Simulation of Gate Circuits with Feedback in Multi-Valued Algebras

Janusz Brzozowski
University of Waterloo

Yuli Ye
University of Toronto

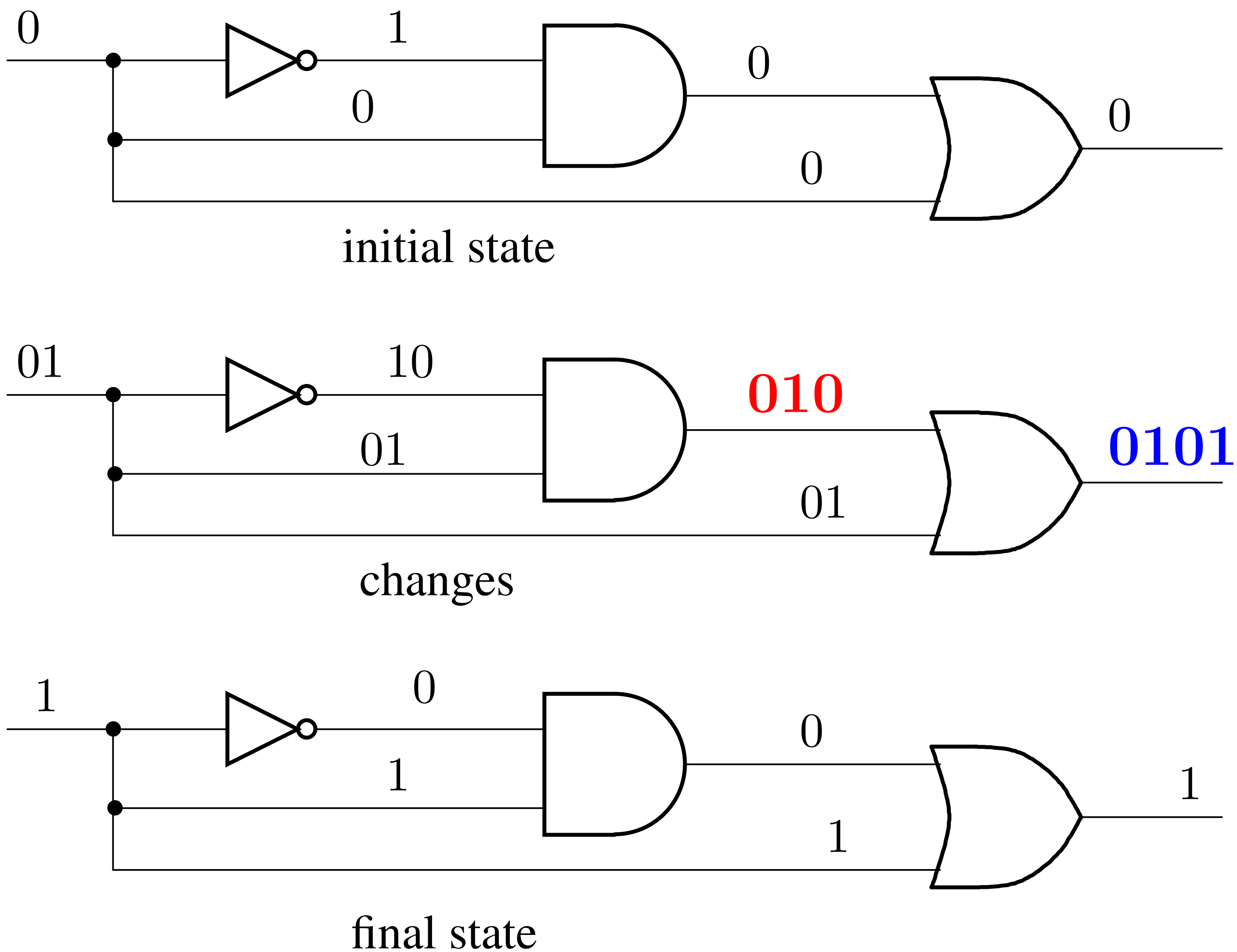
Oslo, May 15, 2007

Outline

- Efficient detection of hazards and oscillations in gate circuits
- Simulation in multivalued algebras C_2, C_3, $\ldots$ C_k, $\ldots$
  - C_2 - 3 values, C_3 - 5 values, C_k - $\ldots$ $(2k-1)$ values, $\ldots$
- Algorithm A in C_k shows worst-case **sequences of changes**
- Algorithm B in C_k shows **final values,** possibly uncertain
- Results of Algorithms A and B **detect hazards and oscillations**

Main Result: **Algorithm B in algebra $\mathbf{C_k}$, k $>$ 2, gives the same results as Algorithm B in algebra $\mathbf{C_2}$, the ternary algebra**

Hazards in Gate Circuits



Hazards: **010 - static**, **0101 - dynamic**

Infinite Algebra $C = (\mathbf{T}, \vee, \wedge, ^{-}, 0, 1)$

- Words for waveforms

1010

- Transients: $\mathbf{T} = \{0, 1, 01, 10, 010, 101, 0101, \ldots\}$

- **Gates process transients**
- **Extend gate functions:**
- INVERTER - complement each bit
  - $\overline{\mathbf{1010}} = \mathbf{0101}$

OR **and** AND **Functions**

- Transients **0 and 1:**
  - $t \vee 0 = 0 \vee t = t$
  - $t \vee 1 = 1 \vee t = 1$
- Longer transients:
  - **first letter:** OR of first letters
  - **last letter:** OR of last letters
  - **number of 0s:** sum of the **numbers of 0s** minus 1
- Examples:
  - $0 \vee 1010 = 1010$
  - $1 \vee 1010 = 1$
  - $\mathbf{0}1\mathbf{0} \vee \mathbf{1}01\mathbf{0} = \mathbf{1}0101\mathbf{0}$
- AND function is dual: count 1s instead of 0s

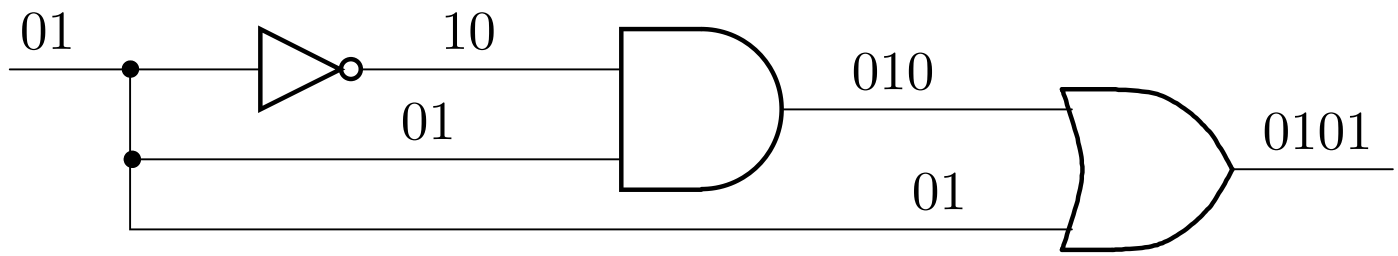
Algorithm A in Algebra C

- **Initial binary state**: $\mathbf{s}^0 = b = (b_1, \ldots, b_n)$
- **Binary inputs** initial: $\hat{a} = (\hat{a}_1, \ldots, \hat{a}_m)$ final: $a = (a_1, \ldots, a_m)$
- **Simulation input:** $\mathbf{a} = (\mathbf{a}_1, \ldots, \mathbf{a}_m) = \hat{a} \circ a$
  - $0 \circ 0 = 0 \quad 0 \circ 1 = 01 \quad 1 \circ 0 = 10 \quad 1 \circ 1 = 1$
- **Extended excitation functions:** $\mathbf{f} : \mathbf{T}^{m+n} \longrightarrow \mathbf{T}^n, \quad \mathbf{f} = (\mathbf{f}_1, \ldots, \mathbf{f}_n)$

