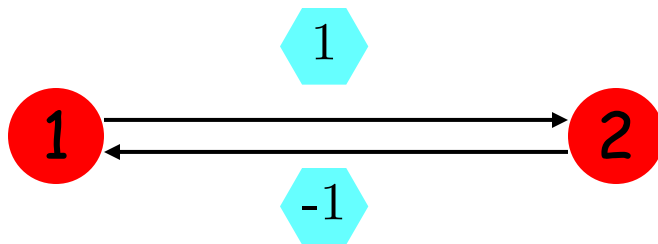
$$W = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$
$$T = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

The state space:



Example 3

Fully-Parallel Mode - Antisymmetric Weight Matrix



$$T = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad W = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

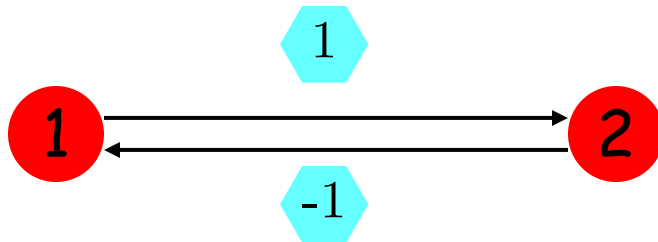
Q: how does the state space look?

$$\begin{aligned} \text{sgn} \left(\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right) &= \begin{bmatrix} -1 \\ 1 \end{bmatrix} \\ \text{sgn} \left(\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right) &= \begin{bmatrix} -1 \\ -1 \end{bmatrix} \\ \text{sgn} \left(\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \end{bmatrix} \right) &= \begin{bmatrix} 1 \\ -1 \end{bmatrix} \\ \text{sgn} \left(\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right) &= \begin{bmatrix} 1 \\ 1 \end{bmatrix} \end{aligned}$$

cycle of length 4

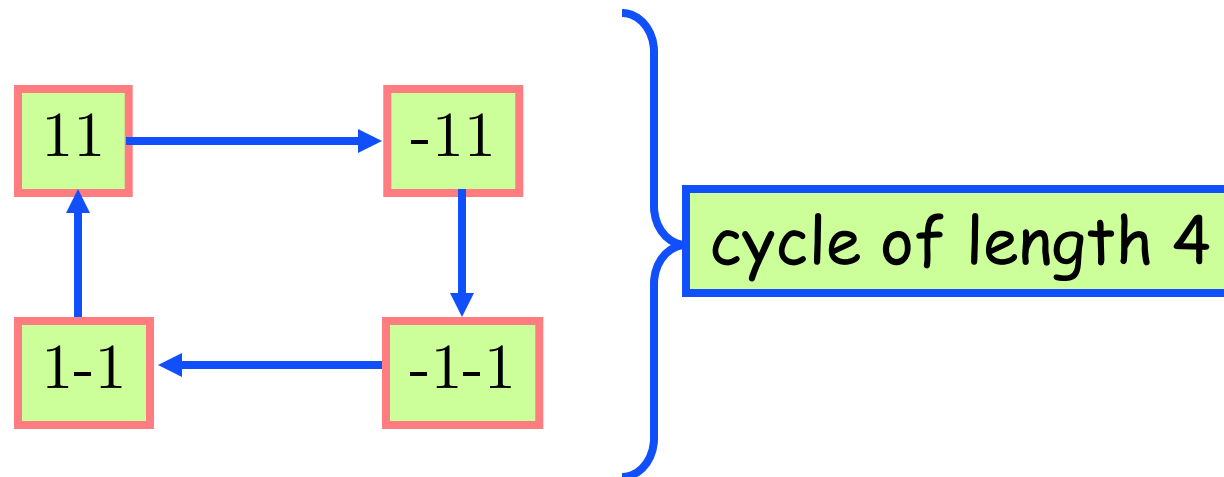
Example 3

Fully-Parallel Mode - Antisymmetric Weight Matrix

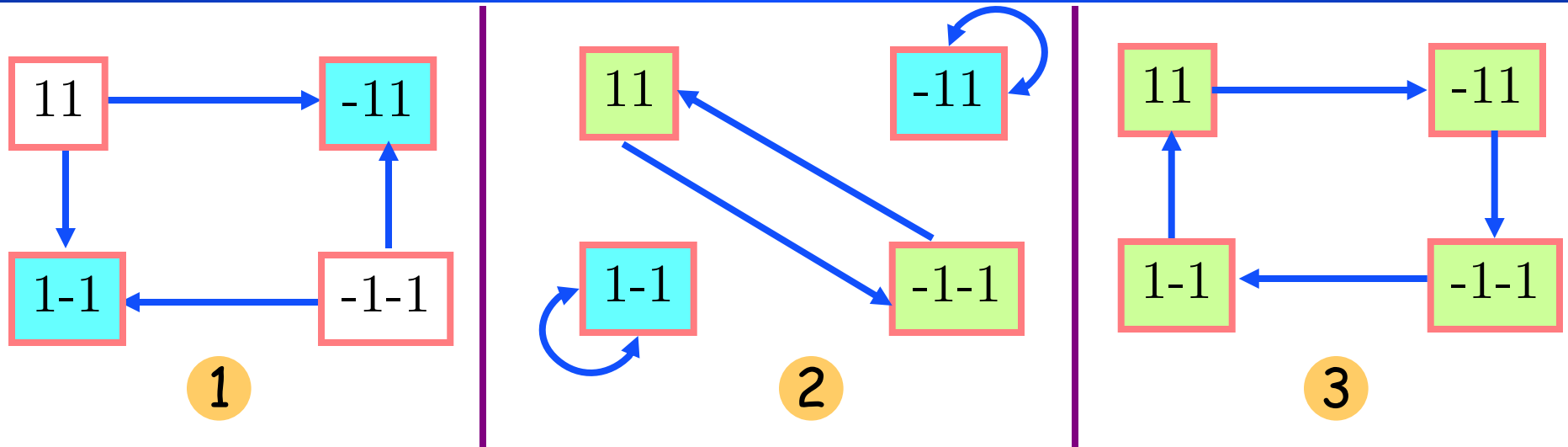


$$T = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad W = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

The state space:



The Three Cases



Cycle lengths

Example #

mode \ W	symmetric	antisymmetric
serial	1	?
fully-parallel	1,2	4

The Three Cases

It Is True in General!
It Is True in General!

Cycle lengths

Example #

mode \ W	symmetric	antisymmetric
serial	1	?
fully-parallel	2	3

1 → Hopfield 1982

2 → Goles 1985

3 → Goles 1986

Questions on Convergence

Cycle lengths	mode \ W	symmetric	antisymmetric
	serial	1	?
Example #	fully-parallel	2	3

1 → Hopfield 1982

2 → Goles 1985

3 → Goles 1986

Q1: Are the three cases "distinct"?

Q2: Elementary proof? (wo/energy)

Q3: Other "interesting" cases?

