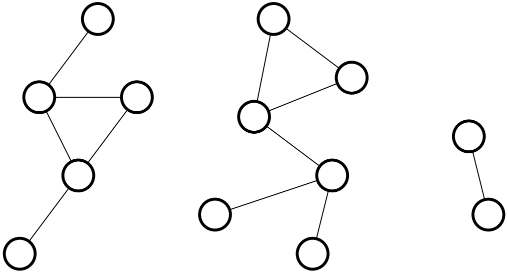
Self-Stabilizing Silent Disjunction in an Anonymous Network

Ajoy K. Datta   Stéphane Devismes   Lawrence L. Larmore

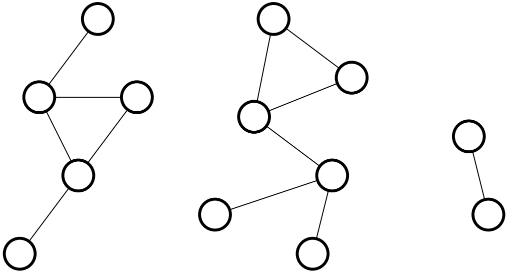
University of Nevada Las Vegas
Université Joseph Fourier, Grenoble

The Disjunction Problem



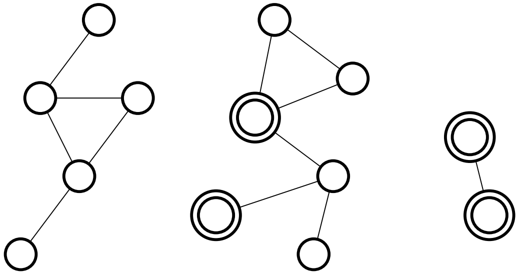
The Disjunction Problem

- **Anonymous** Network of **Processes**.



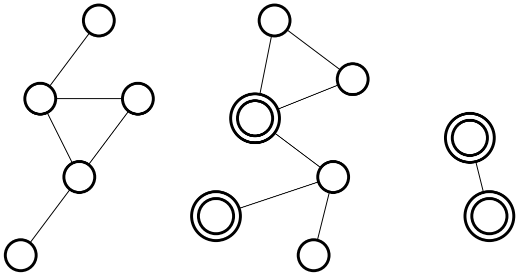
The Disjunction Problem

- Anonymous Network of Processes.
  - Each Process has **Input Bit**.



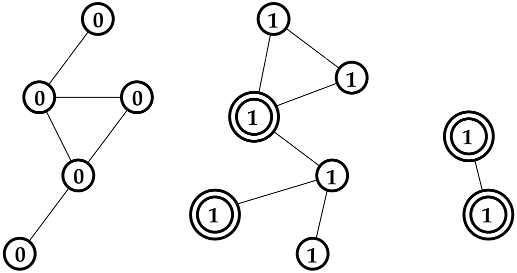
The Disjunction Problem

- Anonymous Network of Processes.
    - Each Process has Input Bit.
    - **Double Circle** $\Longleftrightarrow$ Input Bit = 1. Others are 0.



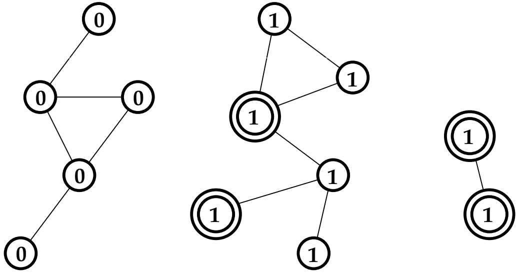
The Disjunction Problem

- Anonymous network of Processes.
  - Each Process has **Input Bit**.
  - Double Circle $\Longleftrightarrow$ Input Bit = 1. Others are 0.
  - **Output Bit** shown inside circle.



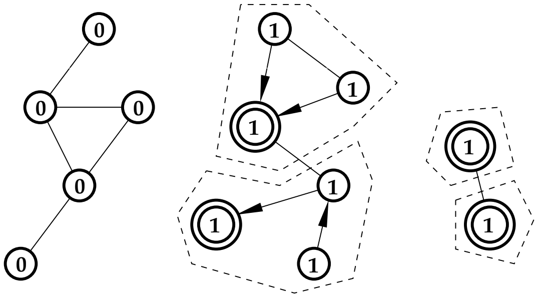
The Disjunction Problem

- Anonymous network of Processes.
  - Each Process has **Input Bit**.
  - Double Circle $\Longleftrightarrow$ Input Bit = 1. Others are 0.
  - Output Bit shown inside circle.
  - **Output Bit** of each Process is **Disjunction** of all Input Bits of Processes in its Component.



The Disjunction Problem

- Anonymous network of Processes.
  - Each Process has Input Bit.
  - Double Circle $\Longleftrightarrow$ Input Bit = 1. Others are 0.
  - Output Bit shown inside circle.
  - Output Bit of each Process is Disjunction of all Input Bits of Processes in its Component.
- (Construct **BFS Forest** Rooted at Processes with Input 1.)



Use Leader Election?

Use Leader Election?

- Makes the Disjunction Problem **Easy**.

Use Leader Election?

- Makes the Disjunction Problem Easy. **But** . . .
- **Impossible** in an **Anonymous** Network.

Use Leader Election?

- Makes the Disjunction Problem Easy. But . . .
- Impossible in an Anonymous Network.

Construct Spanning Tree?

Use Leader Election?

- Makes the Disjunction Problem Easy. But . . .
- Impossible in an Anonymous Network.

Construct Spanning Tree?

- Makes the Disjunction Problem **Easy**.

Use Leader Election?

- Makes the Disjunction Problem Easy. But . . .
- Impossible in an Anonymous Network.

Construct Spanning Tree?

- Makes the Disjunction Problem Easy. **But** . . .
- **Impossible** in an **Anonymous** Network.

Use Leader Election?

- Makes the Disjunction Problem Easy. But . . .
- Impossible in an Anonymous Network.

Construct Spanning Tree?

- Makes the Disjunction Problem Easy. But . . .
- Impossible in an Anonymous Network.

Build Clusters!

Use Leader Election?

- Makes the Disjunction Problem Easy. But . . .
- Impossible in an Anonymous Network.

Construct Spanning Tree?

- Makes the Disjunction Problem Easy. But . . .
- Impossible in an Anonymous Network.

Build Clusters!

- Each Process with Input Bit 1 is a **Clusterhead**.

Use Leader Election?

- Makes the Disjunction Problem Easy. But . . .
- Impossible in an Anonymous Network.

Construct Spanning Tree?

- Makes the Disjunction Problem Easy. But . . .
- Impossible in an Anonymous Network.

Build Clusters!

- Each Process with Input Bit 1 is a Clusterhead.
- Each Process **Joins** the Nearest Clusterhead.

Use Leader Election?

- Makes the Disjunction Problem Easy. But . . .
- Impossible in an Anonymous Network.

Construct Spanning Tree?

- Makes the Disjunction Problem Easy. But . . .
- Impossible in an Anonymous Network.

Build Clusters!

- Each Process with Input Bit 1 is a Clusterhead.
- Each Process Joins the Nearest Clusterhead.
- **Local BFS Tree** in Every Cluster, Rooted at Clusterhead. (Defined by Parent Pointers and Levels.)

Simple Flooding

Simple Flooding

- Works if **Clean Start.**

Simple Flooding

- Works if Clean Start.
- If **Arbitrary Start:**

Simple Flooding

- Works if Clean Start.
- If **Arbitrary Start:**
  - Works if Some Process has **Input 1.**

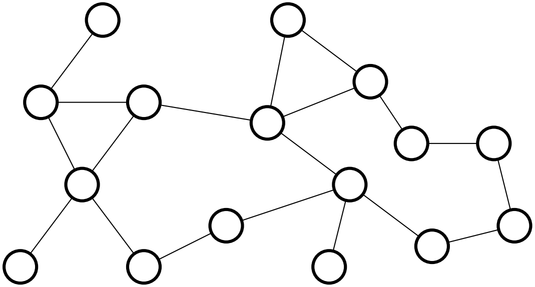
Simple Flooding

- Works if Clean Start.
- If **Arbitrary Start:**
  - Works if Some Process has Input 1.
  - All Processes have **Input 0:** might go into **Endless Loop!**

Clean Start

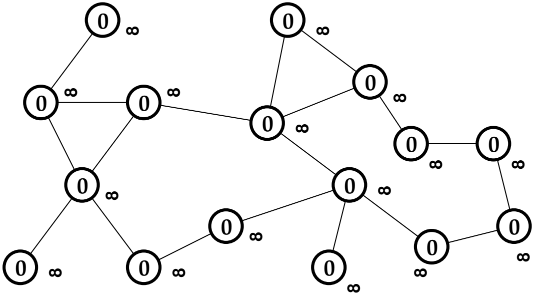
Clean Start

- **All Input Bits 0.**



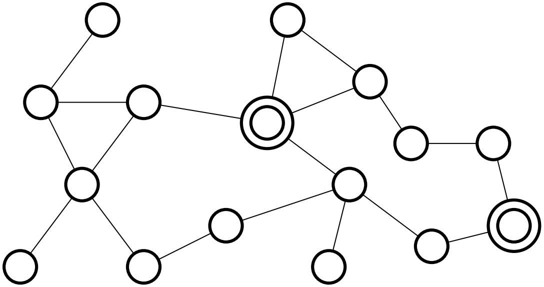
Clean Start

- All Input Bits 0.
- Clean Configuration is **Final.**



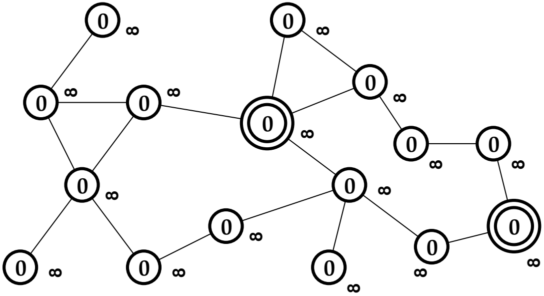
Clean Start

- **Some Input Bits = 1.**



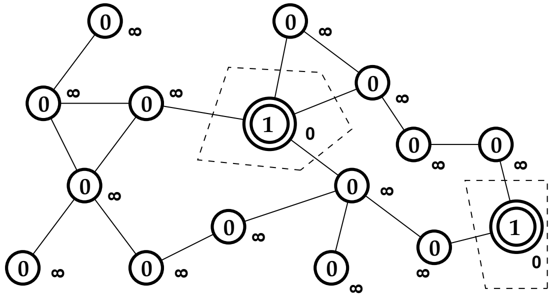
Clean Start

- Some Input Bits = 1.
- Clean Configuration is **Not Final.**



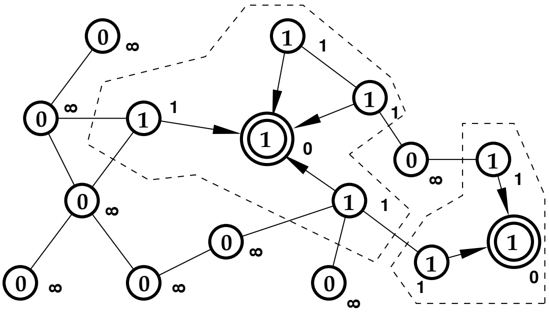
Clean Start

- Some Input Bits = 1.
- Clean Configuration is not Final.
- Clusterheads Execute **Reset.**



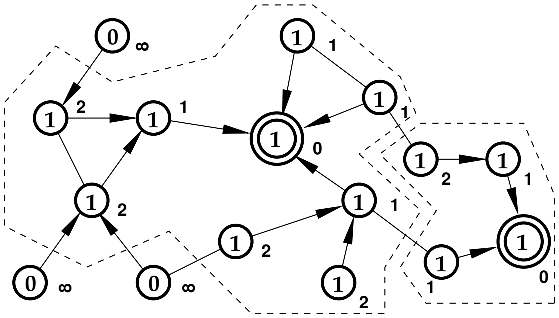
Clean Start

- Some Input Bits = 1.
- Clean Configuration is not Final.
- Clusterheads Execute Reset.
- Adjacent Processes Execute **Join.**



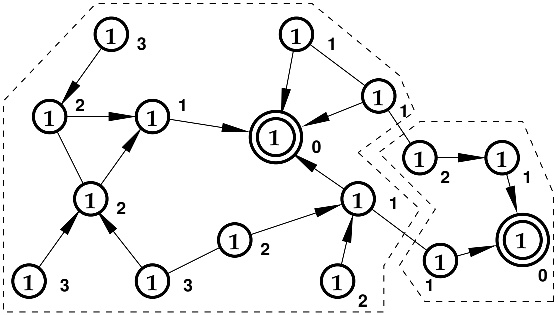
Clean Start

- Some Input Bits = 1.
- Clean Configuration is not Final.
- Clusterheads Execute Reset.
- Adjacent Processes Execute **Join.**



Clean Start

- Some Input Bits = 1.
- Clean Configuration is not Final.
- Clusterheads Execute Reset.
- Adjacent Processes Execute Join.
- **Flooding:** All Parent Pointers and Levels are Computed in at Most $(1 + \mathit{Diam})$ **Rounds.**



Simple Flooding Is Not Self-Stabilizing!