Algorithm A

$h := 0;$
$\mathbf{a} := \hat{a} \circ a;$
$\mathbf{s}^0 := b;$
repeat
    $h := h + 1;$
    $\mathbf{s}^h := \mathbf{f}(\mathbf{a}, \mathbf{s}^{h-1});$
until $\mathbf{s}^h = \mathbf{s}^{h-1};$

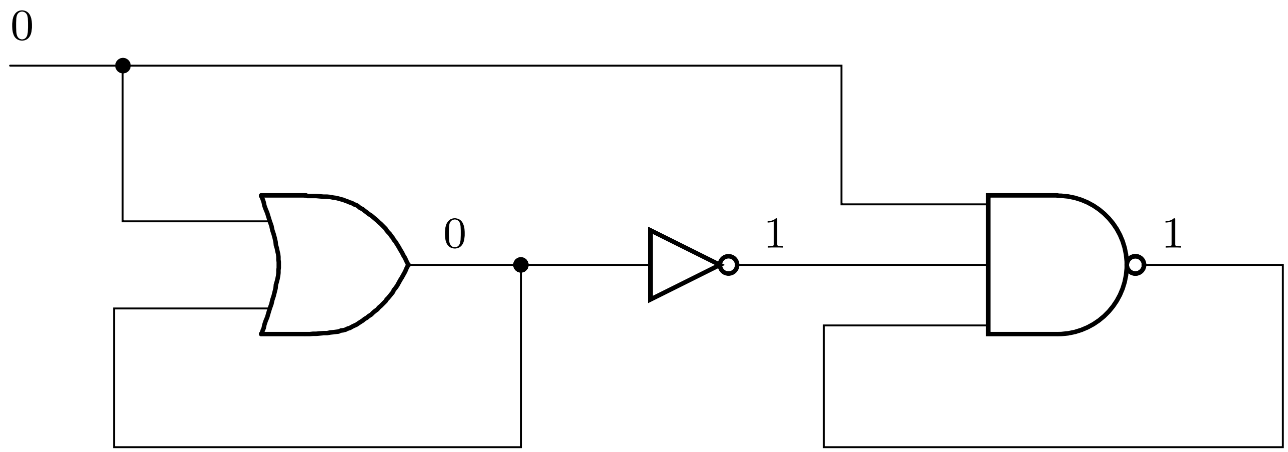
Result of Simulation of a Feedback-Free Circuit