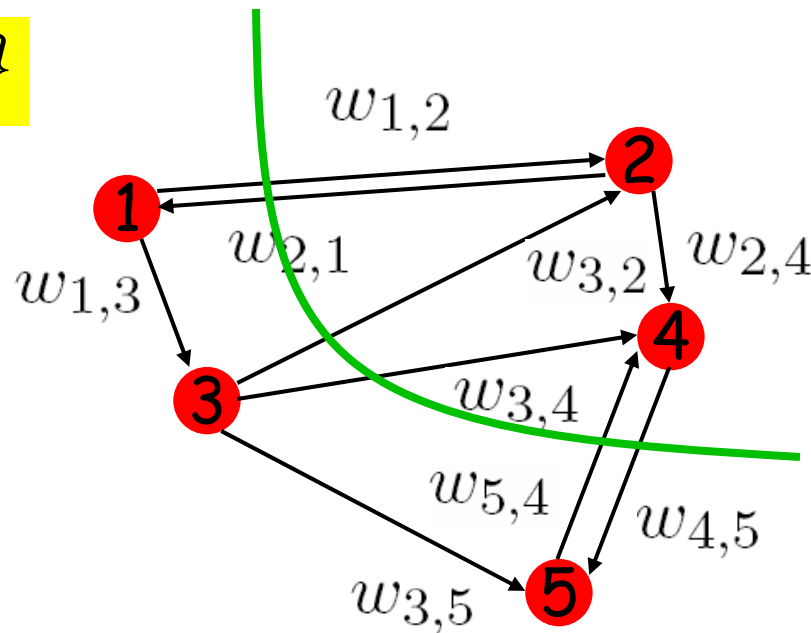
NO YES

Cuts in Graphs

Partition the set of nodes to two sets: V_1 and V_{-1}

$$V_1 = \{1, 3, 5\}$$

$$V_{-1} = \{2, 4\}$$

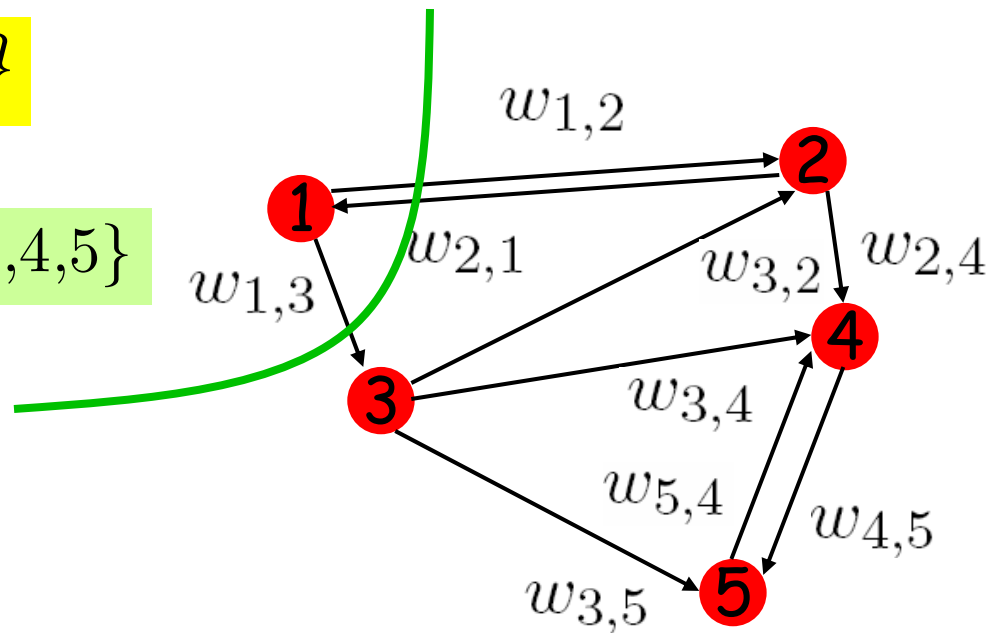


Cuts in Graphs

Partition the set of nodes to two set: V_1 and V_{-1}

$$V_1 = \{1\}$$

$$V_{-1} = \{2,3,4,5\}$$

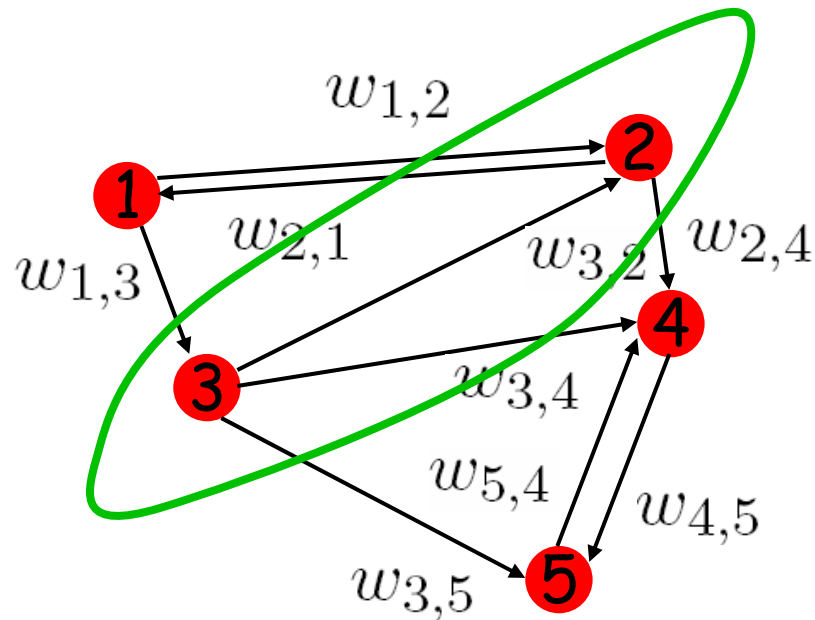


Cuts in Graphs

Partition the set of nodes to two set: V_1 and V_{-1}

$$V_1 = \{1, 4, 5\}$$

$$V_{-1} = \{2, 3\}$$

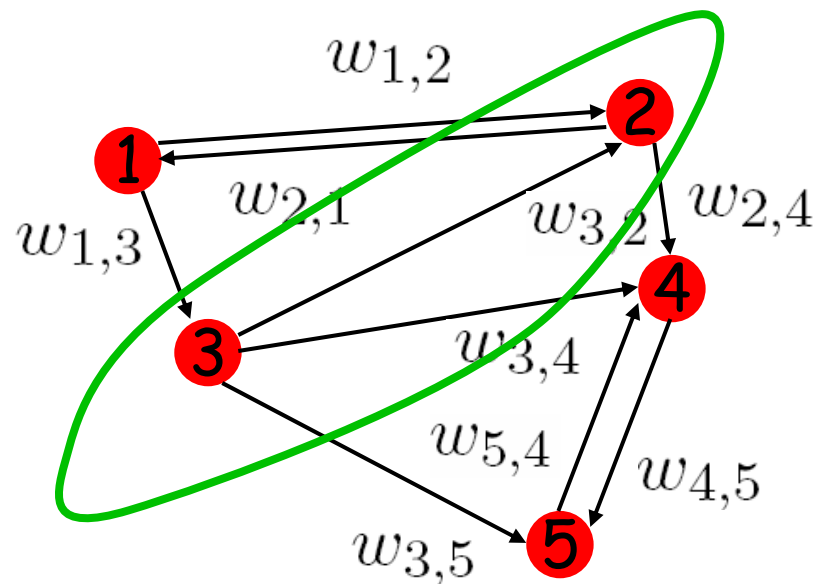


Cuts in Graphs

Partition the set of nodes to two set: V_1 and V_{-1}

$$V_1 = \{1, 4, 5\}$$

$$V_{-1} = \{2, 3\}$$



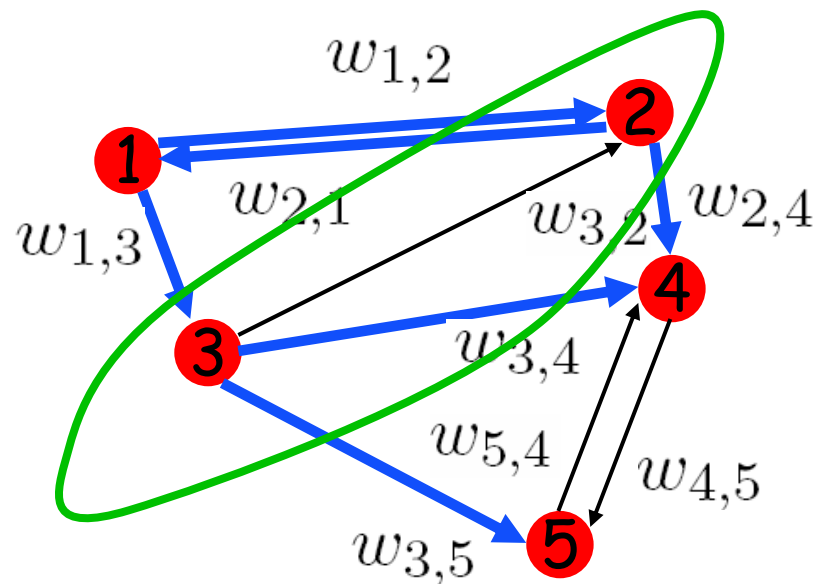
CUT = The set of edges incident at one node in V_1 and one node in V_{-1}