Simple Flooding Is Not Self-Stabilizing!

- **Arbitrary Initial Configuration?**

Simple Flooding Is Not Self-Stabilizing!

- Arbitrary Initial Configuration.
- No Problem if Some Process has **Input 1.**

Simple Flooding Is Not Self-Stabilizing!

- Arbitrary Initial Configuration.
- No Problem if Some Process has Input 1.
- Otherwise, might go into **Endless Loop.**

Arbitrary Start: Output = 1: No Problem!

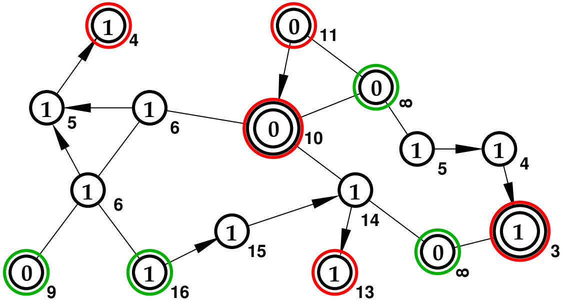
VIII

Arbitrary Start: Output = 1: No Problem!

- Some Input Bits 1.

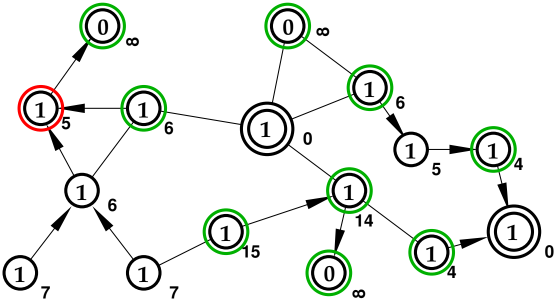
Arbitrary Start: Output = 1: No Problem!

- Some Input Bits 1.
- Red = Enabled to Reset.
- Green = Enabled to Join.



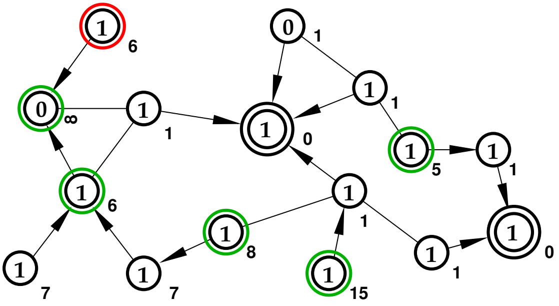
Arbitrary Start: Output = 1: No Problem!

- Some Input Bits 1.
- Red = Enabled to Reset.
- Green = Enabled to Join.
- **Clusterheads Reset**.



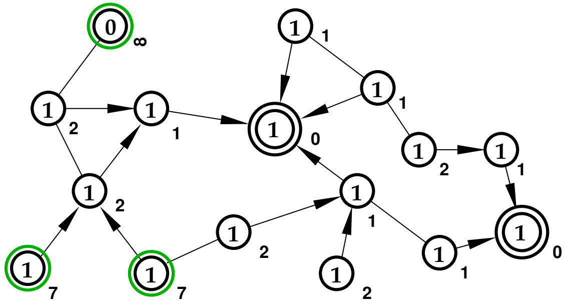
Arbitrary Start: Output = 1: No Problem!

- Some Input Bits 1.
- Red = Enabled to Reset.
- Green = Enabled to Join.
- Clusterheads Reset.
- **Flooding** Moves at Speed 1.



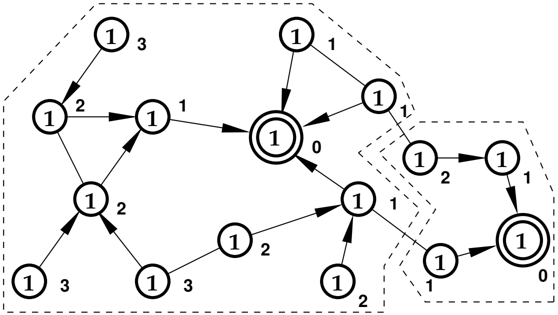
Arbitrary Start: Output = 1: No Problem!

- Some Input Bits 1.
- Red = Enabled to Reset.
- Green = Enabled to Join.
- Clusterheads Reset.
- Flooding Moves at Speed 1.



Arbitrary Start: Output = 1: No Problem!

- Some Input Bits 1.
- Red = Enabled to Reset.
- Green = Enabled to Join.
- Clusterheads Reset.
- Flooding Moves at Speed 1.
- **Convergence** within $1 + \mathit{Diam}$ **Rounds**.



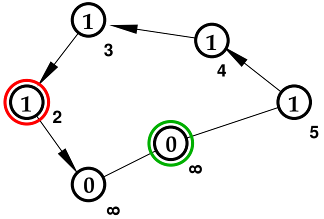
Arbitrary Start: Output = 0: Serious Problem!

Arbitrary Start: Output = 0: Serious Problem!

- All Input Bits 0.

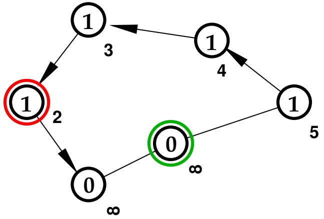
Arbitrary Start: Output = 0: Serious Problem!

- All Input Bits 0.
- **Output Bits Inside Circles.**



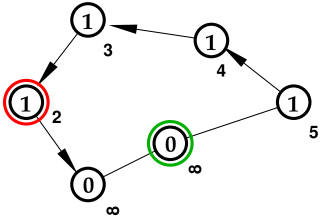
Arbitrary Start: Output = 0: Serious Problem!

- All Input Bits 0.
- Output Bits Inside Circles.
- Red = Enabled to Reset.



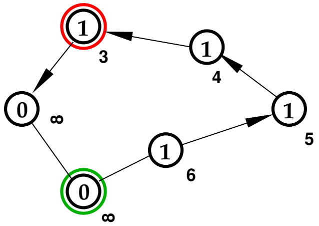
Arbitrary Start: Output = 0: Serious Problem!

- All Input Bits 0.
- Output Bits Inside Circles.
- Red = Enabled to Reset.
- Green = Enabled to Join.



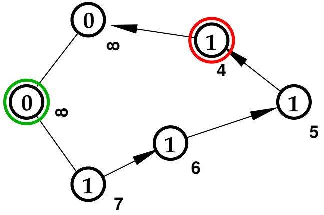
Arbitrary Start: Output = 0: Serious Problem!

- All Input Bits 0.
- Output Bits Inside Circles.
- Red = Enabled to Reset.
- Green = Enabled to Join.
- **Chain Deletes** at Head End.
- **But Recruits** at Tail (Leaf) End.



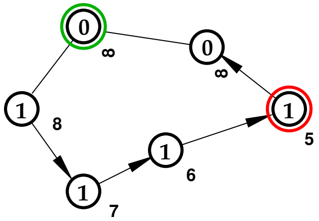
Arbitrary Start: Output = 0: Serious Problem!

- All Input Bits 0.
- Output Bits Inside Circles.
- Red = Enabled to Reset.
- Green = Enabled to Join.
- Chain Deletes at Head End.
- But Recruits at Tail (Leaf) End.
- Keeps Going **Around!**



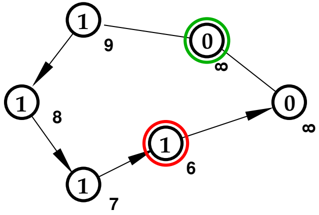
Arbitrary Start: Output = 0: Serious Problem!

- All Input Bits 0.
- Output Bits Inside Circles.
- Red = Enabled to Reset.
- Green = Enabled to Join.
- Chain Deletes at Head End.
- But Recruits at Tail (Leaf) End.
- Keeps Going **Around!**



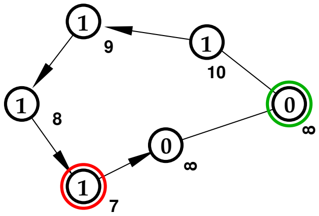
Arbitrary Start: Output = 0: Serious Problem!

- All Input Bits 0.
- Output Bits Inside Circles.
- Red = Enabled to Reset.
- Green = Enabled to Join.
- Chain Deletes at Head End.
- But Recruits at Tail (Leaf) End.
- Keeps Going **Around!**



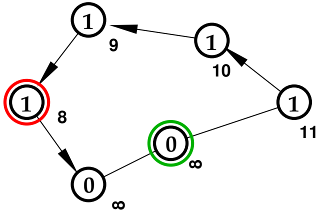
Arbitrary Start: Output = 0: Serious Problem!

- All Input Bits 0.
- Output Bits Inside Circles.
- Red = Enabled to Reset.
- Green = Enabled to Join.
- Chain Deletes at Head End.
- But Recruits at Tail (Leaf) End.
- Keeps Going **Around!**



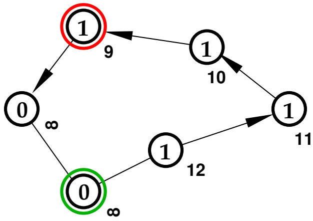
Arbitrary Start: Output = 0: Serious Problem!

- All Input Bits 0.
- Output Bits Inside Circles.
- Red = Enabled to Reset.
- Green = Enabled to Join.
- Chain Deletes at Head End.
- But Recruits at Tail (Leaf) End.
- Keeps Going Around.
- **Return** to **First Configuration**, Except for Levels.



Arbitrary Start: Output = 0: Serious Problem!

- All Input Bits 0.
- Output Bits Inside Circles.
- Red = Enabled to Reset.
- Green = Enabled to Join.
- Chain Deletes at Head End.
- But Recruits at Tail (Leaf) End.
- Keeps Going Around.
- Return to First Configuration, Except for Levels.
- **Endless!**



Color Wave Algorithm

Color Wave Algorithm

Self-Stabilizing and Silent

Color Wave Algorithm

Self-Stabilizing and Silent

Complexities

Color Wave Algorithm

Self-Stabilizing and Silent

Complexities

- **Arbitrary Initial Configuration.**

Color Wave Algorithm

Self-Stabilizing and Silent

Complexities

- **Arbitrary Initial Configuration.**
  - $3Diam + O(1)$ **Rounds** to Achieve Legitimacy if Output = 1.

Color Wave Algorithm

Self-Stabilizing and Silent

Complexities

- **Arbitrary Initial Configuration.**
  - $3Diam + O(1)$ Rounds to Achieve Legitimacy if Output = 1.
  - $5Diam + O(1)$ **Rounds** to Achieve Silence if Output = 1.

Color Wave Algorithm

Self-Stabilizing and Silent

Complexities

- **Arbitrary Initial Configuration.**
  - $3Diam + O(1)$ Rounds to Achieve Legitimacy if Output = 1.
  - $5Diam + O(1)$ Rounds to Achieve Silence if Output = 1.
  - $O(n)$ **Rounds** if Output = 0. **The Hard Case!**

Color Wave Algorithm

Self-Stabilizing and Silent

Complexities

- **Arbitrary Initial Configuration.**
  - $3Diam + O(1)$ Rounds to Achieve Legitimacy if Output = 1.
  - $5Diam + O(1)$ Rounds to Achieve Silence if Output = 1.
  - $O(n)$ Rounds if Output = 0. **The Hard Case!**
- **Clean Initial Configuration.**

Color Wave Algorithm

Self-Stabilizing and Silent

Complexities

- **Arbitrary Initial Configuration.**
  - $3Diam + O(1)$ Rounds to Achieve Legitimacy if Output = 1.
  - $5Diam + O(1)$ Rounds to Achieve Silence if Output = 1.
  - $O(n)$ Rounds if Output = 0. **The Hard Case!**
- **Clean Initial Configuration.**
  - **If Output = 1: Same as Arbitrary.** (No Help.)