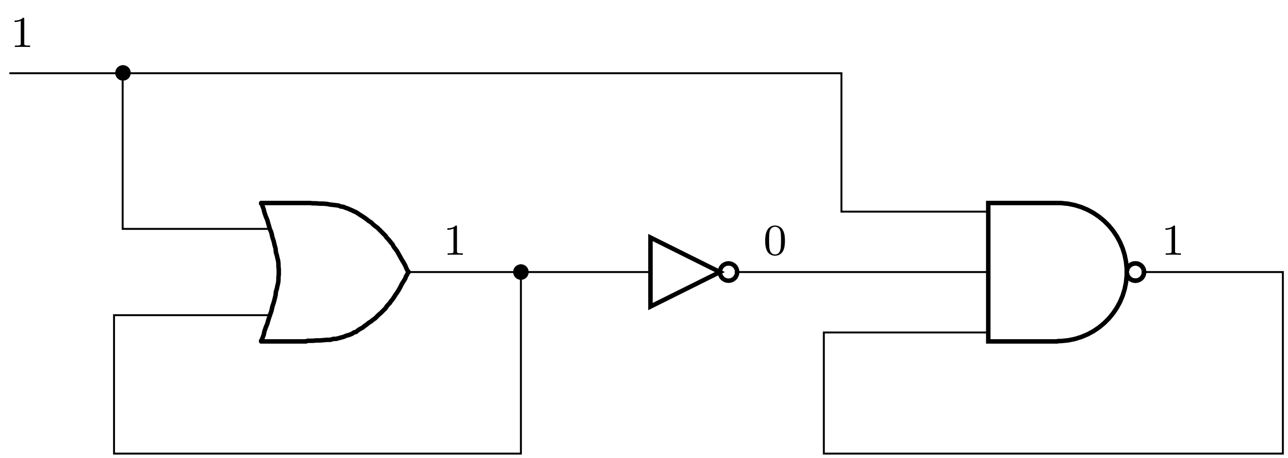
For feedback-free circuits,

Algorithm A always terminates in Algebra C

Circuit with Feedback



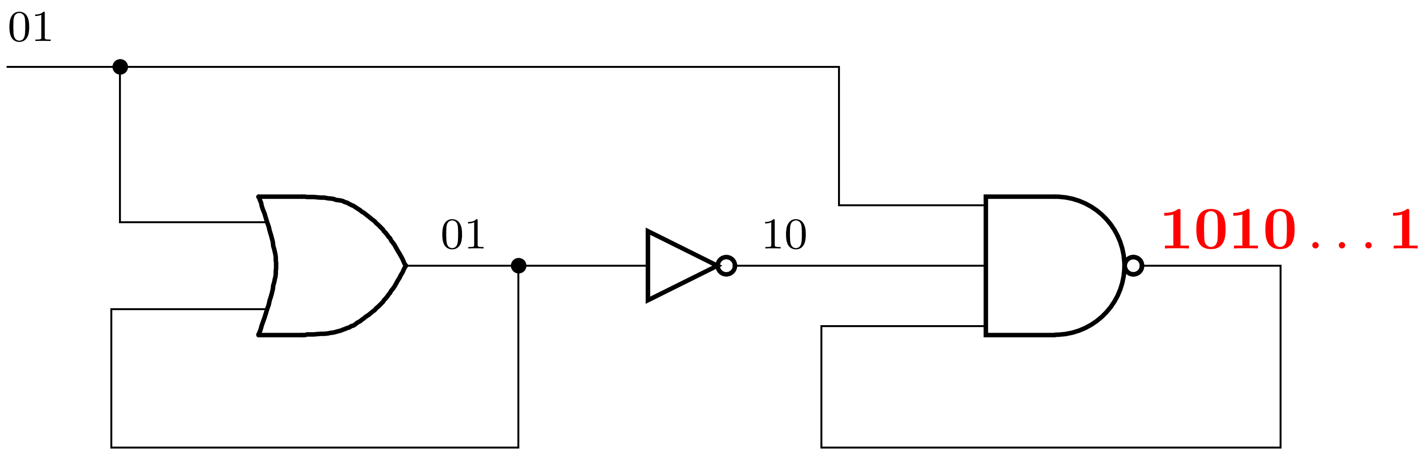
initial state



final state

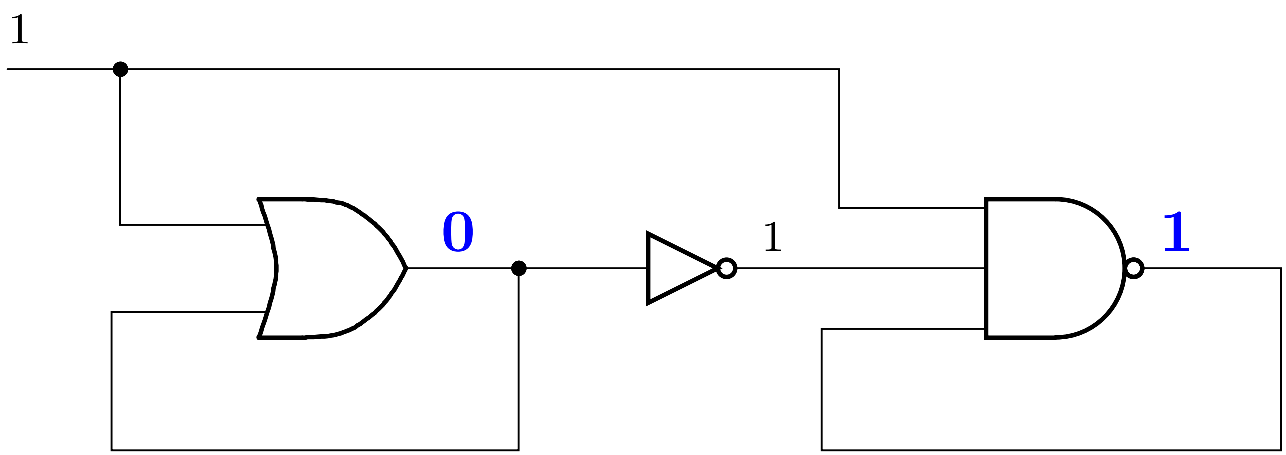
A combinational behavior

Result of Simulation in Algebra C



- **Algorithm A does not terminate**
- NAND **transient keeps growing**

A Transient Oscillation



- **Simulation does not terminate**
- NAND gate oscillates if OR gate is slow to change
- Eventually OR gate and then INVERTER change
- When INVERTER becomes 0, NAND gate becomes 1

Counting to a Threshold

- Relation $\sim_k$ in algebra $C = (\mathbf{T}, \vee, \wedge, ^-, 0, 1)$
- $t \sim_k s$ if
  - $t = s$
  - or t and s are both of length $\geq k$.
- $\sim_k$ is a congruence relation on $\mathbf{T}$
- **Quotient algebra** $C_k = (\mathbf{T}_k, \vee, \wedge, ^-, 0, 1)$
- Example: k=3

0 01

$$\Phi_3 = \{010, 101, 0101, 1010, 01010, \ldots\}$$

1 10

OR Gate in Algebra C_3

$\vee$	0	01	Φ_3	10	1
0	0	01	Φ_3	10	1
01	01	01	Φ_3	Φ_3	1
Φ_3	Φ_3	Φ_3	Φ_3	Φ_3	1
10	10	Φ_3	Φ_3	10	1
1	1	1	1	1	1

- Gate functions must be modified appropriately
- Transients of length $\geq k$ become Φ_k

Algorithms A and B in Algebra C_k

Algorithm A: same as before, but use **Algebra** C_k instead of C

$h := 0;$

$\mathbf{a} := \hat{a} \circ a;$

$\mathbf{s}^0 := b;$

repeat

  $h := h + 1;$

  $\mathbf{s}^h := \mathbf{f}(\mathbf{a}, \mathbf{s}^{h-1});$

until $\mathbf{s}^h = \mathbf{s}^{h-1};$ { **Result** $\mathbf{s}^A$ }

Algorithm B: use **final binary input** a, and **result** $\mathbf{s}^A$ of Algorithm A as initial state