Cuts in Graphs

Partition the set of nodes to two set: V_1 and V_{-1}

$$V_1 = \{1, 4, 5\}$$

$$V_{-1} = \{2, 3\}$$



CUT = The set of edges incident at one node in V_1 and one node in V_{-1}

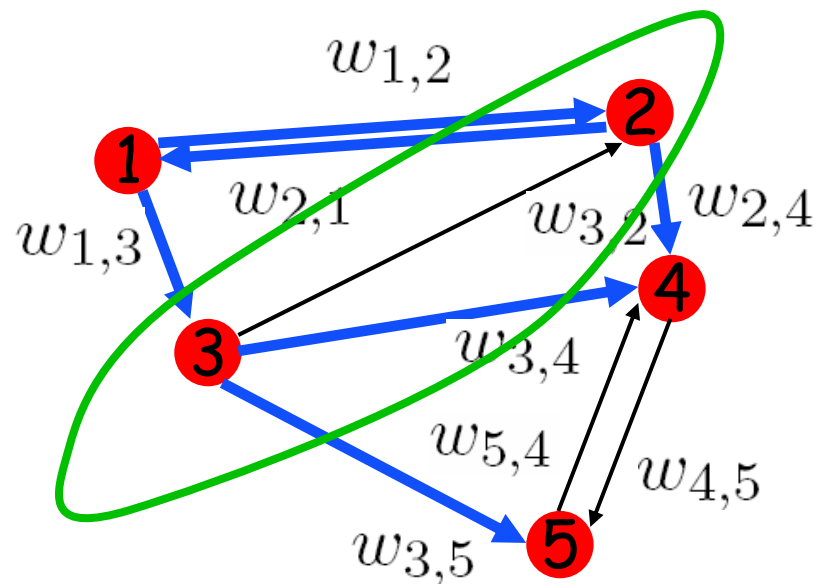
Cuts in Graphs

Partition the set of nodes to two set: V_1 and V_{-1}

$$V_1 = \{1, 4, 5\}$$

$$V_{-1} = \{2, 3\}$$

Weight of a cut:
sum of edge weights
in a cut



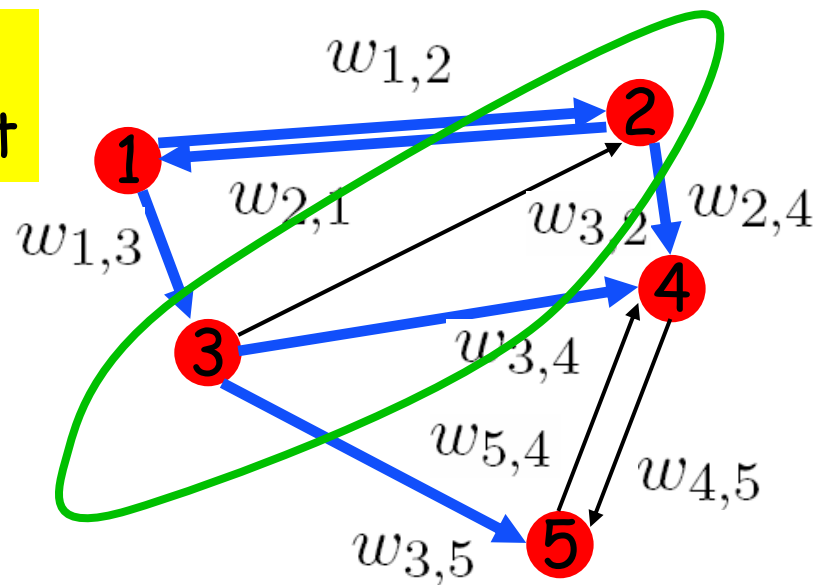
CUT = The set of edges incident at one node in V_1 and one node in V_{-1}

Cuts in Graphs

Partition the set of nodes to two set: V_1 and V_{-1}

Minimal Cut (local):
a cut with minimal weight

Weight of a cut:
sum of edge weights
in a cut



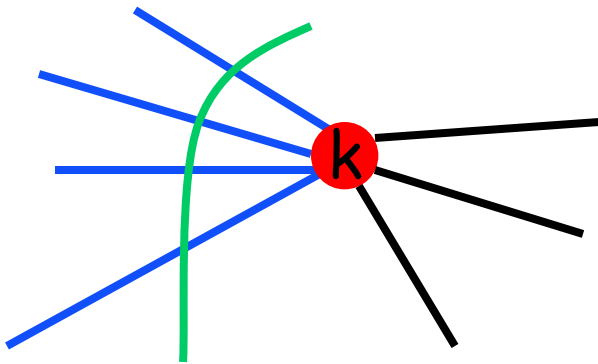
CUT = The set of edges incident at one node in V_1 and one node in V_{-1}

Algorithm (called L) for finding a Locally Minimal Cut in a Graph

(1) Given a graph, start with a random cut (partition)

(2) Choose a node k at random

(3) Compare the sum of weights of the edges that:
(i) belong to the cut and incident at node k
(ii) the other edges incident at node k



(4) If the blue set is lighter or equal to the black set: leave node k in its location

Otherwise: move node k to the other side of the cut

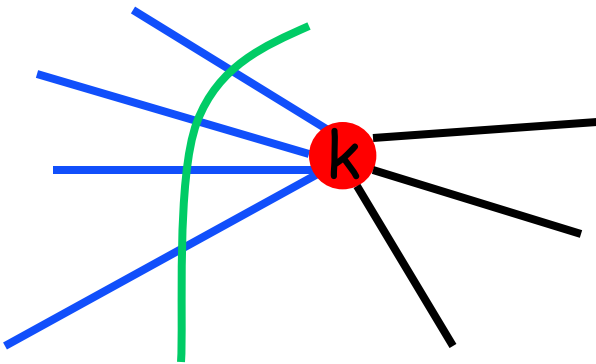
(5) Goto step (2)

Q: Does Algorithm L converge to a locally minimal cut?

(1) Given a graph, start with a random cut (partition)

(2) Choose a node k at random

(3) Compare the sum of weights of the edges that:
(i) belong to the cut and incident at node k
(ii) the other edges incident at node k



(4) If the blue set is lighter or equal to the black set: leave node k in its location

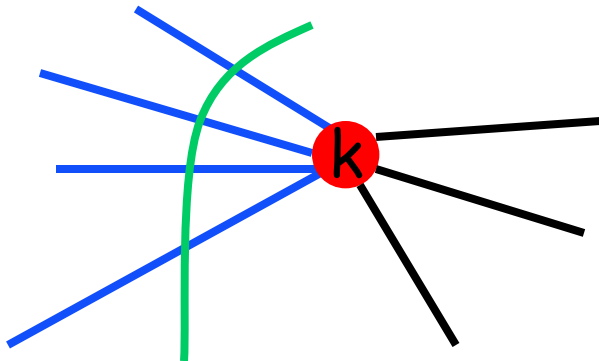
Otherwise: move node k to the other side of the cut

(5) Goto step (2)

Q: Does Algorithm L converge to a locally minimal cut?

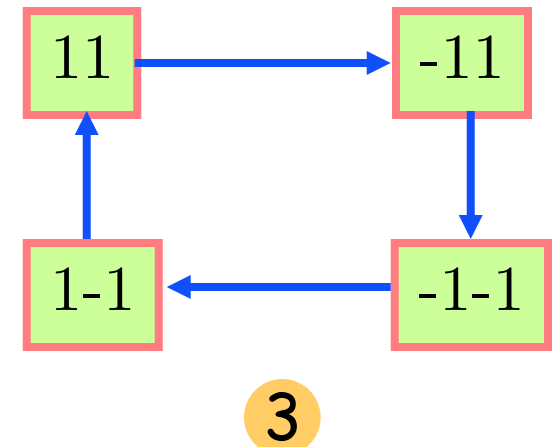
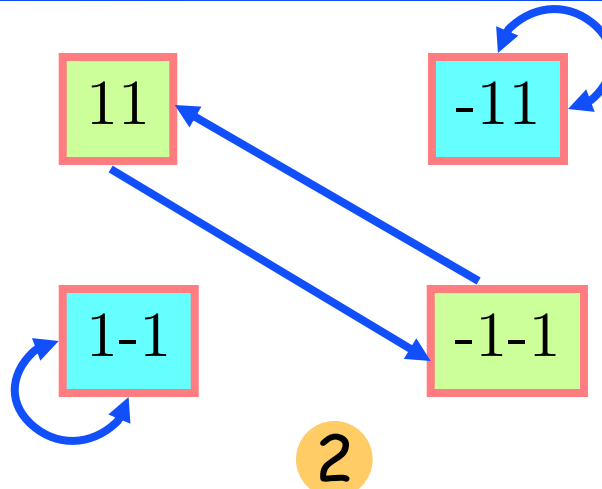
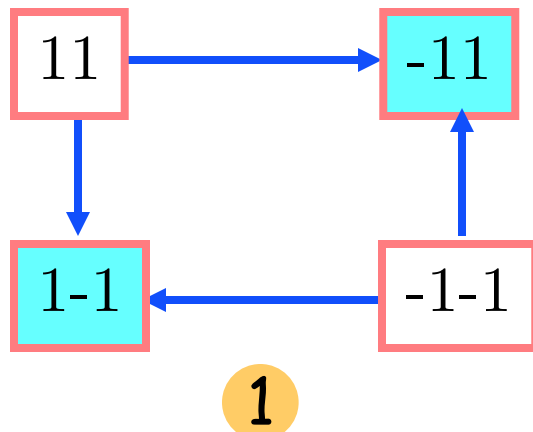
(4) If the blue set is lighter or equal to the black set: **leave** node k in its location