Color Wave Algorithm

Self-Stabilizing and Silent

Complexities

- **Arbitrary Initial Configuration.**
  - $3Diam + O(1)$ Rounds to Achieve Legitimacy if Output = 1.
  - $5Diam + O(1)$ Rounds to Achieve Silence if Output = 1.
  - $O(n)$ Rounds if Output = 0. **The Hard Case!**
- **Clean Initial Configuration.**
  - If Output = 1: Same as Arbitrary. (No Help.)
  - **If Output = 0: Zero Rounds.** (Already in Final Configuration)

Color Wave Algorithm

Self-Stabilizing and Silent

Complexities

- **Arbitrary Initial Configuration.**
  - $3Diam + O(1)$ Rounds to Achieve Legitimacy if Output = 1.
  - $5Diam + O(1)$ Rounds to Achieve Silence if Output = 1.
  - $O(n)$ Rounds if Output = 0. **The Hard Case!**
- **Clean Initial Configuration.**
  - If Output = 1: Same as Arbitrary. (No Help.)
  - If Output = 0: Zero Rounds. (Already in Final Configuration)
- **Space Complexity.**

IX

Color Wave Algorithm

Self-Stabilizing and Silent

Complexities

- **Arbitrary Initial Configuration.**
  - $3Diam + O(1)$ Rounds to Achieve Legitimacy if Output = 1.
  - $5Diam + O(1)$ Rounds to Achieve Silence if Output = 1.
  - $O(n)$ Rounds if Output = 0. **The Hard Case!**
- **Clean Initial Configuration.**
  - If Output = 1: Same as Arbitrary. (No Help.)
  - If Output = 0: Zero Rounds. (Already in Final Configuration)
- **Space Complexity.**
  - $O(\log Diam + Degree)$ **Per Process.**

Purpose of Color Waves

Purpose of Color Waves

- **Prevent Indefinite Growth of Fictitious Trees.**

Purpose of Color Waves

- Prevent Indefinite Growth of Fictitious Trees.

Side Effects of Color Waves

XIV

Purpose of Color Waves

- Prevent Indefinite Growth of Fictitious Trees.

Side Effects of Color Waves

- **Slow Down Algorithm by a Factor of Three**

Purpose of Color Waves

- Prevent Indefinite Growth of Fictitious Trees.

Side Effects of Color Waves

- Slow Down Algorithm by a Factor of Three
- **After Legitimacy, Color Waves could Run Forever.**

Purpose of Color Waves

- Prevent Indefinite Growth of Fictitious Trees.

Side Effects of Color Waves

- Slow Down Algorithm by a Factor of Three
- After Legitimacy, Color Waves could Run Forever.

Counteract Effect with Done Waves

Purpose of Color Waves

- Prevent Indefinite Growth of Fictitious Trees.

Side Effects of Color Waves

- Slow Down Algorithm by a Factor of Three
- After Legitimacy, Color Waves could Run Forever.

Counteract Effect with Done Waves

- **Convergecast: Leaves Detect Algorithm Done**

Purpose of Color Waves

- Prevent Indefinite Growth of Fictitious Trees.

Side Effects of Color Waves

- Slow Down Algorithm by a Factor of Three
- After Legitimacy, Color Waves could Run Forever.

Counteract Effect with Done Waves

- Convergecast: Leaves Detect Algorithm Done
- **Root (Clusterhead) Color Freezes.**

Purpose of Color Waves

- Prevent Indefinite Growth of Fictitious Trees.

Side Effects of Color Waves

- Slow Down Algorithm by a Factor of Three
- After Legitimacy, Color Waves could Run Forever.

Counteract Effect with Done Waves

- Convergecast: Leaves Detect Algorithm Done
- Root (Clusterhead) Color Freezes.
- **Color Lock Results Within $O(Diam)$ Rounds.**

Purpose of Color Waves

- Prevent Indefinite Growth of Fictitious Trees.

Side Effects of Color Waves

- Slow Down Algorithm by a Factor of Three
- After Legitimacy, Color Waves could Run Forever.

Counteract Effect with Done Waves

- Convergecast: Leaves Detect Algorithm Done
- Root (Clusterhead) Color Freezes.
- Color Lock Results Within $O(\mathit{Diam})$ Rounds.
- **Silence.**

Color Wave Details:

Color Wave Details:

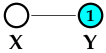
- Process with **Output Bit = 1** has **Color:** $0 = \textcircled{1}$, $1 = \textcircled{1}$.

Color Wave Details:

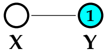
- Process with Output Bit = 1 has Color: $0 = \textcircled{1}$, $1 = \textcircled{1}$.
- If Process **X** Executes **Join**, Attaching to Process **Y**:

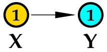
Color Wave Details:

- Process with Output Bit = 1 has Color: 0 = ①, 1 = ①.
- If Process **X** Executes **Join**, Attaching to Process **Y**:
    - **Y** must have Color 1:

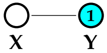


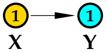
Color Wave Details:

- Process with Output Bit = 1 has Color: 0 = ①, 1 = ①.
- If Process **X** Executes **Join**, Attaching to Process **Y**:
    - **Y** must have Color 1: 

    - Color of **X** Becomes 0: 

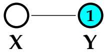
Color Wave Details:

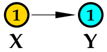
- Process with Output Bit = 1 has Color: 0 $=$, 1 $=$.
- If Process **X** Executes Join, Attaching to Process **Y**:
  - **Y** must have Color 1: 

  - Color of **X** Becomes 0: 

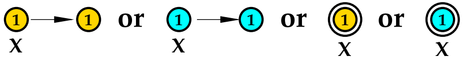
- Process **X** can **Change Color** if the **Following Conditions Hold:**

Color Wave Details:

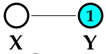
- Process with Output Bit = 1 has Color: 0 = , 1 = .
- If Process **X** Executes Join, Attaching to Process **Y**:
  - **Y** must have Color 1: 

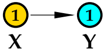
  - Color of **X** Becomes 0: 

- Process **X** can **Change Color** if the **Following Conditions Hold:**
  - **X** has **Same Color** as its **Parent**, or is **Clusterhead:**

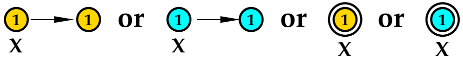


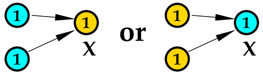
Color Wave Details:

- Process with Output Bit = 1 has Color: 0 = ①, 1 = ①.
- If Process **X** Executes Join, Attaching to Process **Y**:
  - **Y** must have Color 1: 

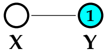
  - Color of **X** Becomes 0: 

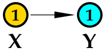
- Process **X** can **Change Color** if the **Following Conditions Hold:**
  - **X** has Same Color as its Parent, or is Clusterhead:



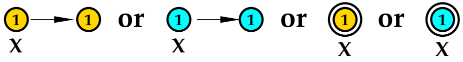
  - **Children** have **Opposite Color**: 

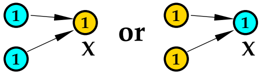
Color Wave Details:

- Process with Output Bit = 1 has Color: $0 =$ ①, $1 =$ ①.
- If Process **X** Executes Join, Attaching to Process **Y**:
  - **Y** must have Color 1: 

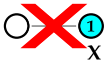
  - Color of **X** Becomes 0: 

- Process **X** can **Change Color** if the **Following Conditions Hold:**
  - **X** has Same Color as its Parent, or is Clusterhead:

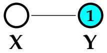


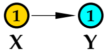
  - Children have Opposite Color: 

  - If **Color = 1, X** **Cannot Change Color** if Any Neighbor is

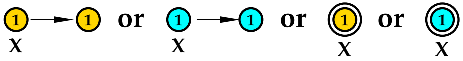
    Enabled to Attach to **X:** 

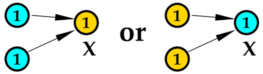
Color Wave Details:

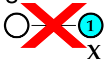
- Process with Output Bit = 1 has Color: 0 = , 1 = .
- If Process **X** Executes Join, Attaching to Process **Y**:
  - **Y** must have Color 1: 

  - Color of **X** Becomes 0: 

- Process **X** can Change Color if the Following Conditions Hold:
  - **X** has Same Color as its Parent, or is Clusterhead:

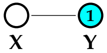


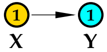
  - Children have Opposite Color: 

  - If Color = 1: **X** Cannot Change Color if Any Neighbor is Enabled to Attach to **X**: 

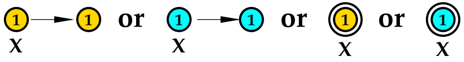
- When **Clusterhead** Changes Color, a **Color Wave** is **Absorbed.**

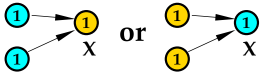
Color Wave Details:

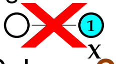
- Process with Output Bit = 1 has Color: 0 = , 1 = .
- If Process **X** Executes Join, Attaching to Process **Y**:
  - **Y** must have Color 1: 

  - Color of **X** Becomes 0: 

- Process **X** can Change Color if the Following Conditions Hold:
  - **X** has Same Color as its Parent, or is Clusterhead:



  - Children have Opposite Color: 

  - If Color = 1: **X** Cannot Change Color if Any Neighbor is Enabled to Attach to **X**: 

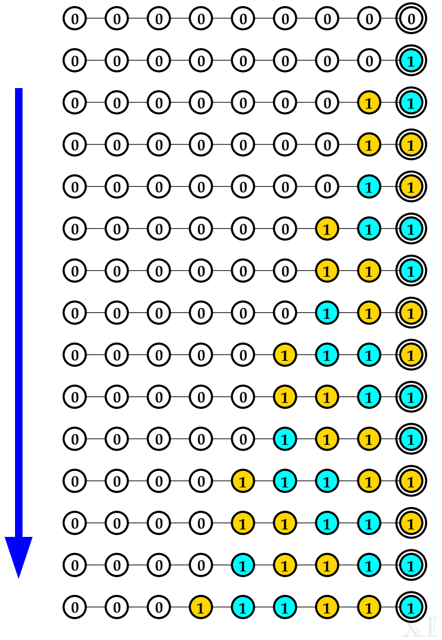
- When **Clusterhead** Changes Color, a **Color Wave** is **Absorbed.**
  - **False Roots Cannot Absorb Color Waves**.

Color Waves:

XI

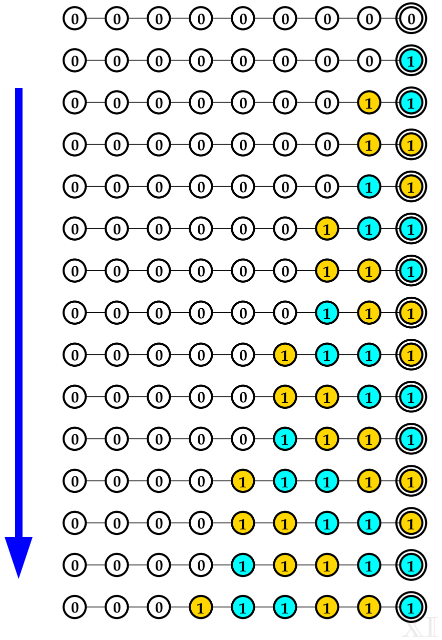
Color Waves:

- **Chain Example.**



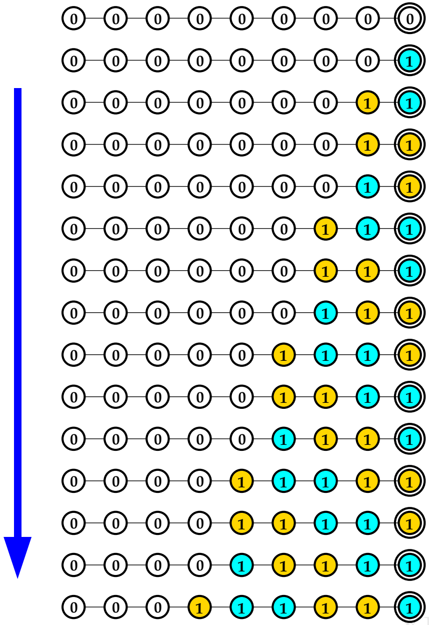
Color Waves:

- Chain Example.
- **One Process has Input = 1.**



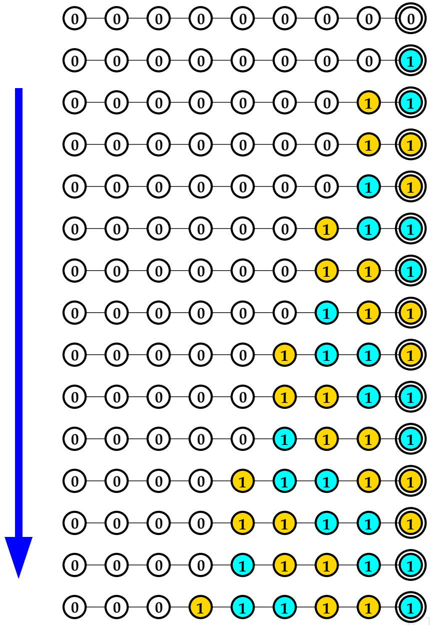
Color Waves:

- Chain Example.
- One Process has Input = 1.
- **Arrow** Shows Flow of Time.