$h := 0;$

$\mathbf{t}^0 := \mathbf{s}^A;$

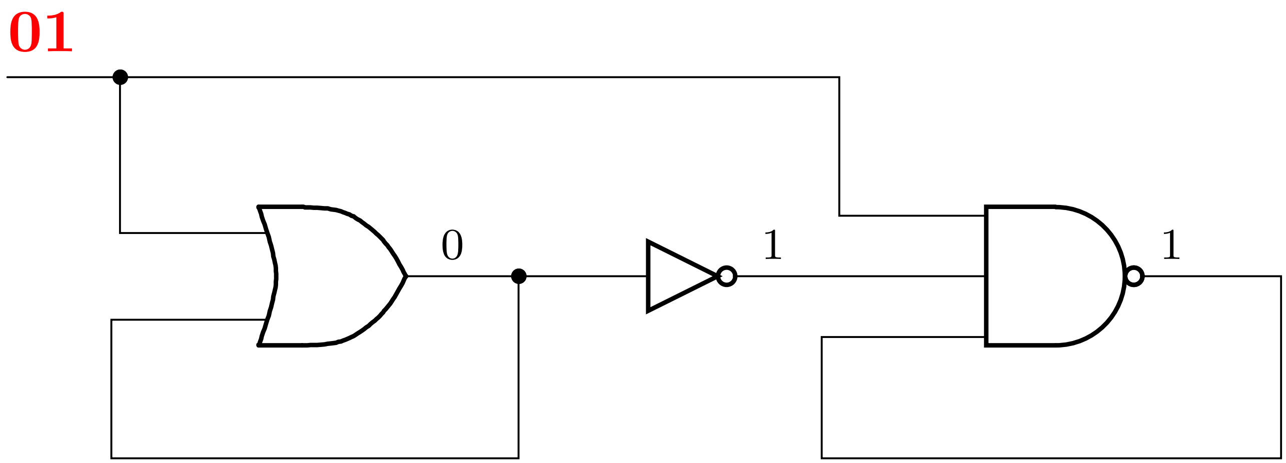
repeat

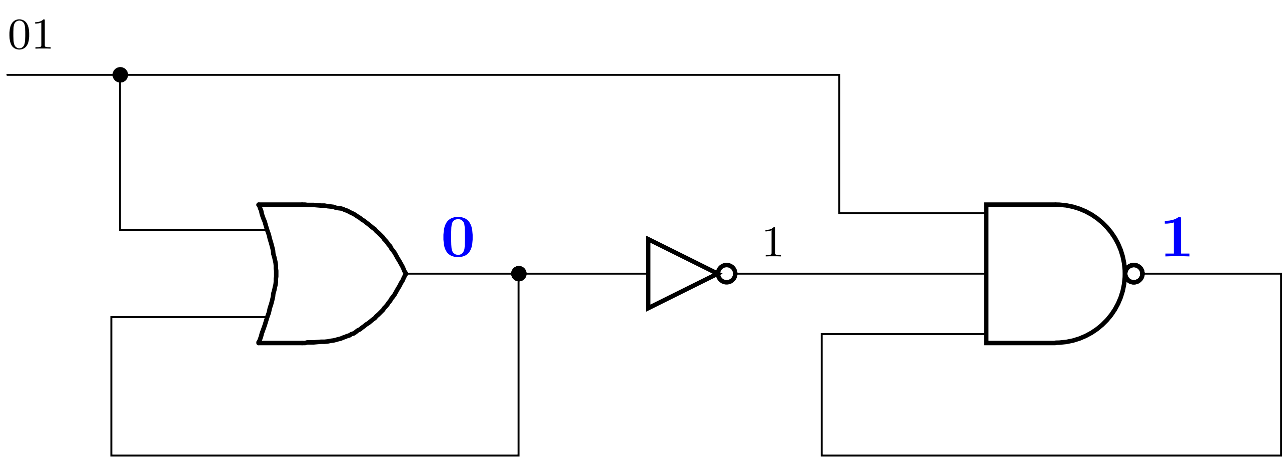
  $h := h + 1;$

  $\mathbf{t}^h := \mathbf{f}(a, \mathbf{t}^{h-1});$

until $\mathbf{t}^h = \mathbf{t}^{h-1};$ { **Result** $\mathbf{t}^B$ }

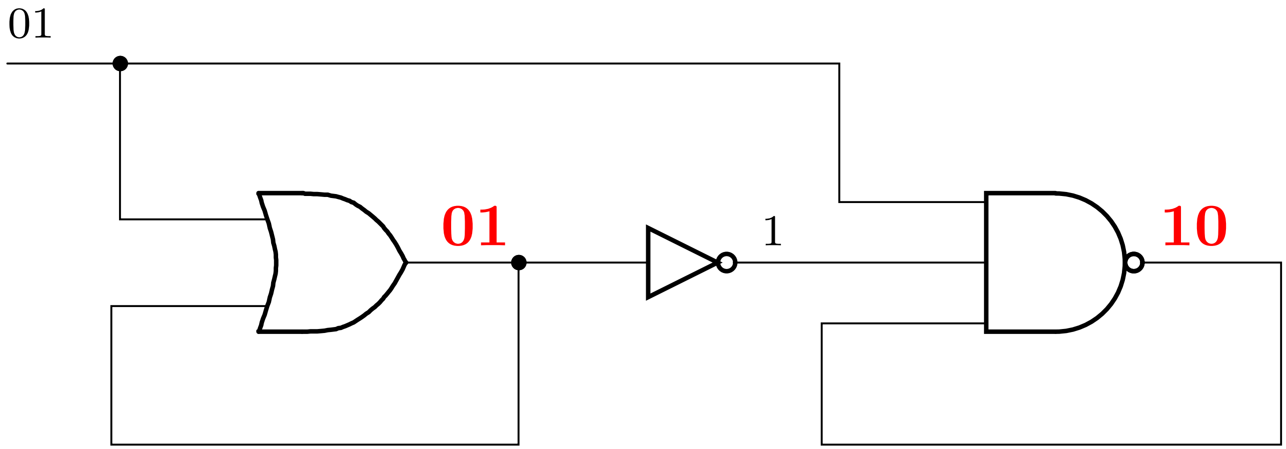
Algorithm A in Algebra C_3 - State s^0