Otherwise: **move** node k to the other side of the cut



Algorithm L converges to a cut that cannot be improved by moving a single node from V_1 to V_{-1}

The Three Cases



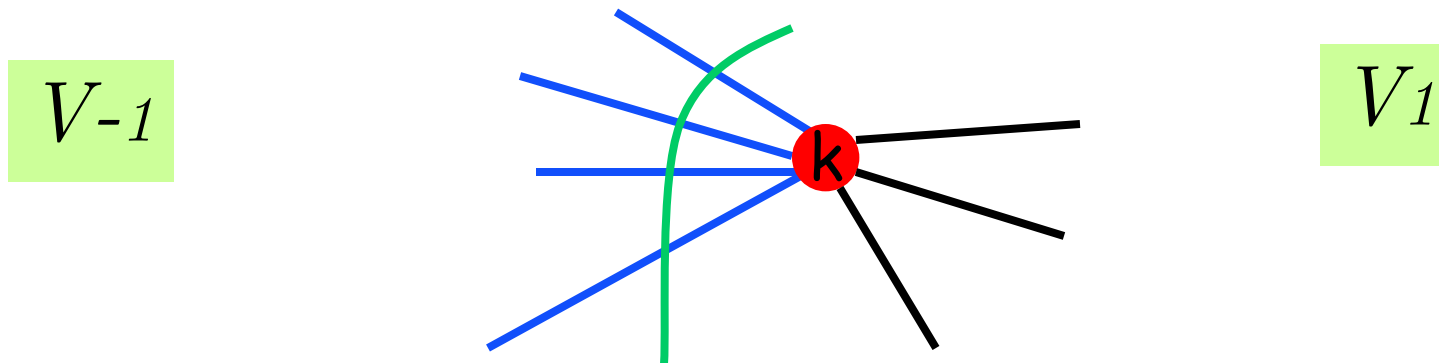
Cycle lengths

Prove that this case follows from convergence of Algorithm L

	W	symmetric	antisymmetric
mode			
serial	1	1	?
fully-parallel	2	1,2	3 4

(4) If the blue set lighter or equal to the black set: **leave** node k in its location

Otherwise: **move** node k to the other side of the cut



$$\text{Sgn}(\text{black} - \text{blue}) = 1$$

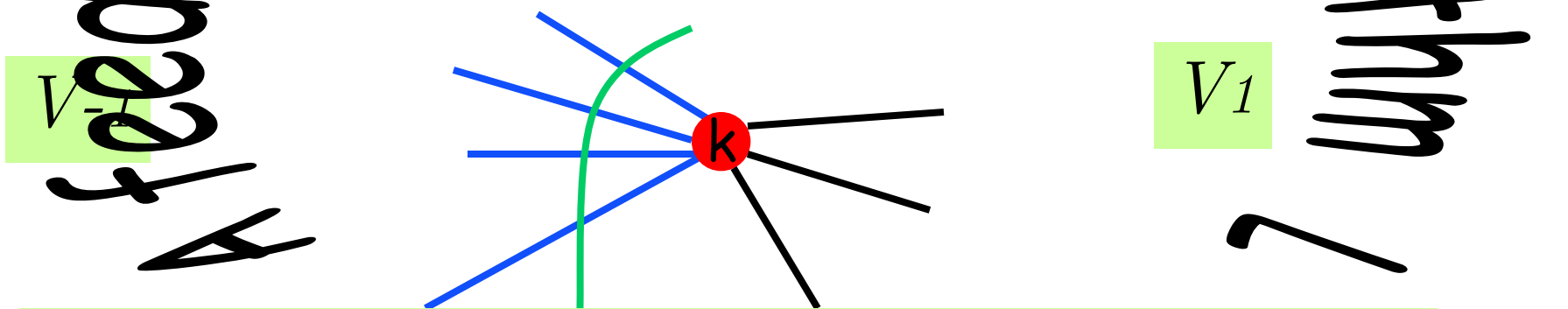
Leave k in the set V_1

$$\text{Sgn}(\text{black} - \text{blue}) = -1$$

Move k to the set V_{-1}

(4) If the blue set lighter or equal to the black set: leave node k in its location

Otherwise: move node k to the other side of the cut



Hence, if Algorithm L converges so does the network

$$\text{Sgn}(\text{black} - \text{blue}) = 1$$

Leave k in the set V_1

$$\text{Sgn}(\text{black} - \text{blue}) = -1$$

Move k to the set V_{-1}

This is exactly an LT gate operation in node k

Questions on Convergence

Cycle lengths	mode \ W	symmetric	antisymmetric
	serial	1	?
Example #	fully-parallel	1,2	4

1 → Hopfield 1982

2 → Goles 1985

3 → Goles 1986

Q1: Are the three cases "distinct"?

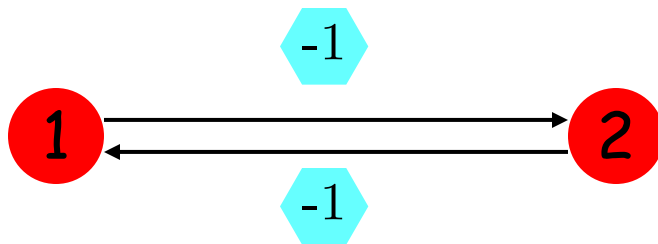
Q2: Elementary proof? (wo/energy)

Q3: Other "interesting" cases?

NO
YES
es
s

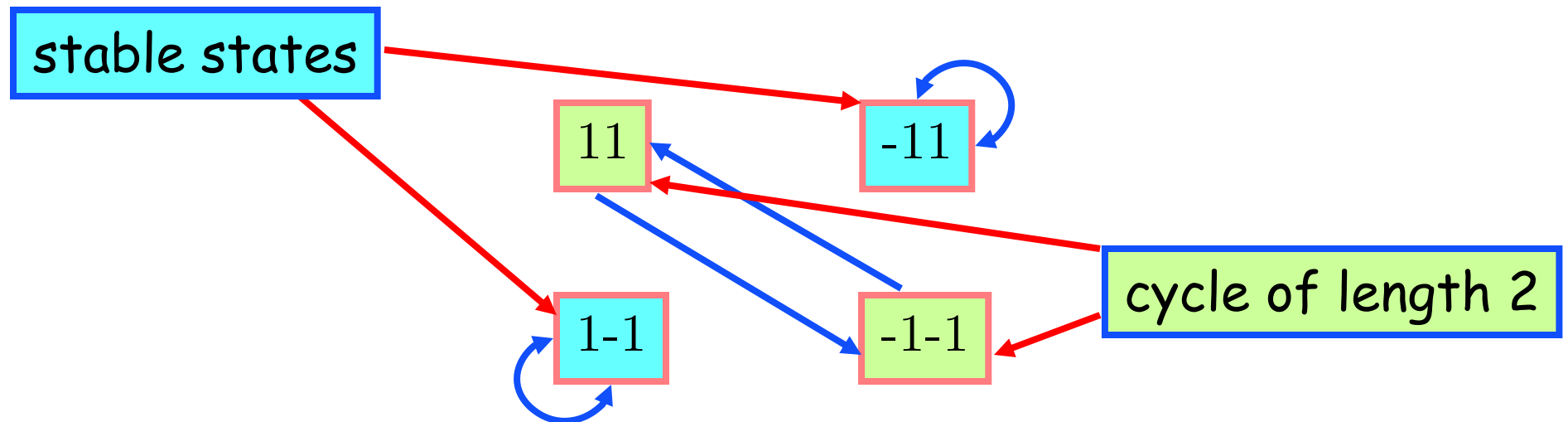
Example 2

Fully-Parallel Mode - Symmetric Weight Matrix



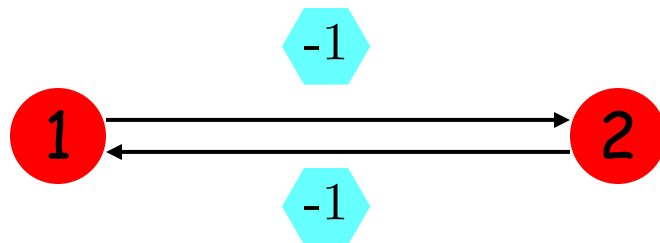
$$W = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$
$$T = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

The state space:



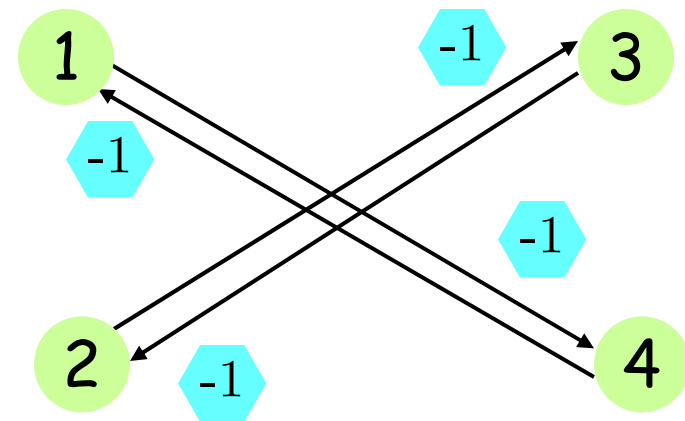
Idea

Simulate a Fully-Parallel Mode with Serial Mode



$$W = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$

$$T = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

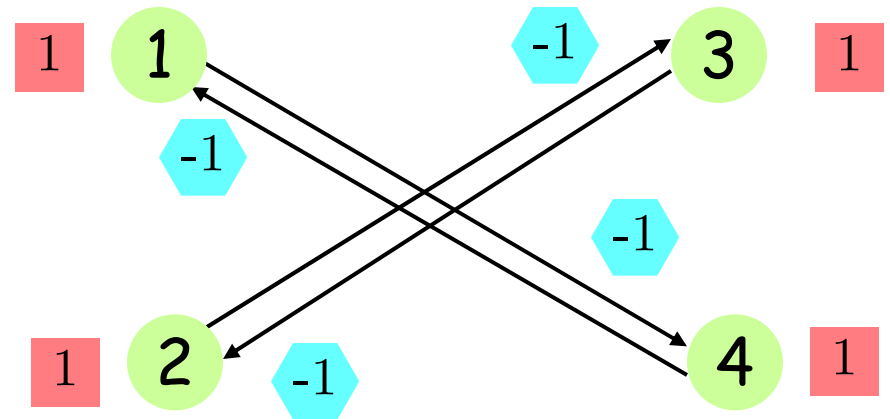
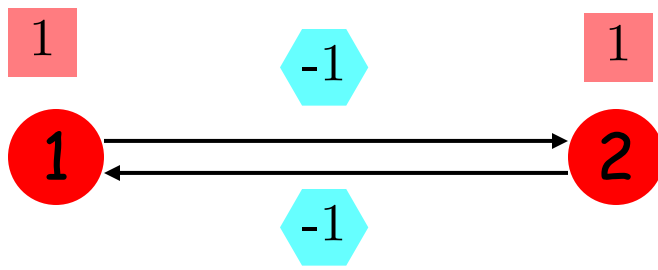


$$W = \begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix}$$

$$T = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

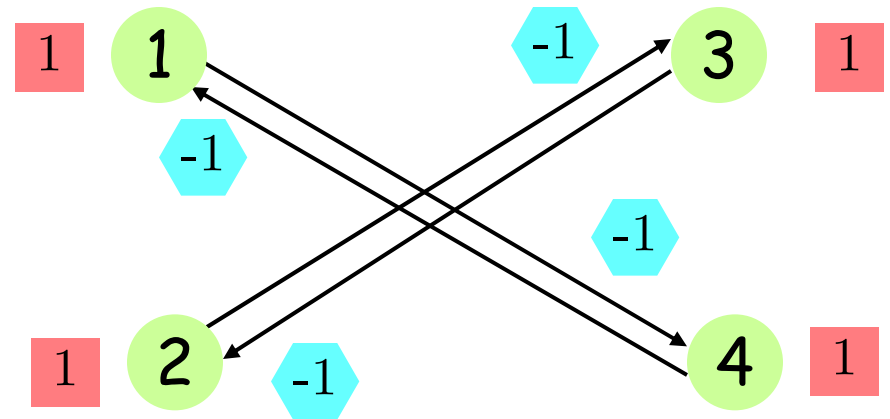
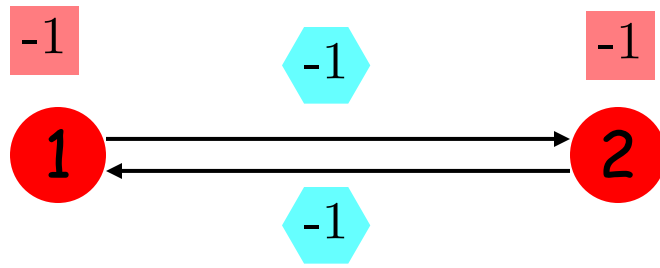
Idea