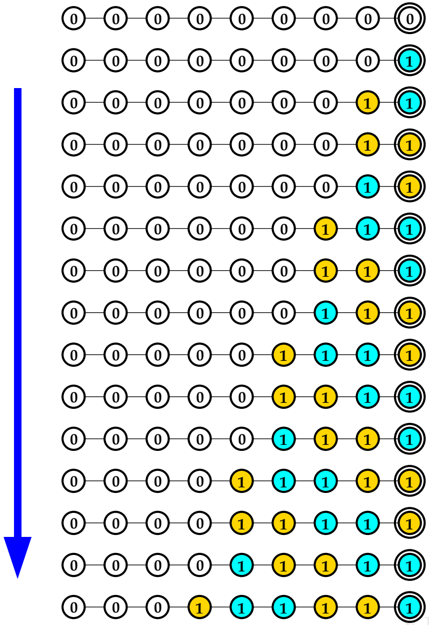
Color Waves:

- Chain Example.
- One Process has Input = 1.
- Arrow Shows Flow of Time.
- **Color Waves** Move
  Toward Clusterhead



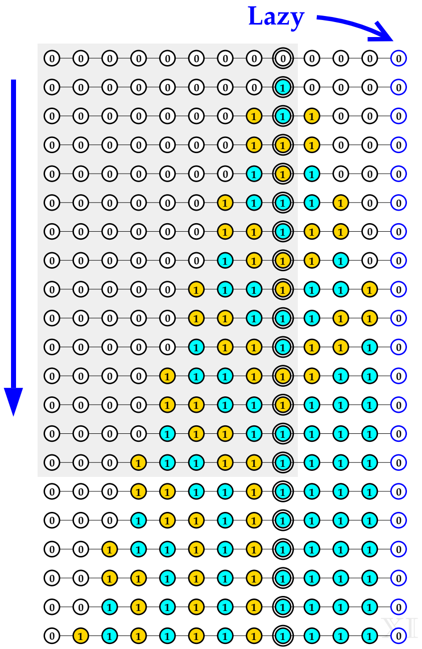
Color Waves:

- Chain Example.
- One Process has Input = 1.
- Arrow Shows Flow of Time.
- Color Waves Move
  Toward Clusterhead
- **Growth Rate = 1/3**.



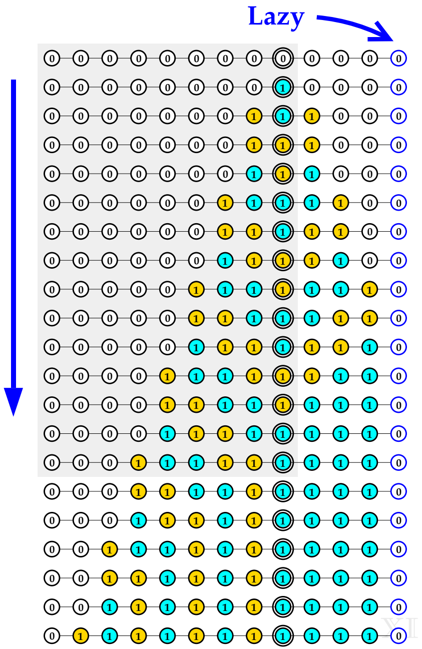
Color Waves:

- Chain Example.
- One Process has Input = 1.
- Arrow Shows Flow of Time.
- Color Waves Move
  Toward Clusterhead
- Growth Rate = 1/3.
- **If Any Process is “Lazy":**



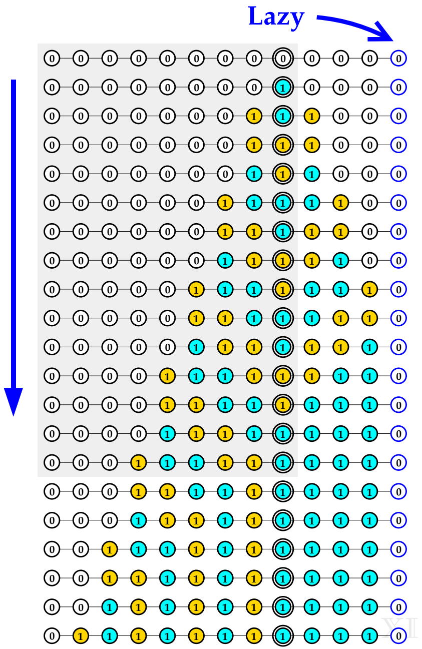
Color Waves:

- Chain Example.
- One Process has Input = 1.
- Arrow Shows Flow of Time.
- Color Waves Move
  Toward Clusterhead
- Growth Rate = 1/3.
- **If Any Process is "Lazy":**
  Eventual **Color Lock**:



Color Waves:

- Chain Example.
- One Process has Input = 1.
- Arrow Shows Flow of Time.
- Color Waves Move
  Toward Clusterhead
- Growth Rate = 1/3.
- **If Any Process is “Lazy":**
  Eventual **Color Lock**:
  That is, Lazy Process =
  Only Enabled Process.



Getting Rid of False Roots:

Getting Rid of False Roots:

- **If Any Process Stops Executing:**

Getting Rid of False Roots:

- **If Any Process Stops Executing:**
    - Eventual **Color Lock**:

Getting Rid of False Roots:

- If Any Process Stops Executing:
    - Eventual Color Lock:
- Consider a **False Root**, **R.**

Getting Rid of False Roots:

- If Any Process Stops Executing:
    - Eventual Color Lock:
- Consider a **False Root**, **R**.
    - **R** is Enabled Only to **Reset**, and thus Cannot Change Color.

Getting Rid of False Roots:

- If Any Process Stops Executing:
  - Eventual Color Lock:
- Consider a **False Root**, **R**.
  - **R** is Enabled Only to Reset, and thus Cannot Change Color.
  - **Tree Rooted at R Cannot Grow Forever.**

XIII

Getting Rid of False Roots:

- If Any Process Stops Executing:
  - Eventual Color Lock:
- Consider a **False Root**, **R**.
  - **R** is Enabled Only to Reset, and thus Cannot Change Color.
  - Tree Rooted at **R** Cannot Grow Forever.
  - **How can you Prove That?**

XIII

Getting Rid of False Roots:

- If Any Process Stops Executing:
  - Eventual Color Lock:
- Consider a **False Root**, **R**.
  - **R** is Enabled Only to Reset, and thus Cannot Change Color.
  - Tree Rooted at **R** Cannot Grow Forever.
  - How can you Prove That?
  - **Use Energy!**

XIII

Energy

Energy

- **Energy(X): Positive Integer** for **X** of **Output = 1**.

Energy

- Energy(**X**): Positive Integer for **X** of Output = 1.
- Defined **Recursively.**

Energy

- Energy(**X**): Positive Integer for **X** of Output = 1.
- Defined **Recursively.**
  - **Energy(X)** $= 1$ if **X** is **Leaf** of **Color = 0.**

Energy

- Energy(**X**): Positive Integer for **X** of Output = 1.
- Defined **Recursively.**
  - Energy(**X**) $= 1$ if **X** is Leaf of Color = 0.
  - **Energy(X)** $= 2$ if **X** is **Leaf** of **Color = 1.**

Energy

- Energy(**X**): Positive Integer for **X** of Output = 1.
- Defined **Recursively.**
  - Energy(**X**) $= 1$ if **X** is Leaf of Color = 0.
  - Energy(**X**) $= 2$ if **X** is Leaf of Color = 1.
  - **X Not Leaf: Energy**(**X**) = **Maximum** of:

Energy

- Energy(**X**): Positive Integer for **X** of Output = 1.
- Defined **Recursively.**
    - Energy(**X**) $= 1$ if **X** is Leaf of Color = 0.
    - Energy(**X**) $= 2$ if **X** is Leaf of Color = 1.
    - **X Not Leaf: Energy**(**X**) = **Maximum** of:
        - **1 + Energy** of any **Child of Opposite Color**.

Energy

- Energy(**X**): Positive Integer for **X** of Output = 1.
- Defined **Recursively.**
    - Energy(**X**) $= 1$ if **X** is Leaf of Color = 0.
    - Energy(**X**) $= 2$ if **X** is Leaf of Color = 1.
    - **X Not Leaf: Energy**(**X**) = **Maximum** of:
        - 1 + Energy of any Child of Opposite Color.
        - **2 + Energy** of any **Child of Matching Color**.

Energy

- Energy(**X**): Positive Integer for **X** of Output = 1.
- Defined Recursively.
  - Energy(**X**) $= 1$ if **X** is Leaf of Color = 0.
  - Energy(**X**) $= 2$ if **X** is Leaf of Color = 1.
  - **X** Not Leaf: Energy(**X**) = Maximum of:
    - 1 + Energy of any Child of Opposite Color.
    - 2 + Energy of any Child of Matching Color.
- **Theorem:** No Action of a **Process of Input = 0** can **Increase Maximum Energy** of the Network.

Energy

- Energy(**X**): Positive Integer for **X** of Output = 1.
- Defined Recursively.
  - Energy(**X**) $= 1$ if **X** is Leaf of Color = 0.
  - Energy(**X**) $= 2$ if **X** is Leaf of Color = 1.
  - **X** Not Leaf: Energy(**X**) = Maximum of:
    - 1 + Energy of any Child of Opposite Color.
    - 2 + Energy of any Child of Matching Color.
- Theorem: No Action of a Process of Input = 0 can Increase Maximum Energy of the Network.
- **Theorem:** If **All Processes have Input = 0:** **Maximum Energy** of the Network Decreases every **Round.** Hence **Convergence** After $O(n)$ **Rounds.**

Energy

- Energy(**X**): Positive Integer for **X** of Output = 1.
- Defined Recursively.
  - Energy(**X**) $= 1$ if **X** is Leaf of Color = 0.
  - Energy(**X**) $= 2$ if **X** is Leaf of Color = 1.
  - **X** Not Leaf: Energy(**X**) = Maximum of:
    - 1 + Energy of any Child of Opposite Color.
    - 2 + Energy of any Child of Matching Color.
- Theorem: No Action of a Process of Input = 0 can
  Increase Maximum Energy of the Network.
- Theorem: If All Processes have Input = 0:
  Maximum Energy of the Network Decreases every Round,
  Hence Convergence After $O(n)$ Rounds.

Silence

XIV

Silence

- **Legitimate Configuration** After Finitely Many **Rounds:** $O(Diam)$ if **Output = 0**, $O(n)$ if **Output = 1.**

Silence

- Legitimate Configuration After Finitely Many Rounds:
  $O(Diam)$ if Output = 0, $O(n)$ if Output = 1.
- How do we Stop **Color Waves** from **Continuing Forever?**

Silence

- Legitimate Configuration After Finitely Many Rounds:
  $O(Diam)$ if Output = 0, $O(n)$ if Output = 1.
- How do we Stop Color Waves from Continuing Forever?
- **Done Waves:**

Silence

- Legitimate Configuration After Finitely Many Rounds:
  $O(Diam)$ if Output = 0, $O(n)$ if Output = 1.
- How do we Stop Color Waves from Continuing Forever?
- **Done Waves:**
  - **Leaf** Initiates when it **Detects** (Local) **Legitimacy.**

Silence

- Legitimate Configuration After Finitely Many Rounds:
  $O(Diam)$ if Output = 0, $O(n)$ if Output = 1.
- How do we Stop Color Waves from Continuing Forever?
- **Done Waves:**
  - Leaf Initiates when it Detects (Local) Legitimacy.
  - Done Wave **Convergecast.**

Silence

- Legitimate Configuration After Finitely Many Rounds: $O(Diam)$ if Output = 0, $O(n)$ if Output = 1.
- How do we Stop Color Waves from Continuing Forever?
- **Done Waves:**
  - Leaf Initiates when it Detects (Local) Legitimacy.
  - Done Wave Convergecast.
  - **Clusterhead** (That is, Process with Input = 1) Becomes **Color Frozen.** Cannot Change Color.