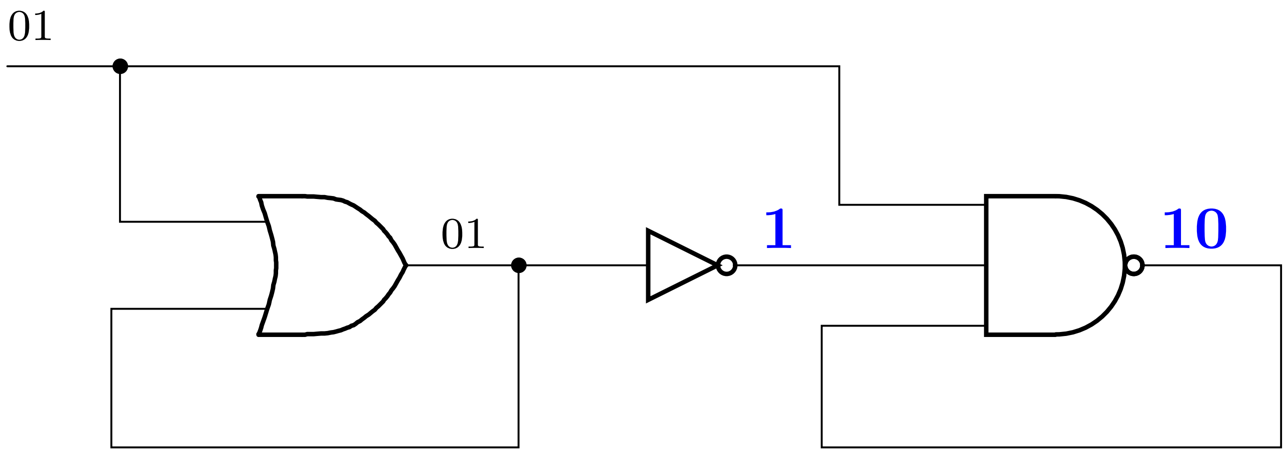




- **Change input** OR **and** NAND **unstable**

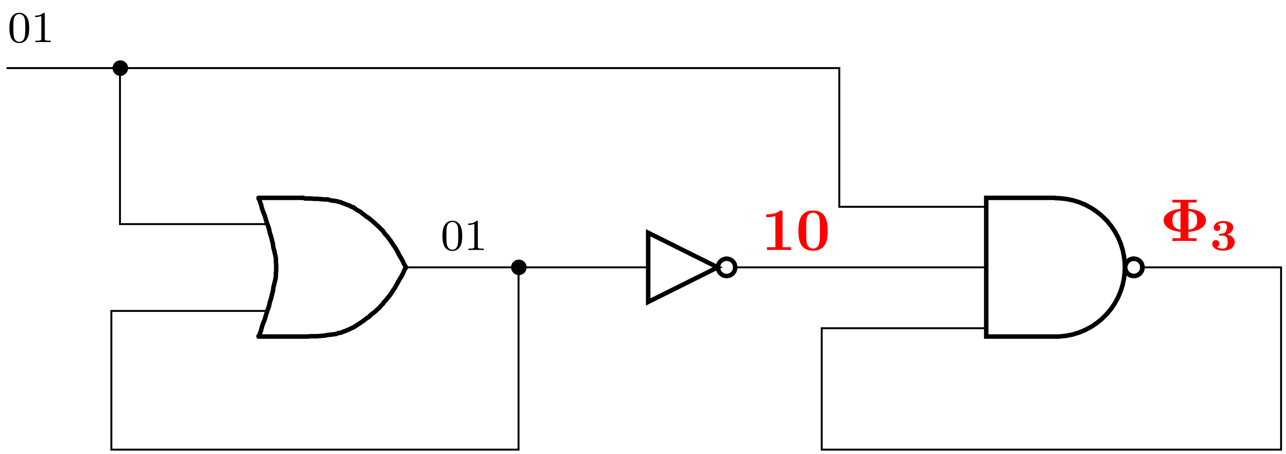
Algorithm A in Algebra C_3 - State s^1

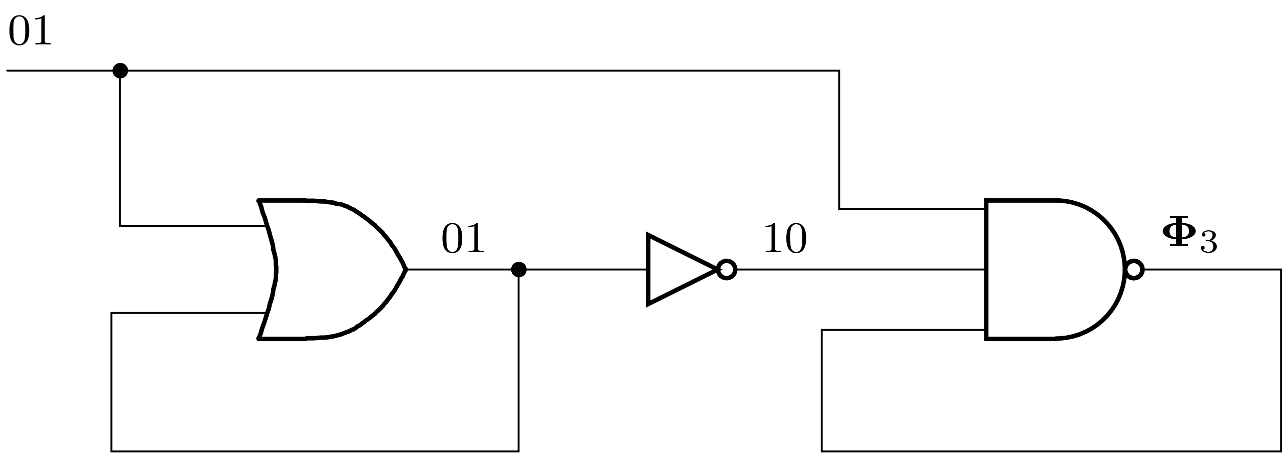




- **Change** OR **and** NAND   INVERTER **and** NAND **unstable**

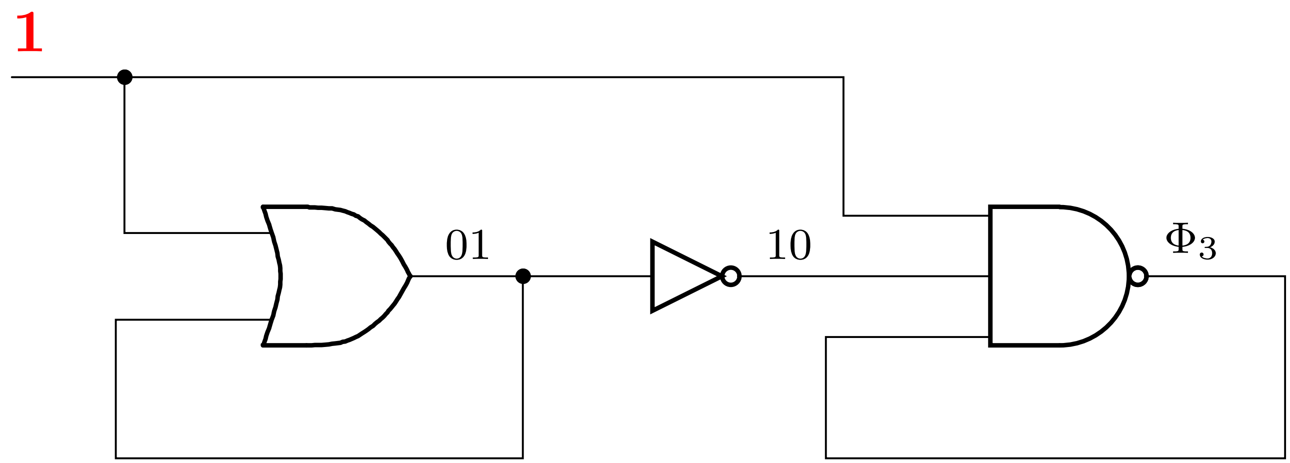
Algorithm A in Algebra C_3 - State $\mathbf{s}^2 = \mathbf{s}^A$