Simulate a Fully-Parallel Mode with Serial Mode



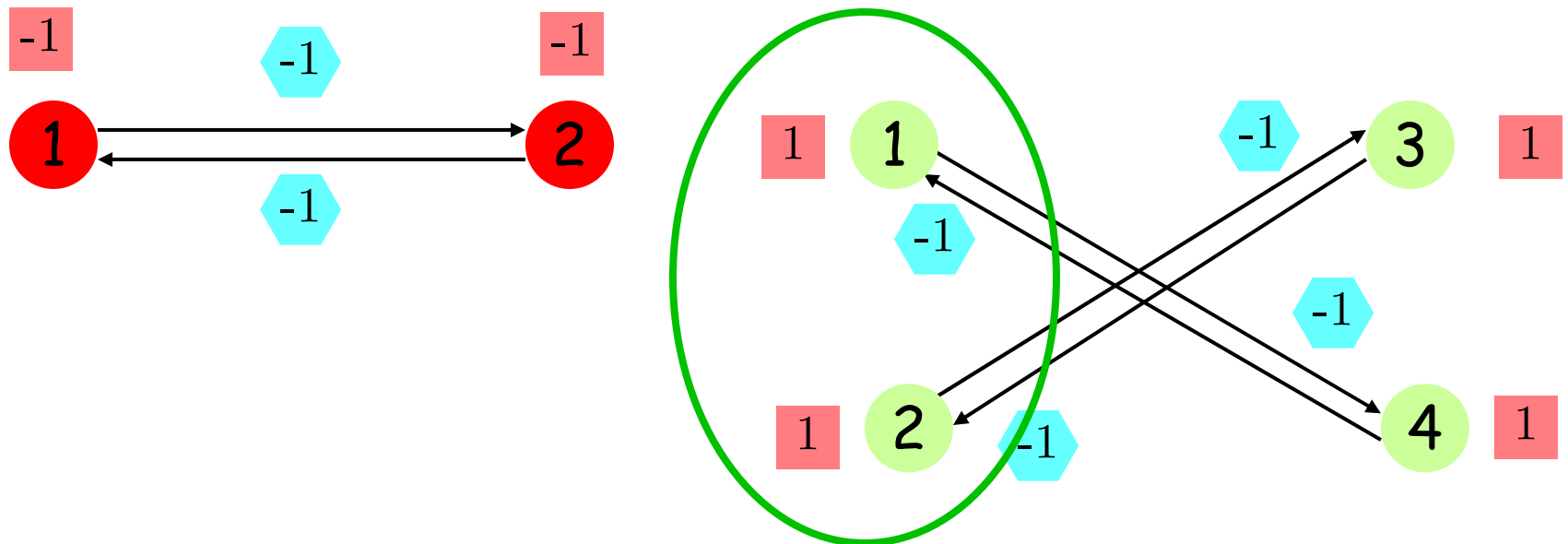
Idea

Simulate a Fully-Parallel Mode with Serial Mode



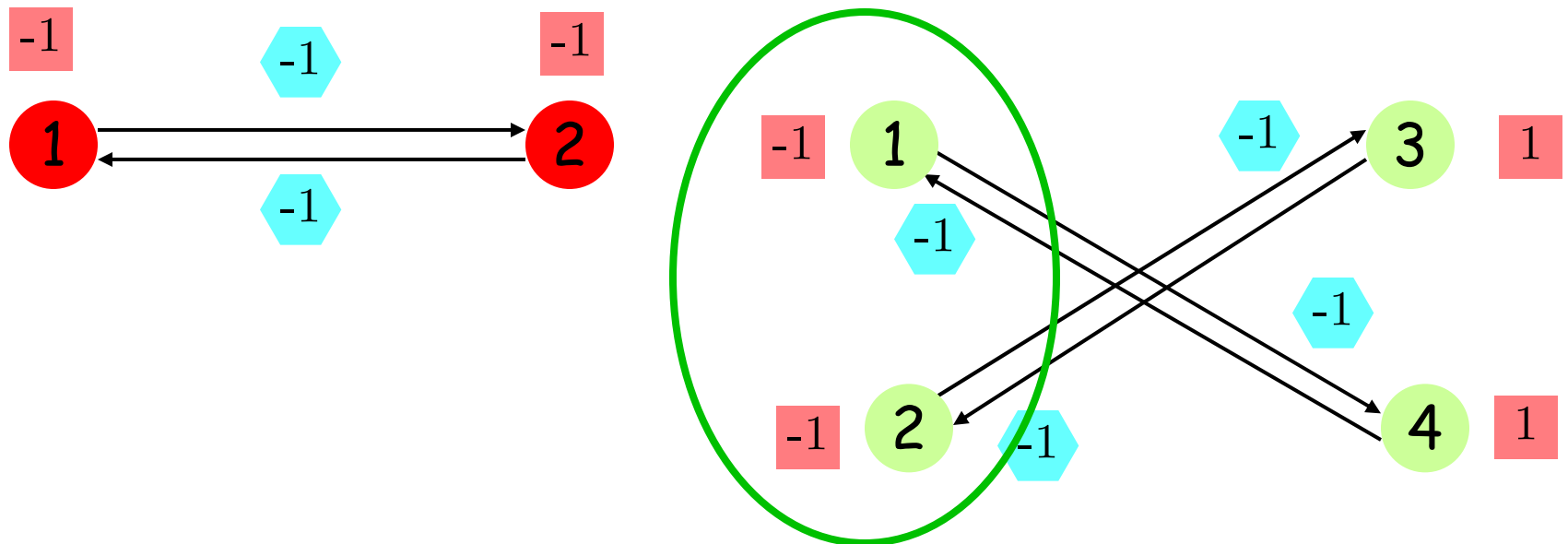
Idea

Simulate a Fully-Parallel Mode with Serial Mode



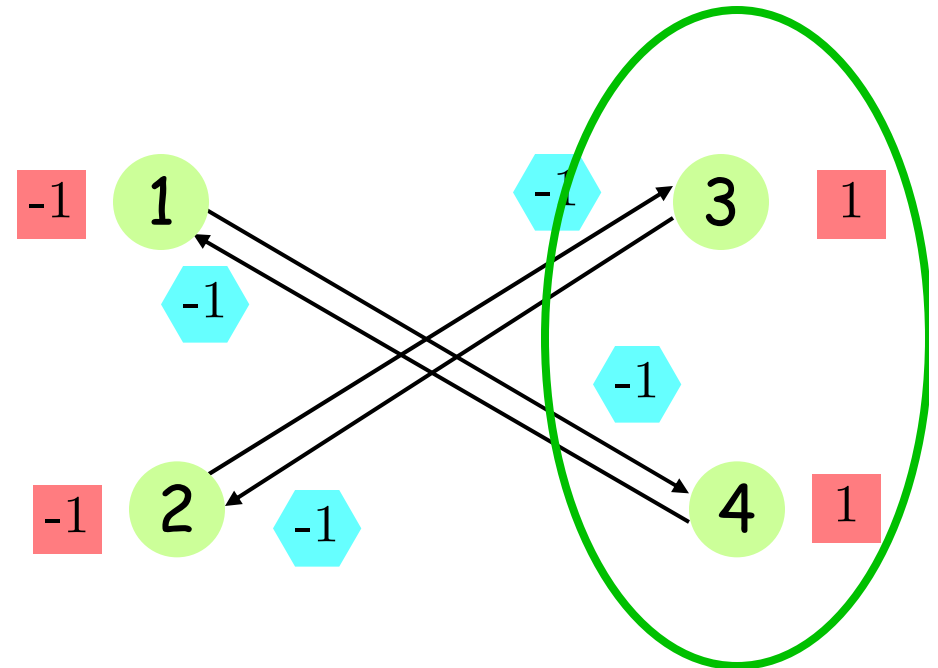
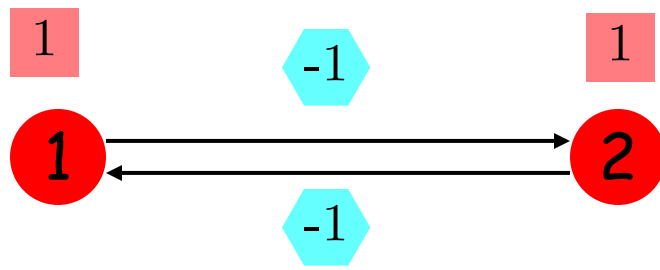
Idea

Simulate a Fully-Parallel Mode with Serial Mode



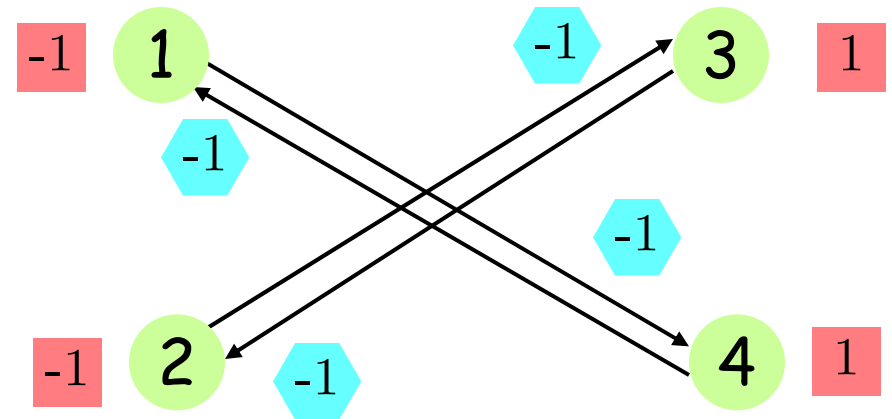
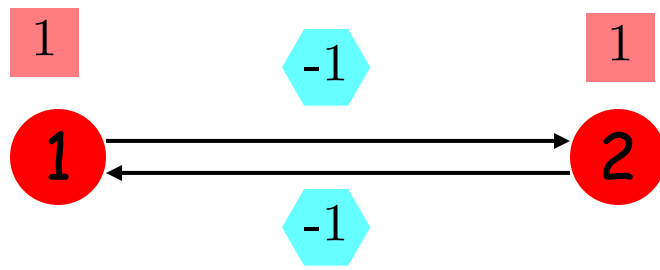
Idea

Simulate a Fully-Parallel Mode with Serial Mode



Idea

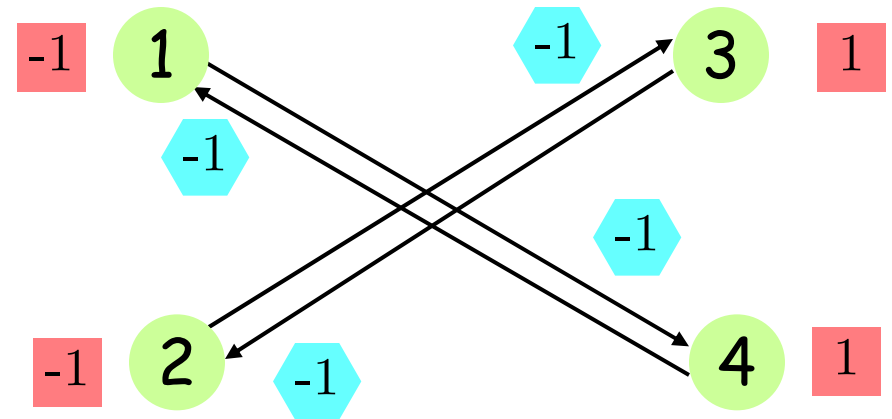
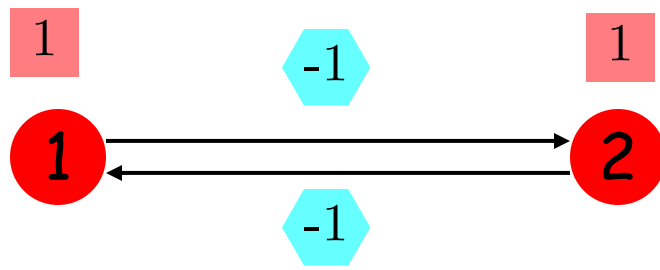
Simulate a Fully-Parallel Mode with Serial Mode



Time t (odd)

Idea

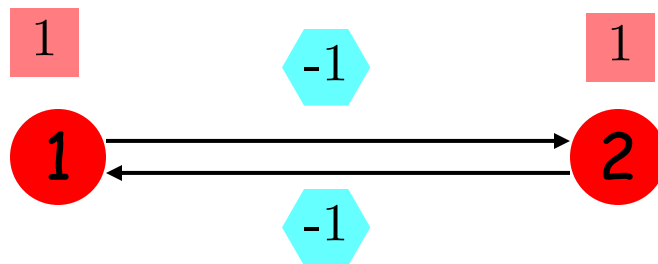
Simulate a Fully-Parallel Mode with Serial Mode