Silence

- Legitimate Configuration After Finitely Many Rounds:
  $O(Diam)$ if Output = 0, $O(n)$ if Output = 1.
- How do we Stop Color Waves from Continuing Forever?
- **Done Waves:**
  - Leaf Initiates when it Detects (Local) Legitimacy.
  - Done Wave Convergecast.
  - Clusterhead (That is, Process with Input = 1) Becomes Color Frozen. Cannot Change Color.
  - **Color Lock:** Within $O(Diam)$ Rounds: **Silence** is Achieved.

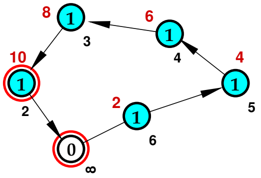
Arbitrary Start: Output = 0: Problem Solved!

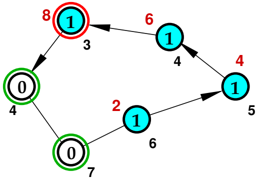
Arbitrary Start: Output = 0: Problem Solved!

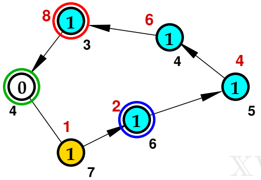
- **Asynchronous Example Computation**

Arbitrary Start: Output = 0: Problem Solved!

- **Asynchronous Example Computation**
- **All Input Bits 0.**

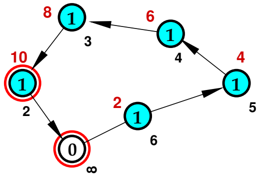


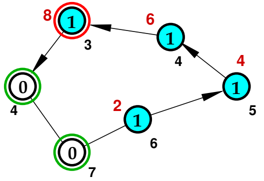


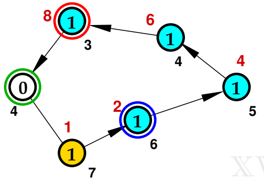


Arbitrary Start: Output = 0: Problem Solved!

- **Asynchronous Example Computation**
- All Input Bits 0.
- Red = Enabled to Reset.

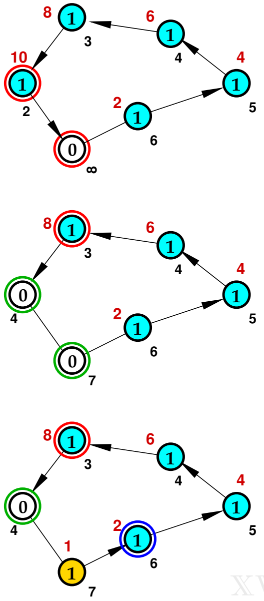






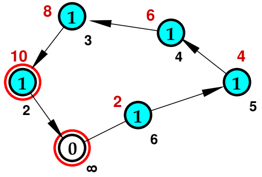
Arbitrary Start: Output = 0: Problem Solved!

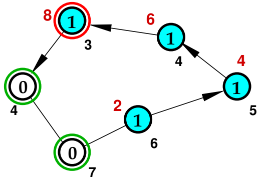
- **Asynchronous Example Computation**
- All Input Bits 0.
- Red = Enabled to Reset.
- Green = Enabled to Join.

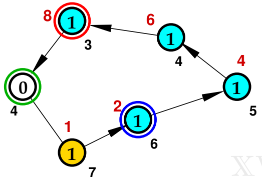


Arbitrary Start: Output = 0: Problem Solved!

- **Asynchronous Example Computation**
- All Input Bits 0.
- Red = Enabled to Reset.
- Green = Enabled to Join.
- Blue = Enabled to Change Color.

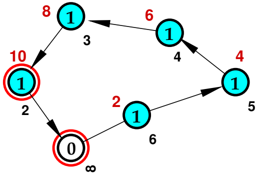


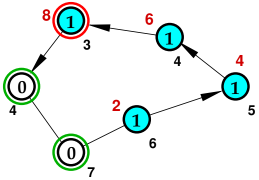


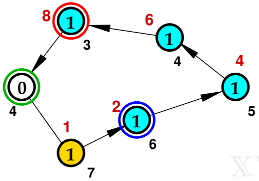


Arbitrary Start: Output = 0: Problem Solved!

- **Asynchronous Example Computation**
- All Input Bits 0.
- Red = Enabled to Reset.
- Green = Enabled to Join.
- Blue = Enabled to Change Color.
- Maximum **Energy** Initially = 10.

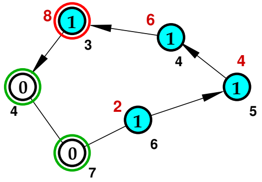


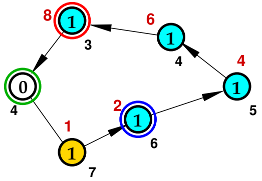


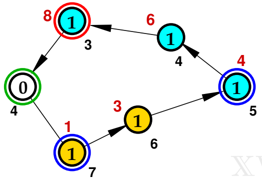


Arbitrary Start: Output = 0: Problem Solved!

- **Asynchronous Example Computation**
- All Input Bits 0.
- Red = Enabled to Reset.
- Green = Enabled to Join.
- Blue = Enabled to Change Color.
- Maximum **Energy** Initially = 10.

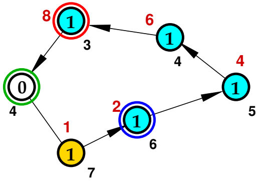


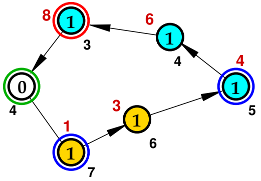


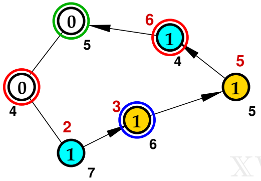


Arbitrary Start: Output = 0: Problem Solved!

- **Asynchronous Example Computation**
- All Input Bits 0.
- Red = Enabled to Reset.
- Green = Enabled to Join.
- Blue = Enabled to Change Color.
- Maximum Energy Initially = 10.
- Maximum **Energy Decreases** each **Round.**

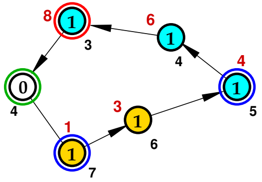
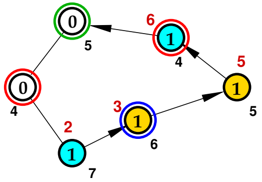
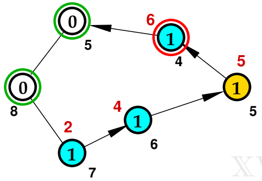






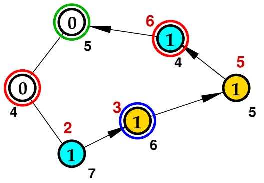
Arbitrary Start: Output = 0: Problem Solved!

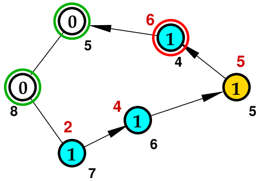
- **Asynchronous Example Computation**
- All Input Bits 0.
- Red = Enabled to Reset.
- Green = Enabled to Join.
- Blue = Enabled to Change Color.
- Maximum Energy Initially = 10.
- Maximum **Energy Decreases** each **Round.**

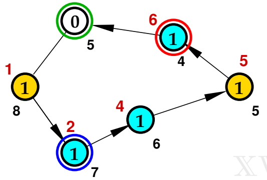




Arbitrary Start: Output = 0: Problem Solved!

- **Asynchronous Example Computation**
- All Input Bits 0.
- Red = Enabled to Reset.
- Green = Enabled to Join.
- Blue = Enabled to Change Color.
- Maximum Energy Initially = 10.
- Maximum **Energy Decreases** each **Round.**

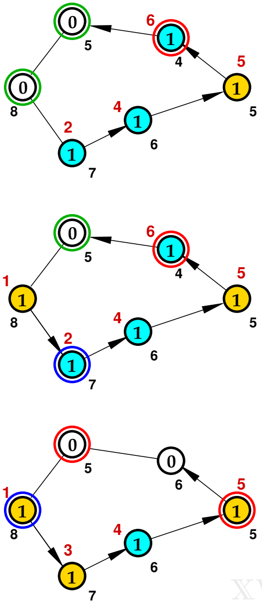






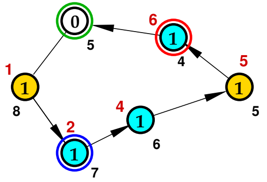
Arbitrary Start: Output = 0: Problem Solved!

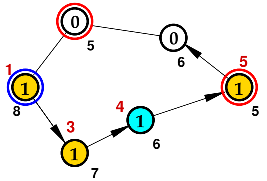
- **Asynchronous Example Computation**
- All Input Bits 0.
- Red = Enabled to Reset.
- Green = Enabled to Join.
- Blue = Enabled to Change Color.
- Maximum Energy Initially = 10.
- Maximum **Energy Decreases** each **Round.**

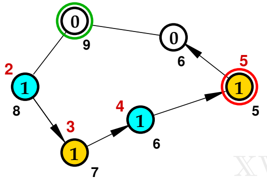


Arbitrary Start: Output = 0: Problem Solved!

- **Asynchronous Example Computation**
- All Input Bits 0.
- Red = Enabled to Reset.
- Green = Enabled to Join.
- Blue = Enabled to Change Color.
- Maximum Energy Initially = 10.
- Maximum **Energy Decreases** each **Round.**

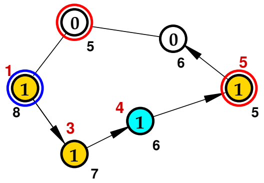


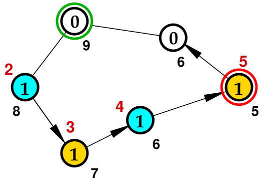


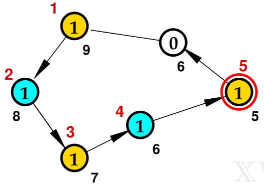


Arbitrary Start: Output = 0: Problem Solved!

- **Asynchronous Example Computation**
- All Input Bits 0.
- Red = Enabled to Reset.
- Green = Enabled to Join.
- Blue = Enabled to Change Color.
- Maximum Energy Initially = 10.
- Maximum Energy Decreases each Round.
- **No Further Growth Possible.**

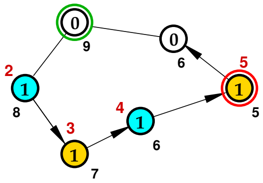


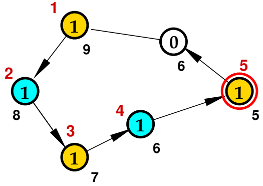


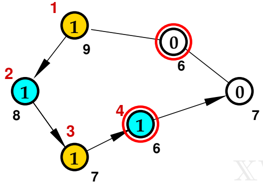


Arbitrary Start: Output = 0: Problem Solved!

- **Asynchronous Example Computation**
- All Input Bits 0.
- Red = Enabled to Reset.
- Green = Enabled to Join.
- Blue = Enabled to Change Color.
- Maximum Energy Initially = 10.
- Maximum Energy Decreases each Round.
- **No Further Growth Possible.**

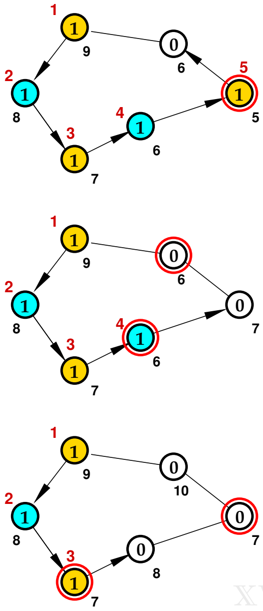






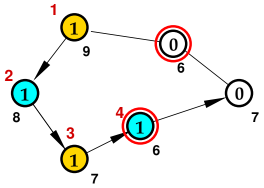
Arbitrary Start: Output = 0: Problem Solved!