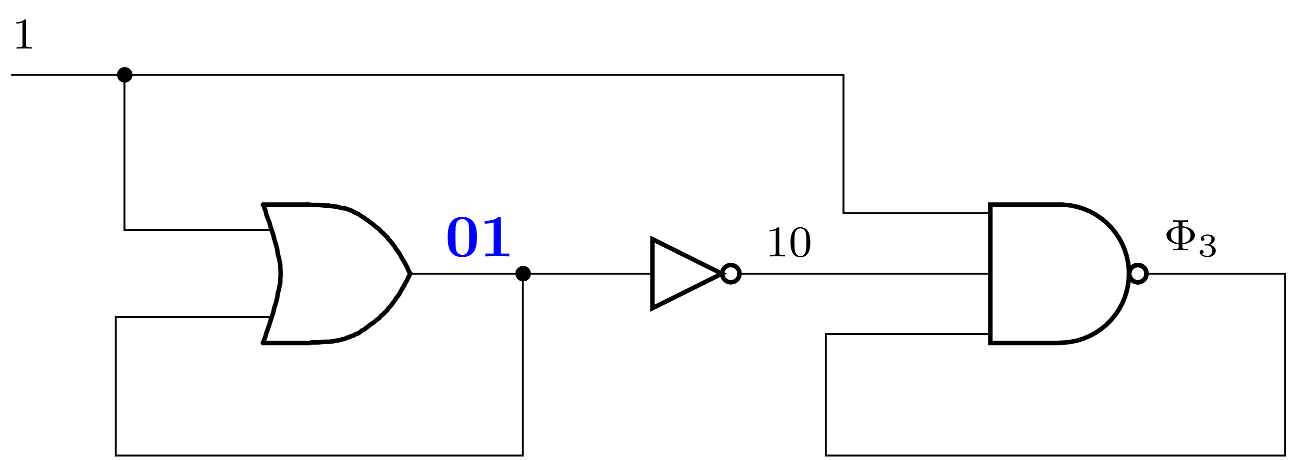




- **Change** INVERTER **and** NAND   Circuit is stable

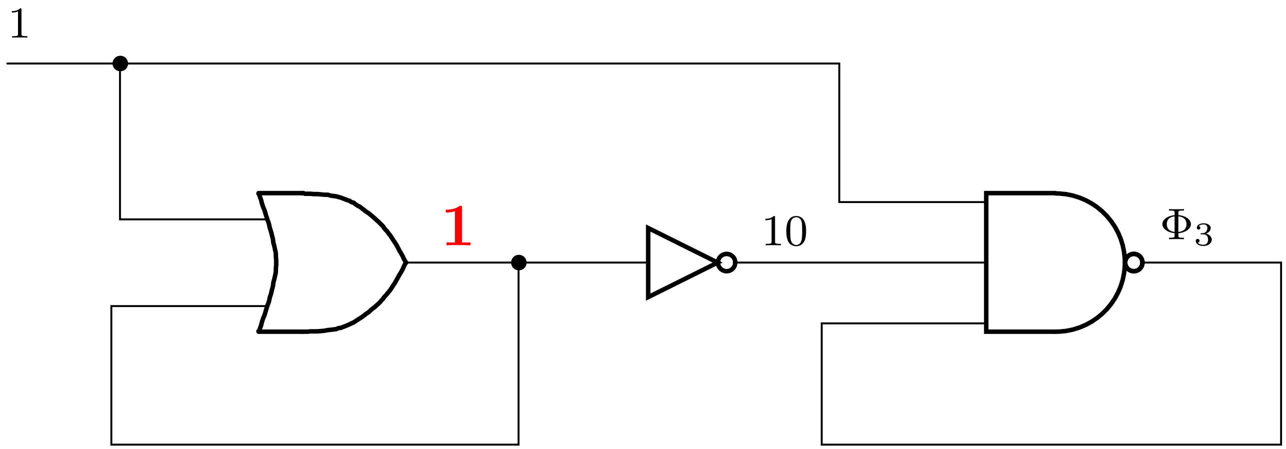
Algorithm B in Algebra C_3 - State t^0

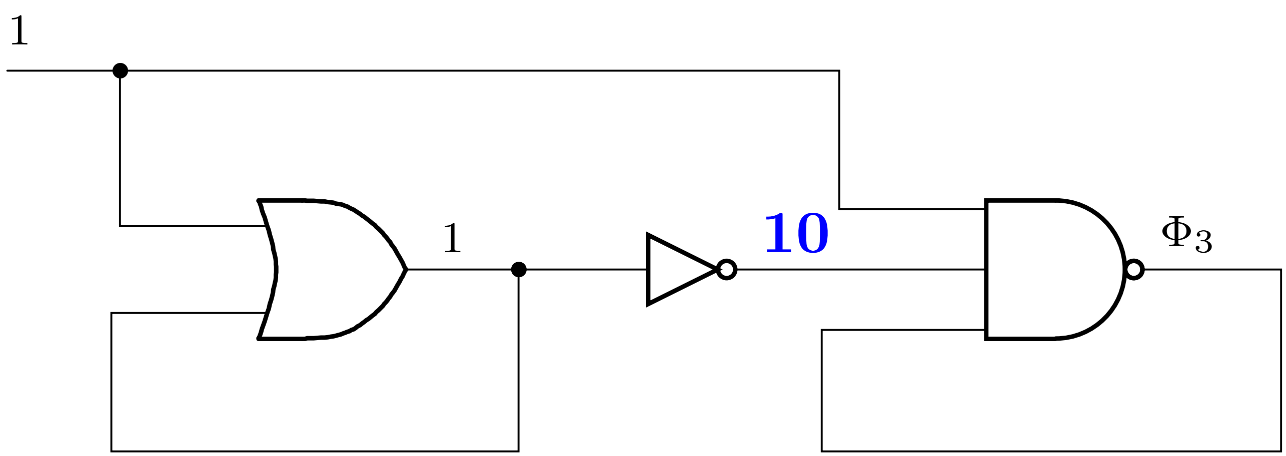




- **Change input** OR **unstable**

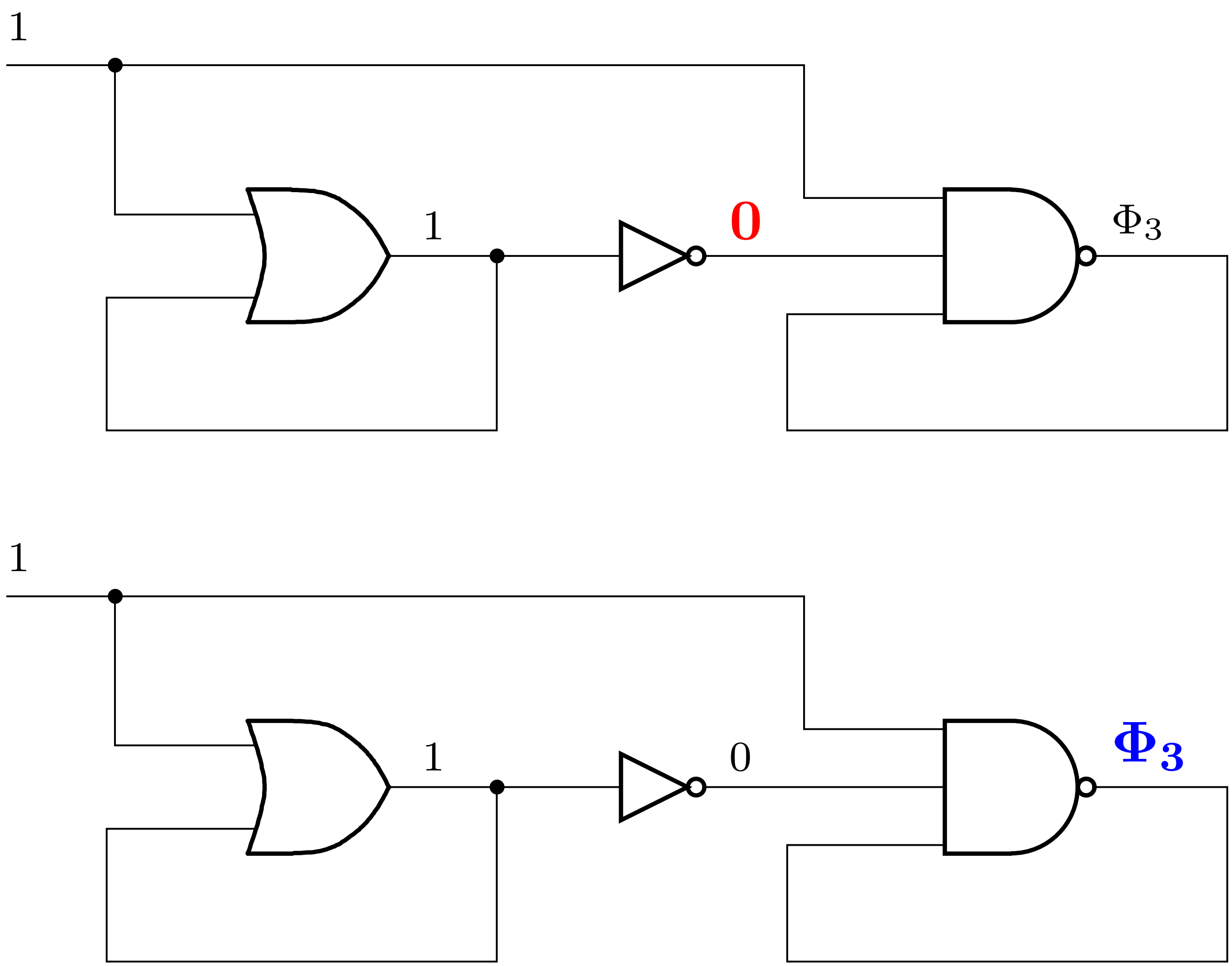
Algorithm B in Algebra C_3 - State t^1





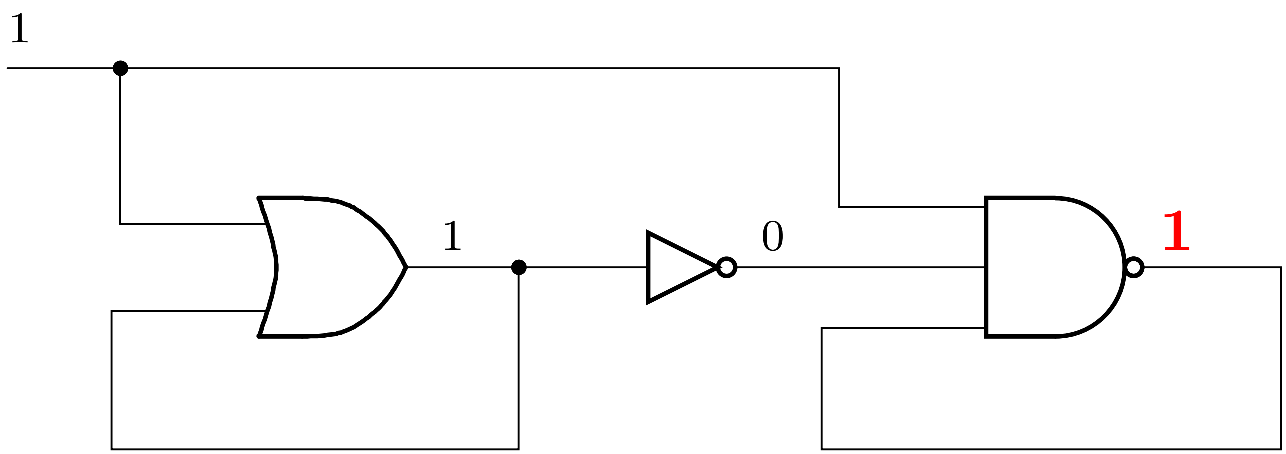
- **Change** OR   INVERTER **unstable**

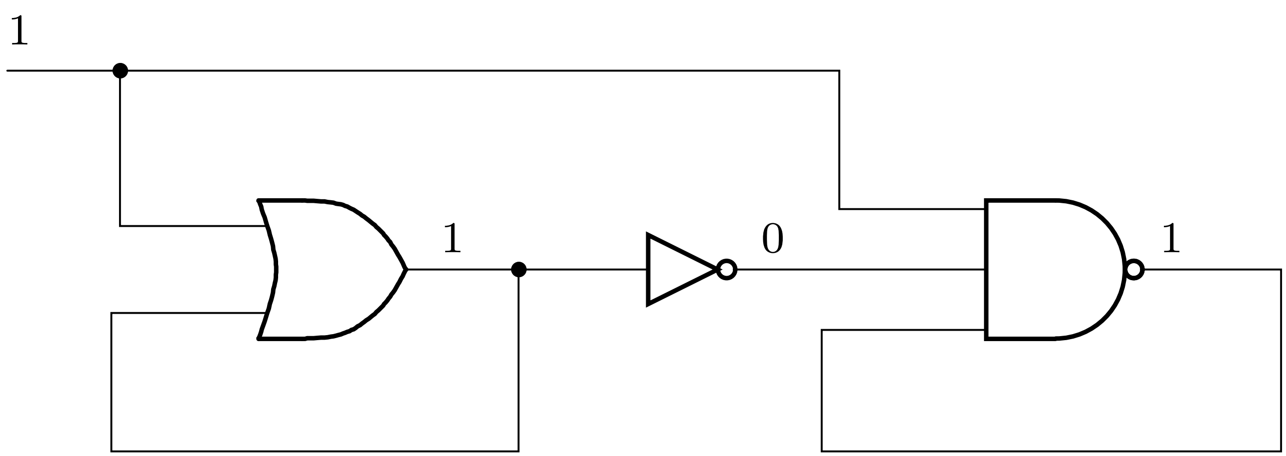
Algorithm B in Algebra C_3 - State t^2



- **Change** INVERTER   NAND **unstable**

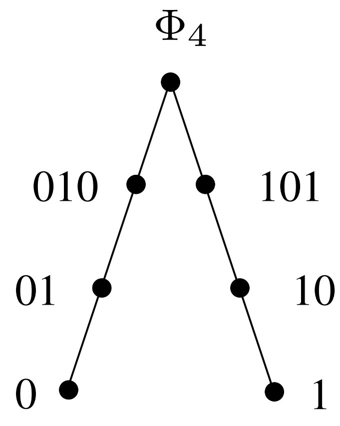
Algorithm B in Algebra C_3 - State $\mathbf{t}^3 = \mathbf{t}^B$





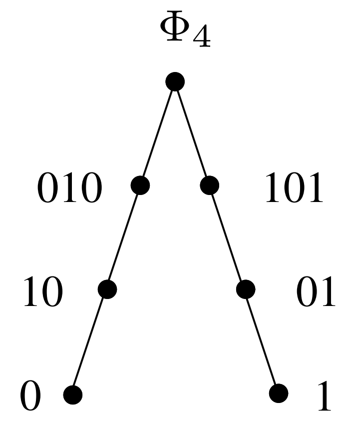
- **Change** NAND **Circuit is stable with correct final values**

Partial Orders in Algebra C_k



(a)

prefix order for C_4



(b)

suffix order for C_4

- Algorithm A is **monotonically increasing in the prefix order**
  - $b = \mathbf{s}^0 \leq \mathbf{s}^1 \leq \mathbf{s}^2 \leq \ldots \leq \mathbf{s}^A$
- Always **terminates**
- Algorithm B is **monotonically decreasing in the suffix order**