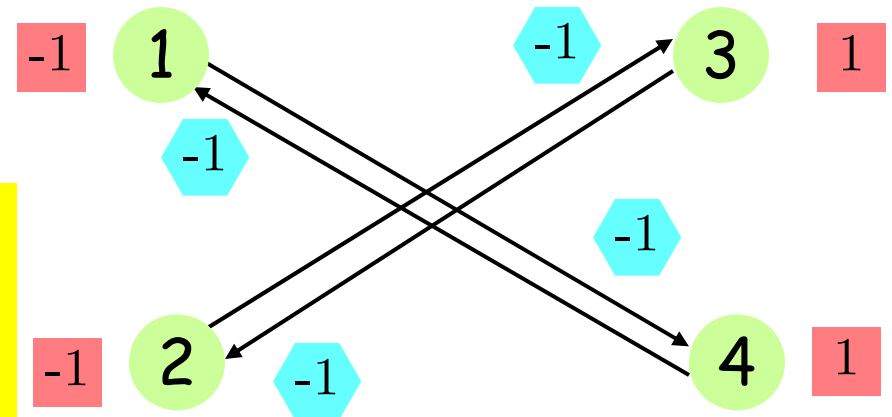
Time t (even)

Idea

Simulate a Fully-Parallel Mode with Serial Mode



The key:
The network on the
right operates in **serial mode**??

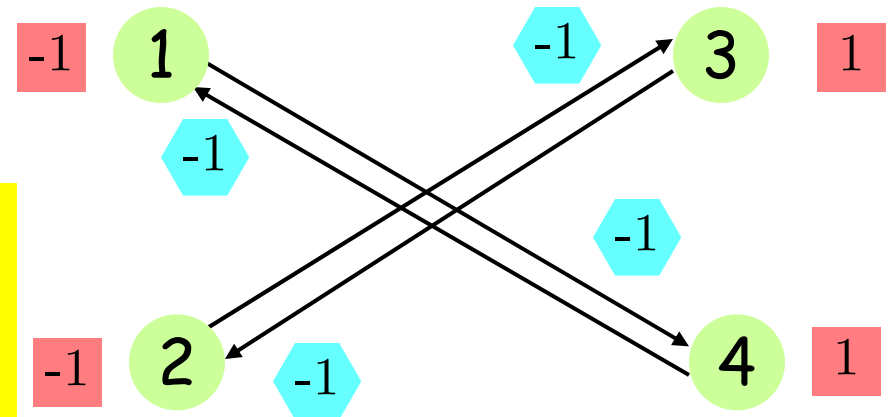
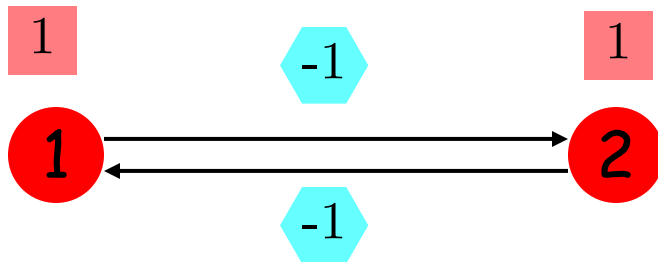


Time t (odd)

Time t (even)

Idea

Simulate a Fully-Parallel Mode with Serial Mode



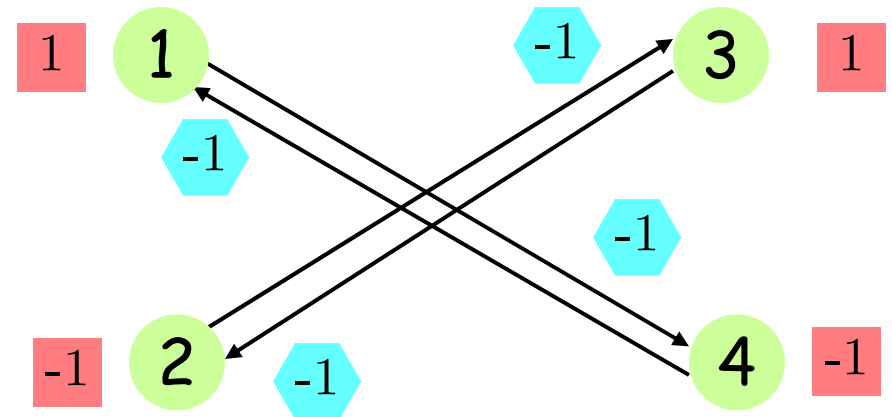
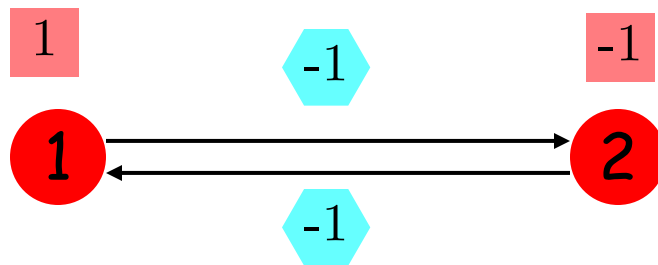
The key:
The network on the right operates in **serial mode**??

It converges to a stable state

The stable state represent either a **cycle of length 2** or a **stable state** in the original network

Idea

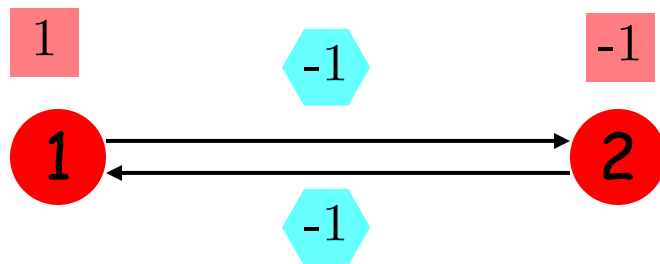
Simulate a Fully-Parallel Mode with Serial Mode



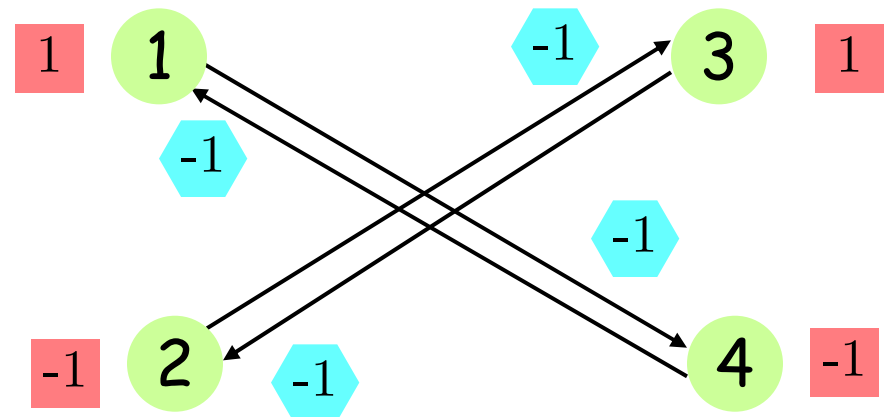
In a **stable state** in the original network the **states on the two sides** in the network on the right are equal

In General

Simulate a Fully-Parallel Mode with Serial Mode



$$N = (W, T)$$



$$\hat{N} = (\hat{W}, \hat{T})$$

$$\hat{W} = \begin{bmatrix} 0 & W \\ W & 0 \end{bmatrix}$$

$$\hat{T} = \begin{bmatrix} T \\ T \end{bmatrix}$$

Questions on Convergence

Cycle lengths	mode \ W	symmetric	antisymmetric
Example #	serial	1	?
	fully-parallel	1,2	4

1 → Hopfield 1982

2 → Goles 1985

3 → Goles 1986

Q1: Are the three cases "distinct"?

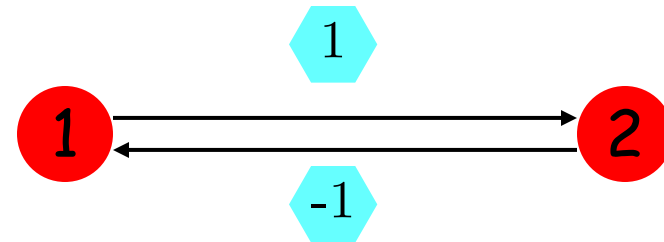
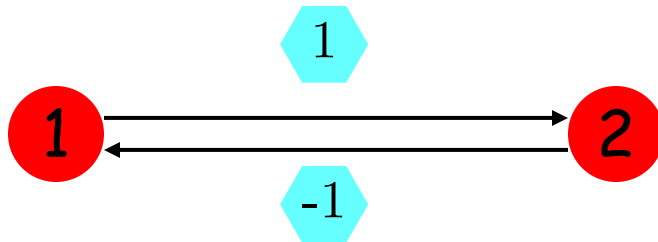
Q2: Elementary proof? (wo/energy)

Q3: Other "interesting" cases?