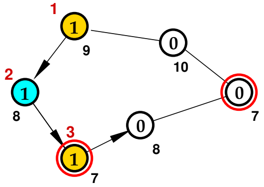
- **Asynchronous Example Computation**
- All Input Bits 0.
- Red = Enabled to Reset.
- Green = Enabled to Join.
- Blue = Enabled to Change Color.
- Maximum Energy Initially = 10.
- Maximum Energy Decreases each Round.
- **No Further Growth Possible.**

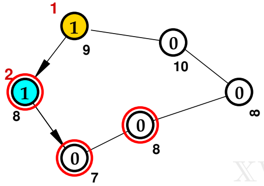


Arbitrary Start: Output = 0: Problem Solved!

- **Asynchronous Example Computation**
- All Input Bits 0.
- Red = Enabled to Reset.
- Green = Enabled to Join.
- Blue = Enabled to Change Color.
- Maximum Energy Initially = 10.
- Maximum Energy Decreases each Round.
- **No Further Growth Possible.**

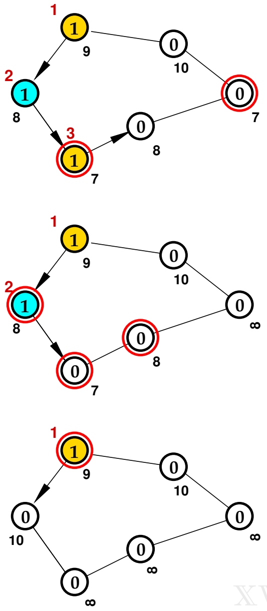






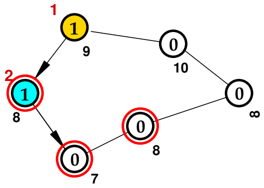
Arbitrary Start: Output = 0: Problem Solved!

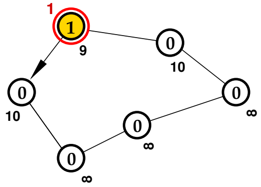
- **Asynchronous Example Computation**
- All Input Bits 0.
- Red = Enabled to Reset.
- Green = Enabled to Join.
- Blue = Enabled to Change Color.
- Maximum Energy Initially = 10.
- Maximum Energy Decreases each Round.
- **No Further Growth Possible.**

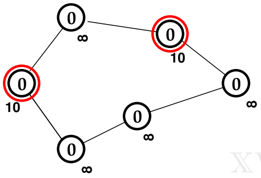


Arbitrary Start: Output = 0: Problem Solved!

- **Asynchronous Example Computation**
- All Input Bits 0.
- Red = Enabled to Reset.
- Green = Enabled to Join.
- Blue = Enabled to Change Color.
- Maximum Energy Initially = 10.
- Maximum Energy Decreases each Round.
- No Further Growth Possible.
- Configuration is Now **Legitimate.**

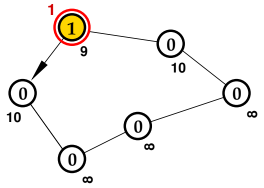


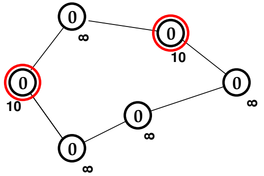


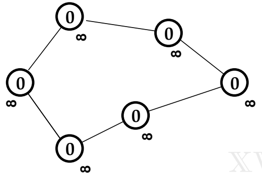


Arbitrary Start: Output = 0: Problem Solved!

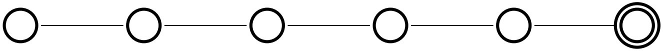
- **Asynchronous Example Computation**
- All Input Bits 0.
- Red = Enabled to Reset.
- Green = Enabled to Join.
- Blue = Enabled to Change Color.
- Maximum Energy Initially = 10.
- Maximum Energy Decreases each Round.
- No Further Growth Possible.
- Configuration is Now Legitimate.
- Configuration is Now **Final. Silent.**



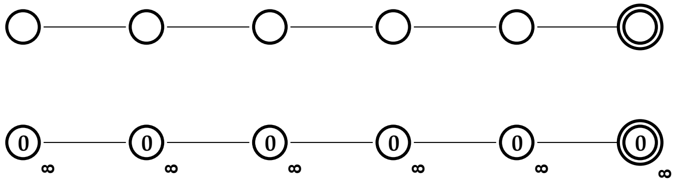




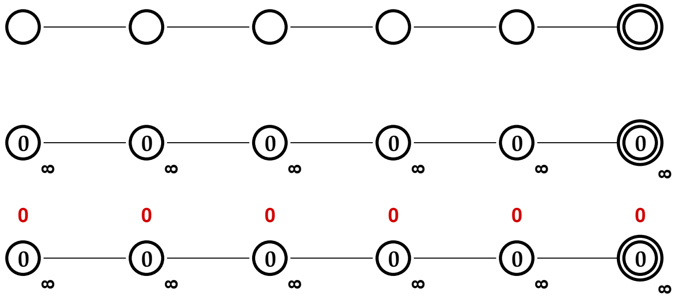
- **Chain Example. Double Circle:** Input Bit = 1, Otherwise 0.



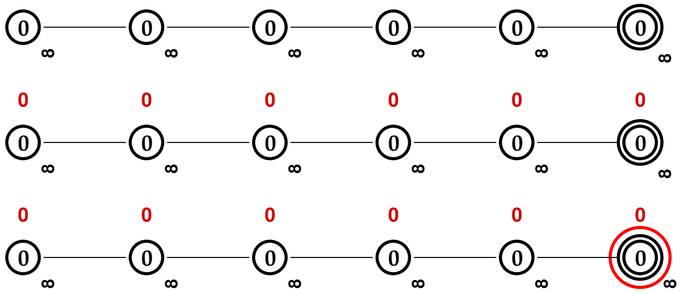
- **Chain Example. Double Circle:** Input Bit = 1, Otherwise 0.
- **Output Bit:** Inside Circle. **Black Numeral:** Level.



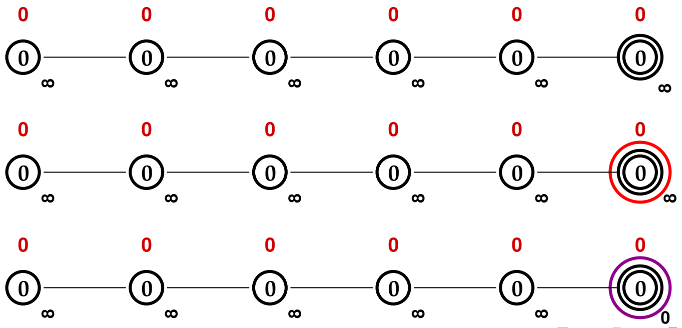
- **Chain Example. Double Circle:** Input Bit = 1, Otherwise 0.
- **Output Bit:** Inside Circle. **Black Numeral:** Level.
- **Red Numeral:** Energy.



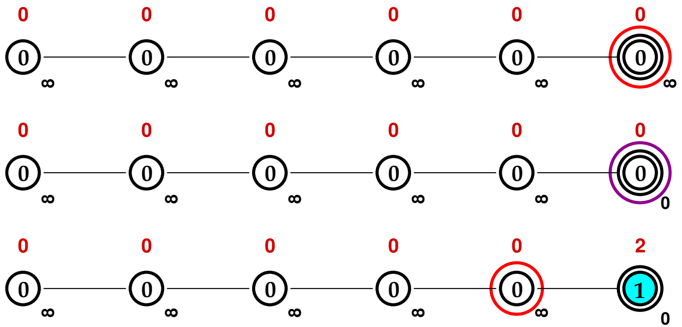
- **Chain Example. Double Circle:** Input Bit = 1, Otherwise 0.
- **Output Bit:** Inside Circle. **Black Numeral:** Level.
- **Red Numeral:** Energy.
- **Red Circle:** Enabled to Reset.



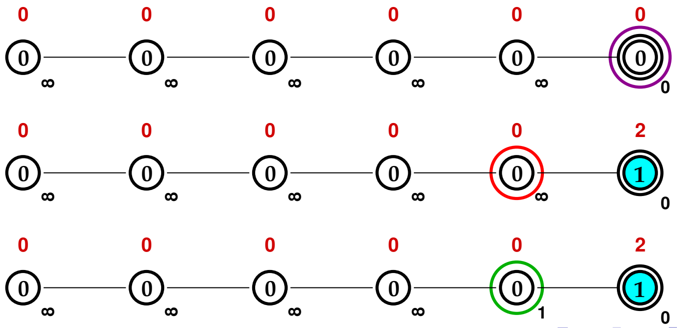
- **Chain Example. Double Circle:** Input Bit = 1, Otherwise 0.
- **Output Bit:** Inside Circle. **Black Numeral:** Level.
- **Red Numeral:** Energy.
- **Red Circle:** Enabled to Reset.
- **Magenta Circle:** Enabled to Initialize.



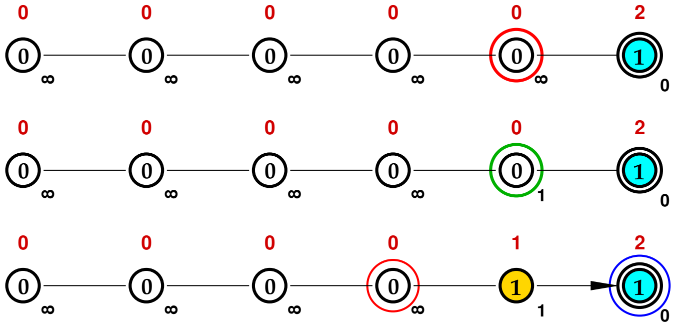
- **Chain Example. Double Circle:** Input Bit = 1, Otherwise 0.
- **Output Bit:** Inside Circle. **Black Numeral:** Level.
- **Red Numeral:** Energy.
- **Red Circle:** Enabled to Reset.
- **Magenta Circle:** Enabled to Initialize.
- **Cyan:** Color = 1.



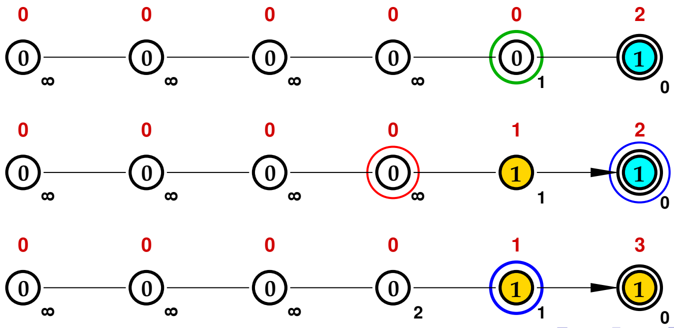
- **Chain Example. Double Circle:** Input Bit = 1, Otherwise 0.
- **Output Bit:** Inside Circle. **Black Numeral:** Level.
- **Red Numeral:** Energy.
- **Red Circle:** Enabled to Reset.
- **Magenta Circle:** Enabled to Initialize.
- **Cyan:** Color = 1.
- **Green Circle:** Enabled to Join. Can only attach to Color 1.



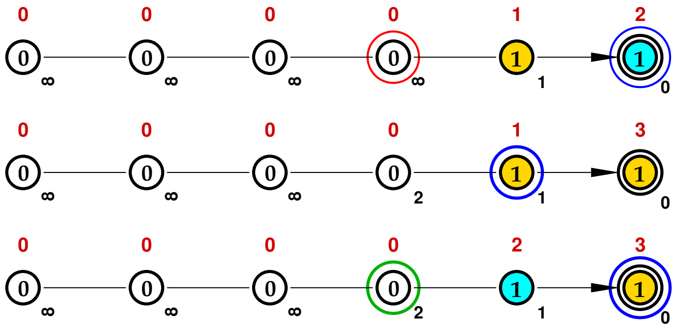
- **Chain Example. Double Circle:** Input Bit = 1, Otherwise 0.
- **Output Bit:** Inside Circle. **Black Numeral:** Level.
- **Red Numeral:** Energy.
- **Red Circle:** Enabled to Reset.
- **Cyan:** Color = 1; **Gold:** Color = 0.
- **Green Circle:** Enabled to Join. Can only attach to Color 1.
- **Blue Circle:** Enabled to Change Color.