  - $\mathbf{s}^A = \mathbf{t}^0 \succeq \mathbf{t}^1 \succeq \mathbf{t}^1 \succeq \ldots \succeq \mathbf{t}^B$
- Always **terminates**

Relation among Stable States

$$\mathbf{f}(\mathbf{a}, \mathbf{s}^A) = \mathbf{s}^A$$

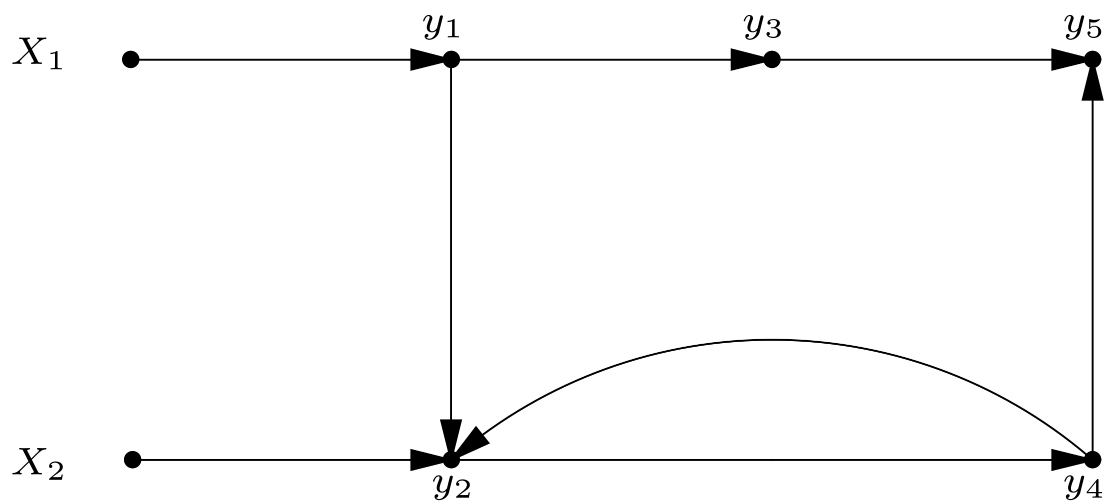
$\leq$ $\succeq$

$$\mathbf{f}(a, \mathbf{t}^B) = \mathbf{t}^B$$

$$\mathbf{f}(\hat{a}, b) = b$$

$$\hat{a} \leq \mathbf{a} \succeq a$$

Graph Model of a Circuit



- **Circuit is a directed graph with Boolean functions**
- Input nodes — indegree 0;   gate nodes — indegree > 0
- Gate excitation functions

  $f_1 = \overline{X_1}$; $f_2 = X_2 \wedge y_1 \wedge y_4$; $f_3 = y_1$; $f_4 = \overline{y_2}$; $f_5 = y_3 \vee y_4$.

Gate Types

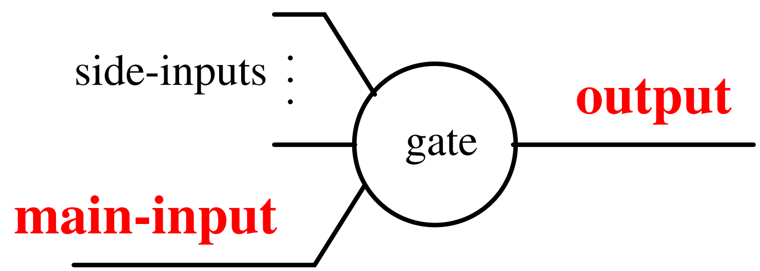
- $\mathcal{H} = \{$ OR, XOR $\}$ - multi-input
- One-input OR gate is an identity gate
- $\tilde{\mathcal{H}}$ is the set of functions obtained by complementing any number of inputs and/or the output of functions from $\mathcal{H}$
- $\mathcal{G} = \mathcal{H} \cup \tilde{\mathcal{H}}$
- All functions of two variables included
- Multi-input AND, NAND, NOR, XNOR included