NO
YES
NO

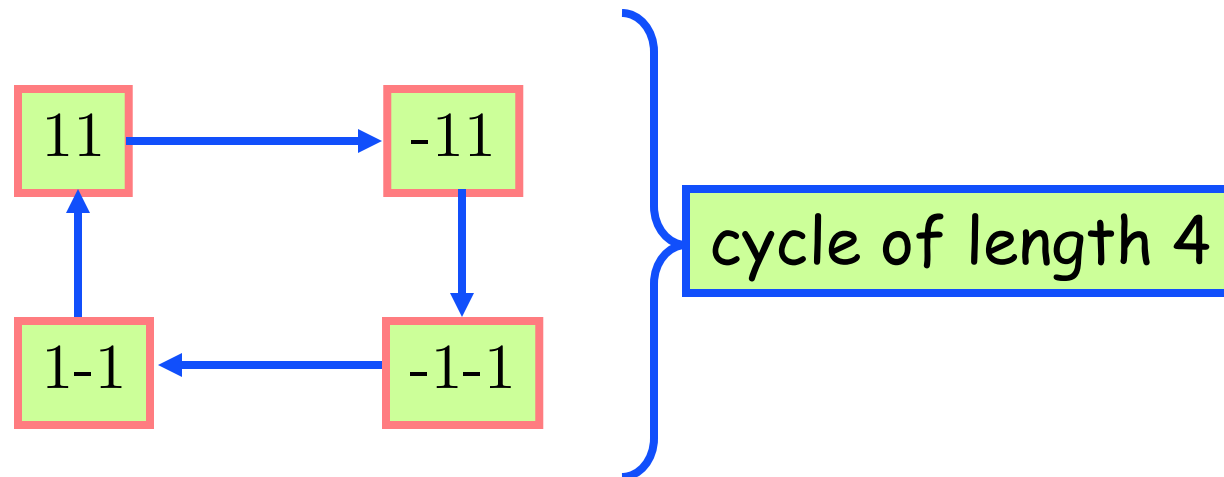
Answer to Q3

Serial Mode - Antisymmetric Weight Matrix



The state space:

Idea: We can build a counter



Homework # 6

Due Wednesday, March 15, 2006, 2:00 PM, at the TA office

Collaboration and/or discussion are not allowed on this homework set

3. Stable States

In this problem we prove a couple of simple properties of stable states in feedback networks. Assume in this problem that $N = (W, T)$ is a feedback network operating in a serial mode with W a symmetric matrix with zero diagonal and T being the all-zero vector.

- (a) Prove that if V is a stable state then $-V$ is also a stable state.
- (b) Prove that if V is a stable state then all the n vectors that differ from V in exactly one entry are not stable states.
- (c) Let W be the matrix with all -1's (besides the diagonal which is all 0's). What are the stable states of the feedback network related to W ?

1. LT Circuits for Symmetric Functions

In this problem you are asked to implement symmetric functions using circuits of LT gates. While in class we focused on depth-2 circuits, here the goal is to learn about depth size trade-offs.

Recall that a symmetric function $f(x_1, x_2, \dots, x_n)$ can be described by a vector of length $n + 1$ that we call the symmetric function table and denote it by $V(f) = (v_0, v_1, \dots, v_n)$, where

$$v_i = \begin{cases} 1 & \text{if for } |X| = i \quad f(X) = 1 \\ 0 & \text{otherwise} \end{cases}$$

Let f be a 4-variable symmetric function with $V(f) = \{11010\}$.

Let g be a 16-variable symmetric function with $V(g) = \{11010110101101000\}$.

- (a) Show a 4-gate, *layered* LT circuit that computes f .
- (b) Show a 2-gate, *non-layered* LT circuit that computes f .
- (c) Show a 4-gate, *non-layered* LT circuit that computes g .
- (d) Show a 5-gate, depth-3, *non-layered* LT circuit that computes the *XOR* function of 15 variables.
- (e) For $n = 4k^2 - 1$, where k is an integer greater than 1, show how to implement an n -variable *XOR* function using an LT *non-layered* circuit of depth 3 and a total of $3k - 1$ gates.

2. Reductions and Circuits

Reduction is a technique that helps in exploring relationships between circuits and the functions they compute. Let f and g be Boolean functions of n variables. Given a circuit C that computes f , if we show that g can be computed by C by moderately (at most polynomial in n) expanding the number of inputs to C , then we say that g can be reduced to f . Namely, g is easier to compute with circuits than f .

For example, let the Inner-Product-Mod 2 ($IP2$) be the following function that is defined for n even.

$$IP2(x_1, x_2, \dots, x_n) = (x_1 \wedge x_2) \oplus (x_3 \wedge x_4) \oplus \dots \oplus (x_{n-1} \wedge x_n)$$

For example, for $n = 4$,

$$IP2(x_1, x_2, x_3, x_4) = (x_1 \wedge x_2) \oplus (x_3 \wedge x_4)$$

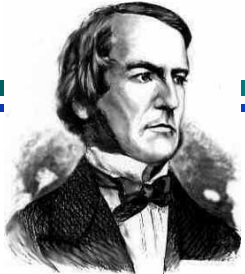
It is easy to see that XOR can be reduced to $IP2$ by using a circuit for $IP2$ with $2n$ inputs, where the n bits of XOR are fed into the odd inputs and the even inputs are set to 1. In the following assume that:

- *Majority* is a function that outputs the majority of the inputs (in case of a tie it outputs 1).
- *Product* is the function that has $2n$ inputs and $2n$ outputs and computes the multiplication of two n -bit integers of n bits each.
- *Sorting* is the function of n^2 inputs and n^2 outputs that gets n n -bit integers as inputs and outputs the n integers in a sorted order.

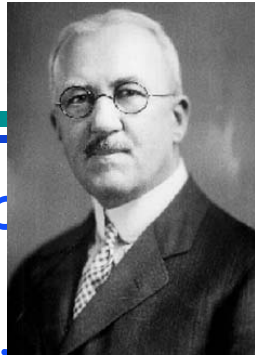
Please prove the following (you can assume that the constant 0 and 1 and both the variables and their complements are available as inputs).

- (a) Every function in TH can be reduced to *Majority*.
- (b) $IP2$ can be reduced to *Product*.
- (c) *Extra Credit*: $IP2$ can be reduced to *Sorting*.

Boole
1815-1864



Huntington
1874-1952



Stone
1903-1989



Fourier
1768-1830



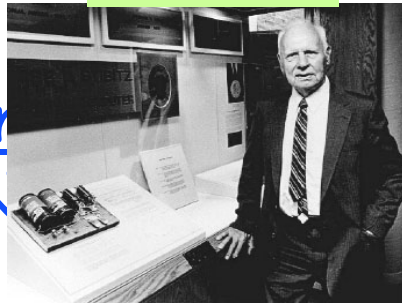
Leibniz
1646-1716



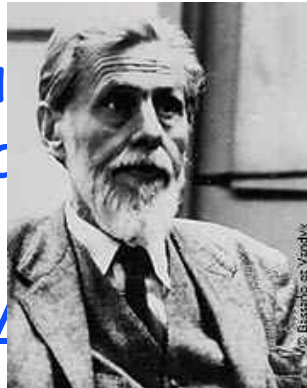
Shannon
1916-2001



Stibitz,
1904-1995



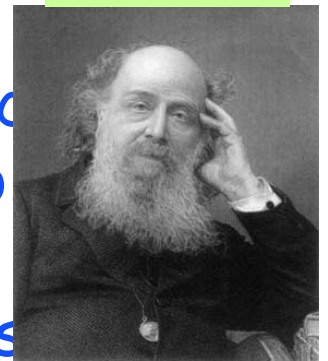
McCulloch
1899 - 1969



Pitts
1923 - 1969



Sylvester
1814-1897



Post
1897-1954



- algebra
- axiom
- and their representation
- methods
- complexity
- depth
- units
- opposed to functions) and implementation
- convergence in LT circuits

CNS 188a Overview

- Boolean algebra as an axiomatic system
- Boolean functions and their representations using Boolean formulas and spectral methods
- Implementing Boolean functions with relay circuits, circuits of AON (AND, OR, NOT) gates and LT (Linear Threshold) gates
- Analyzing the complexity (size and depth) of circuits
- Relations (as opposed to functions) and their implementation in circuits
- Feedback and convergence in LT circuits