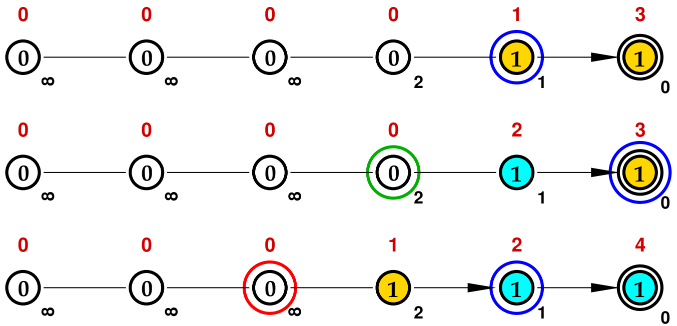
- **Chain Example.** **Double Circle:** Input Bit = 1, Otherwise 0.
- **Red Numeral:** Energy.
- **Red Circle:** Enabled to Reset.
- **Cyan:** Color = 1; **Gold:** Color = 0.
- **Green Circle:** Enabled to Join. Can only attach to Color 1.
- **Blue Circle:** Enabled to Change Color.
- **Color Waves** Convergecast. **Absorbed** by Clusterhead.



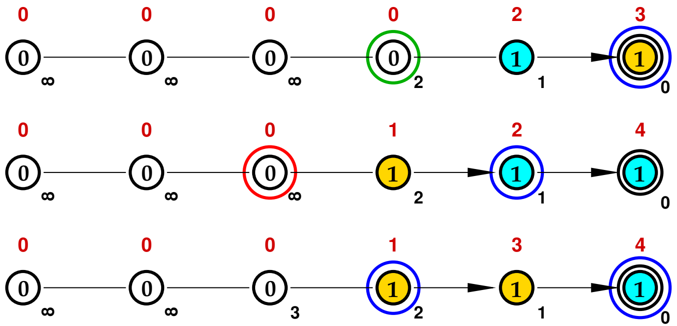
- **Chain Example. Double Circle:** Input Bit = 1, Otherwise 0.
- **Red Numeral:** Energy.
- **Red Circle:** Enabled to Reset.
- **Cyan:** Color = 1; **Gold:** Color = 0.
- **Green Circle:** Enabled to Join. Can only attach to Color 1.
- **Blue Circle:** Enabled to Change Color.
- **Color Waves** Convergecast. **Absorbed** by Clusterhead.



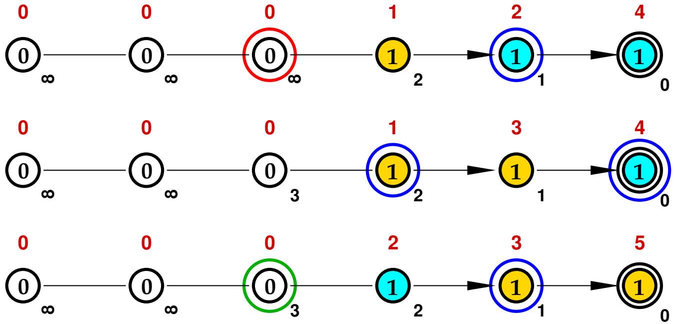
- **Chain Example. Double Circle:** Input Bit = 1, Otherwise 0.
- **Red Numeral:** Energy.
- **Red Circle:** Enabled to Reset.
- **Cyan:** Color = 1; **Gold:** Color = 0.
- **Green Circle:** Enabled to Join. Can only attach to Color 1.
- **Blue Circle:** Enabled to Change Color.
- **Color Waves** Convergecast. **Absorbed** by Clusterhead.



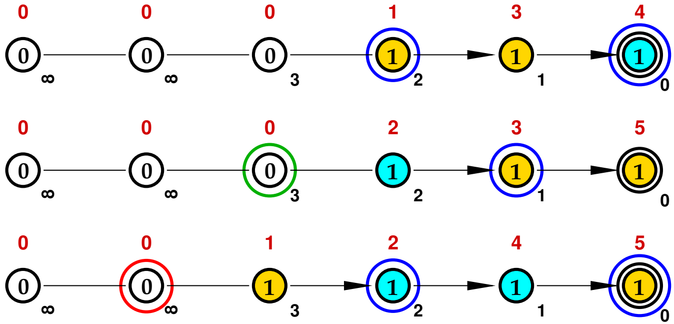
- **Chain Example.** **Double Circle:** Input Bit = 1, Otherwise 0.
- **Red Numeral:** Energy.
- **Red Circle:** Enabled to Reset.
- **Cyan:** Color = 1; **Gold:** Color = 0.
- **Green Circle:** Enabled to Join. Can only attach to Color 1.
- **Blue Circle:** Enabled to Change Color.
- **Color Waves** Convergecast. **Absorbed** by Clusterhead.



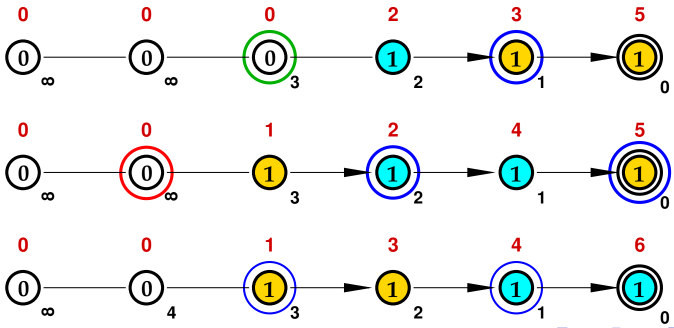
- **Chain Example. Double Circle:** Input Bit = 1, Otherwise 0.
- **Red Numeral:** Energy.
- **Red Circle:** Enabled to Reset.
- **Cyan:** Color = 1; **Gold:** Color = 0.
- **Green Circle:** Enabled to Join. Can only attach to Color 1.
- **Blue Circle:** Enabled to Change Color.
- **Color Waves** Convergecast. **Absorbed** by Clusterhead.



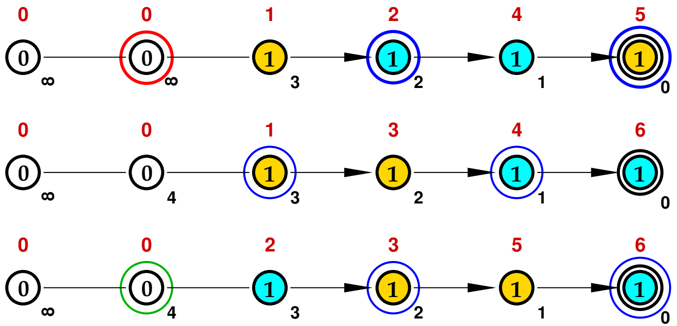
- **Chain Example. Double Circle:** Input Bit = 1, Otherwise 0.
- **Red Numeral:** Energy.
- **Red Circle:** Enabled to Reset.
- **Cyan:** Color = 1; **Gold:** Color = 0.
- **Green Circle:** Enabled to Join. Can only attach to Color 1.
- **Blue Circle:** Enabled to Change Color.
- **Color Waves** Convergecast. **Absorbed** by Clusterhead.



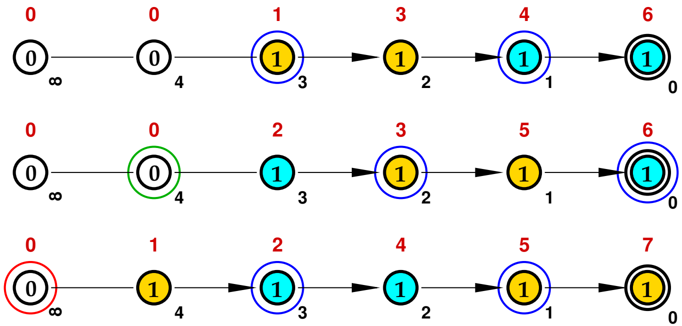
- **Chain Example. Double Circle:** Input Bit = 1, Otherwise 0.
- **Red Numeral:** Energy.
- **Red Circle:** Enabled to Reset.
- **Cyan:** Color = 1; **Gold:** Color = 0.
- **Green Circle:** Enabled to Join. Can only attach to Color 1.
- **Blue Circle:** Enabled to Change Color.
- **Color Waves** Convergecast. **Absorbed** by Clusterhead.



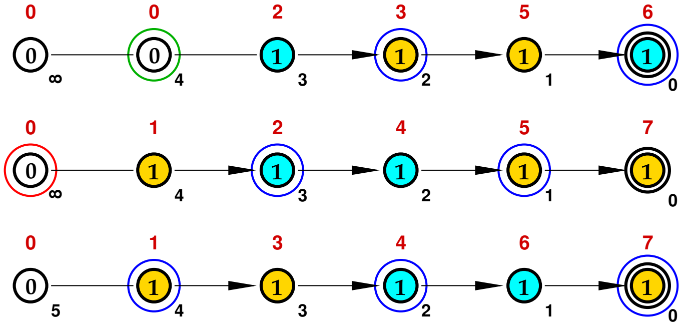
- **Chain Example. Double Circle:** Input Bit = 1, Otherwise 0.
- **Red Numeral:** Energy.
- **Red Circle:** Enabled to Reset.
- **Cyan:** Color = 1; **Gold:** Color = 0.
- **Green Circle:** Enabled to Join. Can only attach to Color 1.
- **Blue Circle:** Enabled to Change Color.
- **Color Waves** Convergecast. **Absorbed** by Clusterhead.



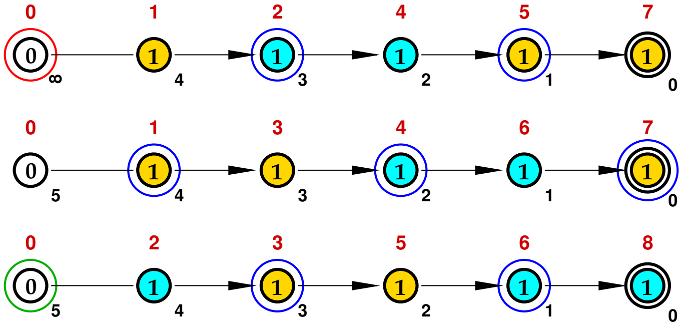
- **Chain Example. Double Circle:** Input Bit = 1, Otherwise 0.
- **Red Numeral:** Energy.
- **Red Circle:** Enabled to Reset.
- **Cyan:** Color = 1; **Gold:** Color = 0.
- **Green Circle:** Enabled to Join. Can only attach to Color 1.
- **Blue Circle:** Enabled to Change Color.
- **Color Waves** Convergecast. **Absorbed** by Clusterhead.



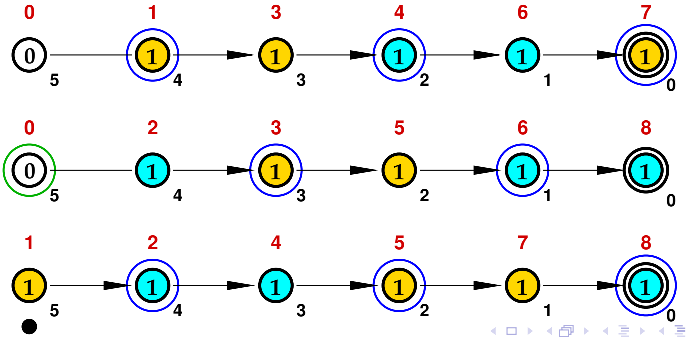
- **Chain Example. Double Circle:** Input Bit = 1, Otherwise 0.
- **Red Numeral:** Energy.
- **Red Circle:** Enabled to Reset.
- **Cyan:** Color = 1; **Gold:** Color = 0.
- **Green Circle:** Enabled to Join. Can only attach to Color 1.
- **Blue Circle:** Enabled to Change Color.
- **Color Waves** Convergecast. **Absorbed** by Clusterhead.



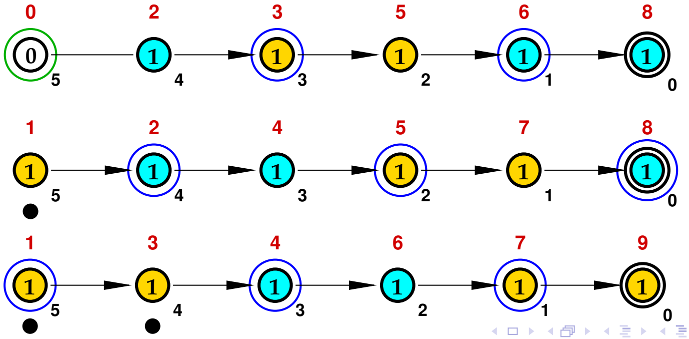
- **Chain Example. Double Circle:** Input Bit = 1, Otherwise 0.
- **Red Numeral:** Energy.
- **Cyan:** Color = 1; **Gold:** Color = 0.
- **Green Circle:** Enabled to Join. Can only attach to Color 1.
- **Blue Circle:** Enabled to Change Color.
- **Color Waves** Convergecast. **Absorbed** by Clusterhead.