Interior Values

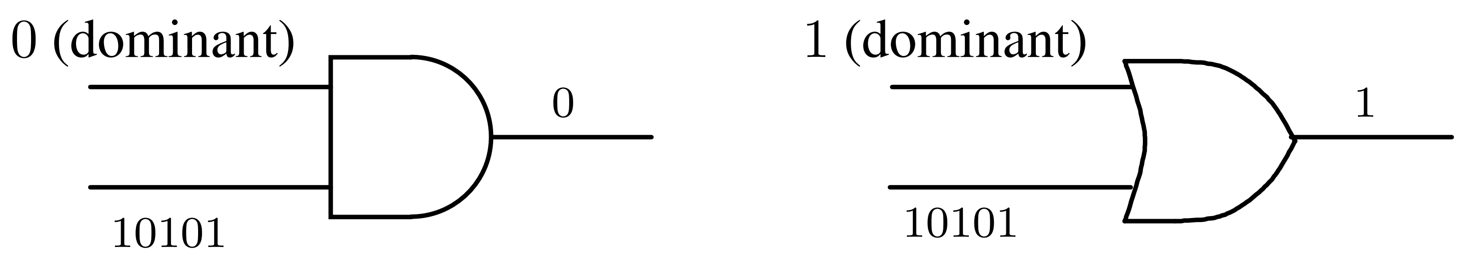
- 0, 1, and Φ_k are **exterior** values
- $01, 10, 010, 101, \ldots$ up to but not including Φ_k are **interior**
- There are no interior values in $\mathcal{C}_2$; $\quad T_2 = \{0, 1, \mathbf{\Phi}_2\}$
- We show there are no interior values in result of Algorithm B in $\mathcal{C}_k$
- This implies our main result

Dominant Input Values

- **Main and Side-Inputs:**



- **Dominant Inputs:**



- **Property of Dominant Inputs:**

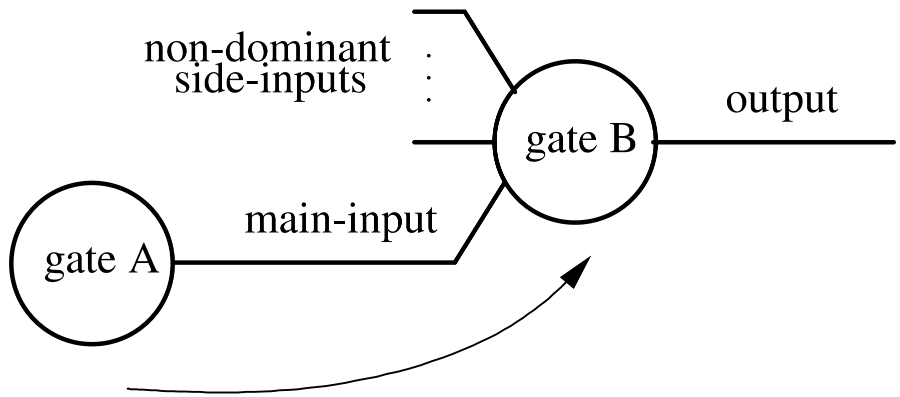
  length(output) $<$ length(main input)

  iff

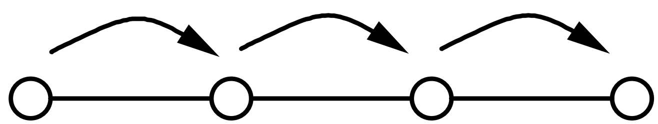
  length(main-input) $>$ 1, and one side-input is dominant

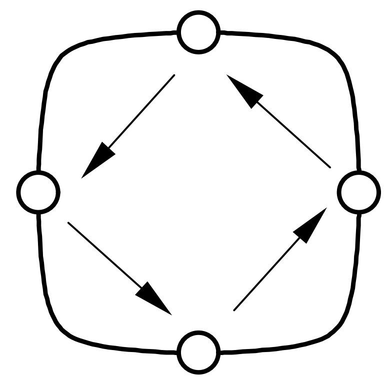
Activation Relation

- **Gate A activates Gate B**



- **Activation is a transitive relation in a chain**

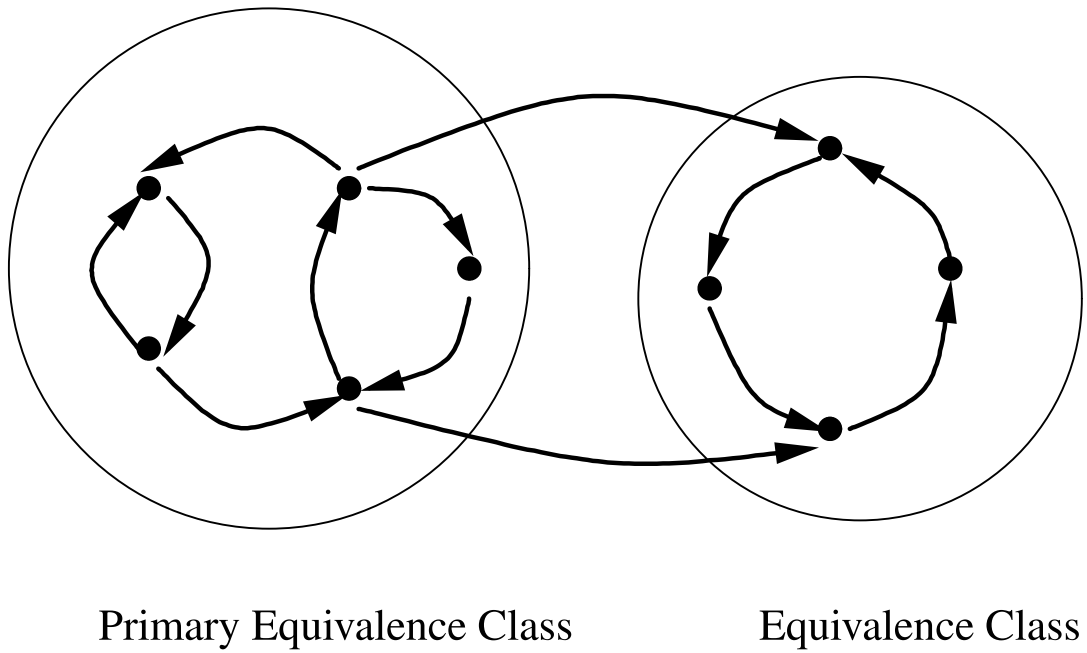




- **Activation is an equivalence relation in a cycle**
- **All transients in an active cycle in a stable state have the same length**

Equivalence Classes of Activation Relation

- **A graph showing activation**



- **Equivalence class**

  Maximal set of gates that activate each other

- **Primary equivalence class**

  No gate in another class activates a gate in a primary class

Proof Sketch of Main Theorem

- Suppose a gate has an interior value in $\mathbf{y}^B$
- Then it belongs to an equivalence class of activation relation
- Then there is a primary equivalence class in $\mathbf{y}^B$
- If a gate has an interior value in $\mathbf{y}^B$, then it has the same value in $\mathbf{y}^A$
- Main argument is done for OR and XOR, then extended to $\mathcal{G}$
- Primary equivalence class cannot have interior values in $\mathbf{y}^A$
- Therefore, we cannot have equivalence class of interior values in $\mathbf{y}^B$

Conclusions

- Do Algorithm A in C_k
- Reduce results to C_2
- Do Algorithm B in C_2

- **Algorithms B in C_2 and in C_k contain the same information**
- **Φ_k means nontransient oscillation; otherwise, result is binary**
- **Complexity can be reduced from $O(n^2 k \log k)$ to $O(n^2)$**