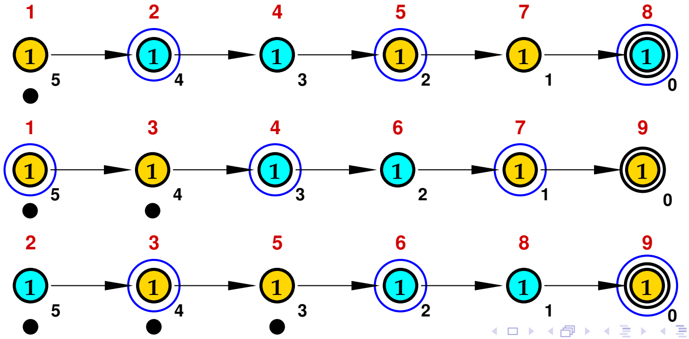
- **Chain Example. Double Circle:** Input Bit = 1, Otherwise 0.
- **Red Numeral:** Energy.
- **Cyan:** Color = 1; **Gold:** Color = 0.
- **Blue Circle:** Enabled to Change Color.
- **Color Waves** Convergecast. **Absorbed** by Clusterhead.
- Configuration is **Legitimate.** All Output Bits = 1.



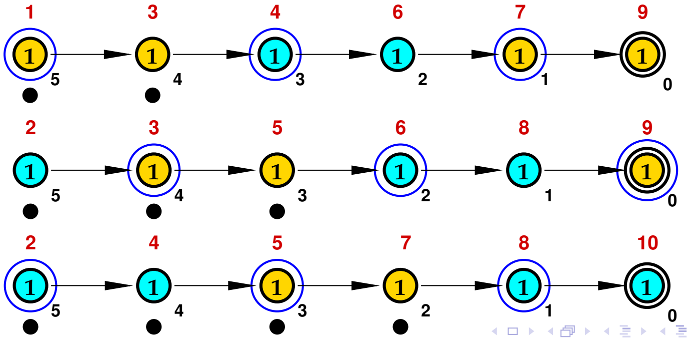
- **Chain Example. Double Circle:** Input Bit = 1, Otherwise 0.
- **Red Numeral:** Energy.
- **Cyan:** Color = 1; **Gold:** Color = 0.
- **Blue Circle:** Enabled to Change Color.
- **Color Waves** Convergecast. **Absorbed** by Clusterhead.
- Configuration is **Legitimate.** All Output Bits = 1.
- **Black Dot:** Done. Convergecast.



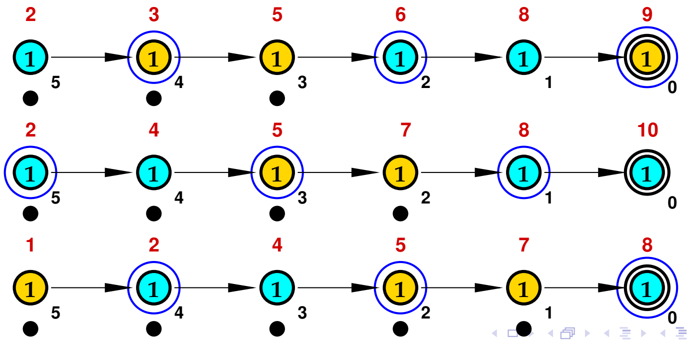
- **Chain Example. Double Circle:** Input Bit = 1, Otherwise 0.
- **Red Numeral:** Energy.
- **Cyan:** Color = 1; **Gold:** Color = 0.
- **Blue Circle:** Enabled to Change Color.
- **Color Waves** Convergecast. **Absorbed** by Clusterhead.
- Configuration is **Legitimate.** All Output Bits = 1.
- **Black Dot:** Done. Convergecast.



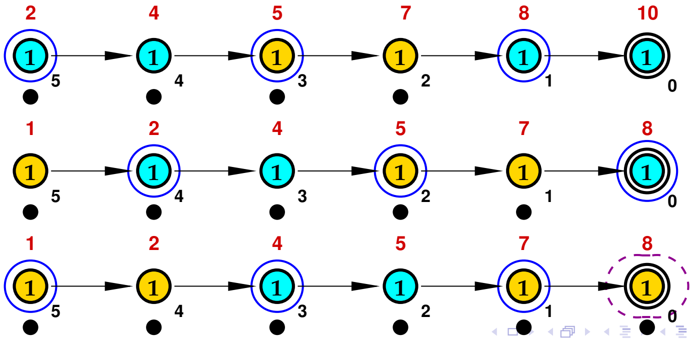
- **Chain Example. Double Circle:** Input Bit = 1, Otherwise 0.
- **Red Numeral:** Energy.
- **Cyan:** Color = 1; **Gold:** Color = 0.
- **Blue Circle:** Enabled to Change Color.
- **Color Waves** Convergecast. **Absorbed** by Clusterhead.
- Configuration is **Legitimate.** All Output Bits = 1.
- **Black Dot:** Done. Convergecast.



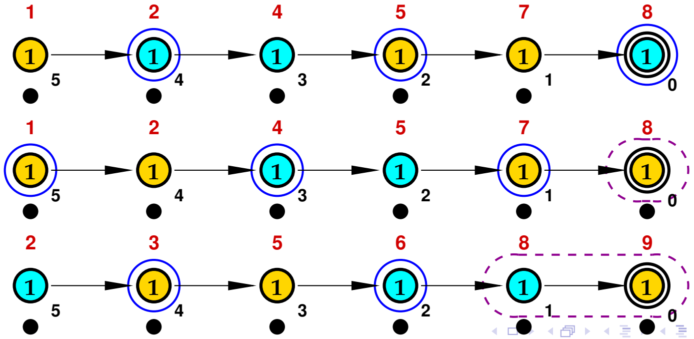
- **Chain Example. Double Circle:** Input Bit = 1, Otherwise 0.
- **Red Numeral:** Energy.
- **Cyan:** Color = 1; **Gold:** Color = 0.
- **Blue Circle:** Enabled to Change Color.
- **Color Waves** Convergecast. **Absorbed** by Clusterhead.
- Configuration is **Legitimate.** All Output Bits = 1.
- **Black Dot:** Done. Convergecast.



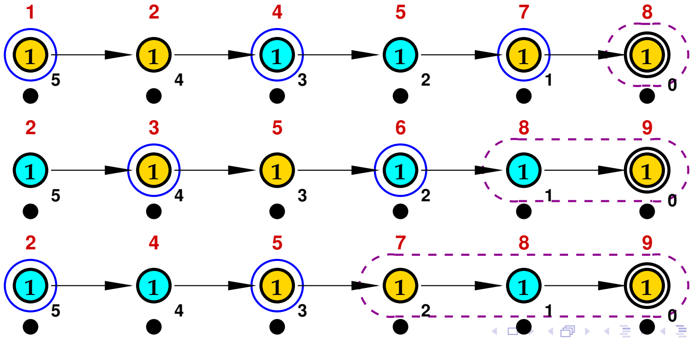
- **Chain Example. Double Circle:** Input Bit = 1, Otherwise 0.
- **Red Numeral:** Energy. No **Increase** if Clusterhead is Frozen.
- **Cyan:** Color = 1; **Gold:** Color = 0.
- Configuration is **Legitimate.** All Output Bits = 1.
- **Black Dot:** Done. Convergecast.
- Clusterhead Done, Hence Color-Frozen.
  Clusterhead will No Longer Absorb Color Waves.



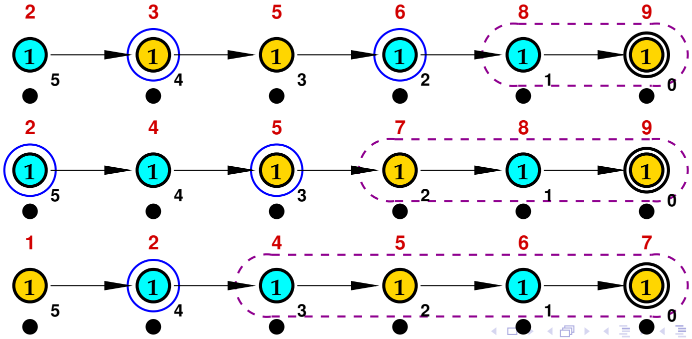
- **Chain Example.** **Double Circle:** Input Bit = 1, Otherwise 0.
- **Red Numeral:** Energy. No **Increase** if Clusterhead is Frozen.
- **Cyan:** Color = 1; **Gold:** Color = 0.
- Configuration is **Legitimate.** All Output Bits = 1.
- Clusterhead Done, Hence Color-Frozen.
  Clusterhead will No Longer Absorb Color Waves.
- **Dashed Oval:** Color-Locked Processes. Alternating Colors.



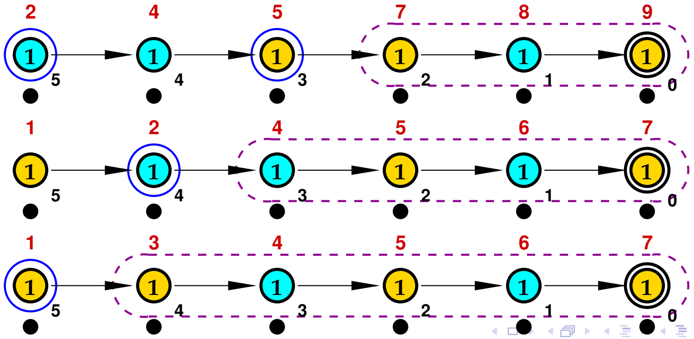
- **Chain Example. Double Circle:** Input Bit = 1, Otherwise 0.
- **Red Numeral:** Energy. No **Increase** if Clusterhead is Frozen.
- **Cyan:** Color = 1; **Gold:** Color = 0.
- Configuration is **Legitimate.** All Output Bits = 1.
- Clusterhead Done, Hence Color-Frozen.
  Clusterhead will No Longer Absorb Color Waves.
- **Dashed Oval:** Color-Locked Processes. Alternating Colors.



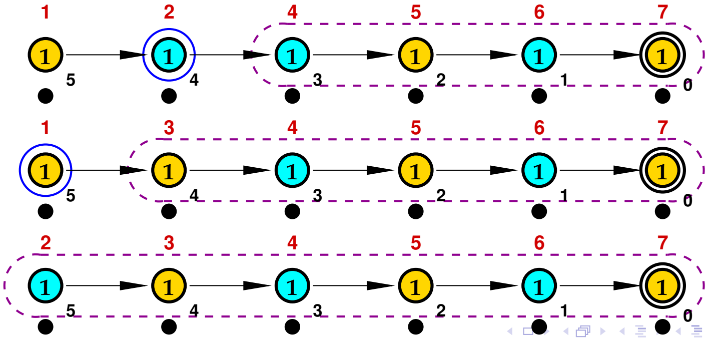
- **Chain Example. Double Circle:** Input Bit = 1, Otherwise 0.
- **Red Numeral:** Energy. No **Increase** if Clusterhead is Frozen.
- **Cyan:** Color = 1; **Gold:** Color = 0.
- Configuration is **Legitimate.** All Output Bits = 1.
- Clusterhead Done, Hence Color-Frozen.
  Clusterhead will No Longer Absorb Color Waves.
- **Dashed Oval:** Color-Locked Processes. Alternating Colors.



- **Chain Example. Double Circle:** Input Bit = 1, Otherwise 0.
- **Red Numeral:** Energy. No **Increase** if Clusterhead is Frozen.
- **Cyan:** Color = 1; **Gold:** Color = 0.
- Configuration is **Legitimate.** All Output Bits = 1.
- Clusterhead Done, Hence Color-Frozen.
  Clusterhead will No Longer Absorb Color Waves.
- **Dashed Oval:** Color-Locked Processes. Alternating Colors.



- **Chain Example. Double Circle:** Input Bit = 1, Otherwise 0.
- **Red Numeral:** Energy. No **Increase** if Clusterhead is Frozen.
- **Cyan:** Color = 1; **Gold:** Color = 0.
- Configuration is **Legitimate.** All Output Bits = 1.
- Clusterhead Done, Hence Color-Frozen.
  Clusterhead will No Longer Absorb Color Waves.
- All Processes **Color-Locked.** Final Configuration.



1 2 4 5 6 7
1 1 1 1 1 1
5 4 3 2 1 0
1 3 4 5 6 7
1 1 1 1 1 1
5 4 3 2 1 0
2 3 4 5 6 7
1 1 1 1 1 1
5 4 3 2 1 0