Lancaster University

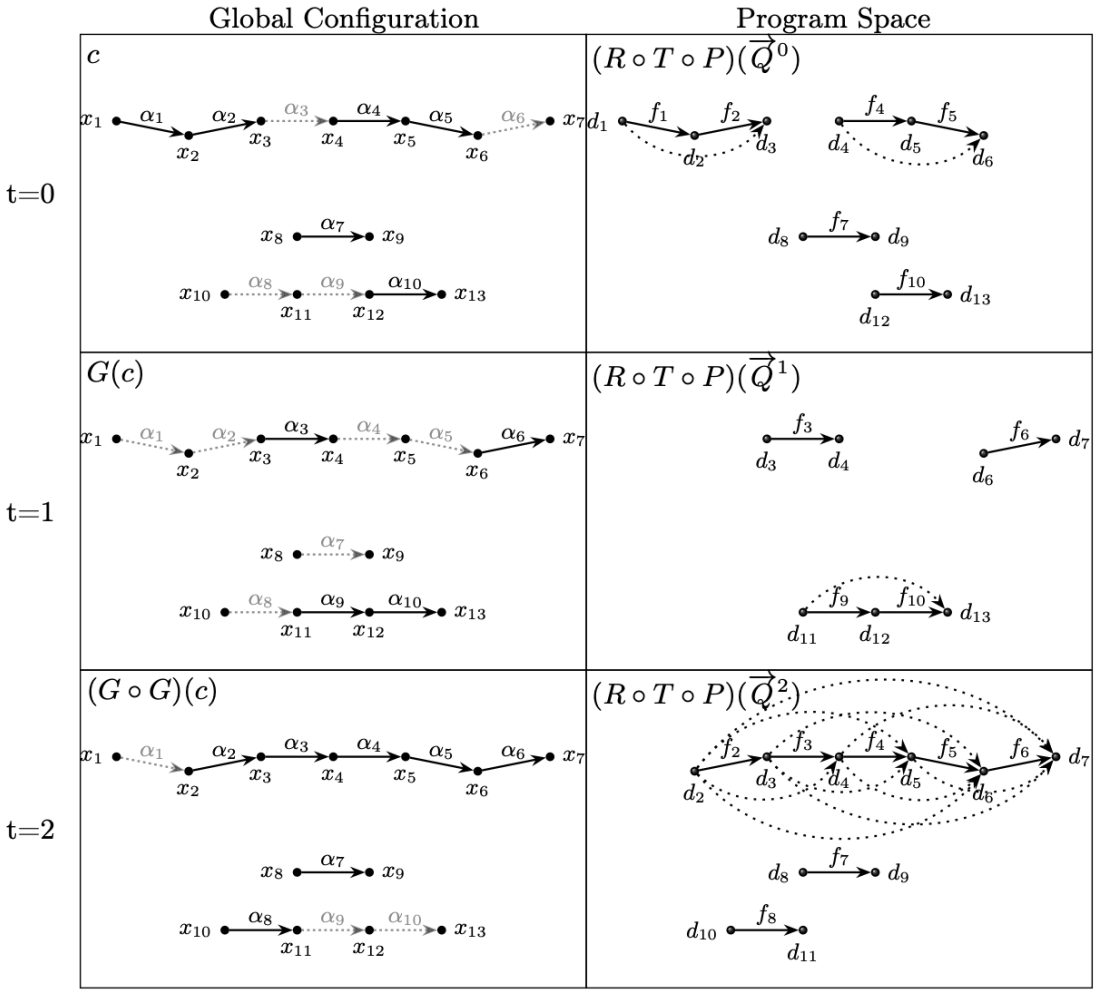
Compositional Control-Driven Boolean Circuits

Damian Arellanes

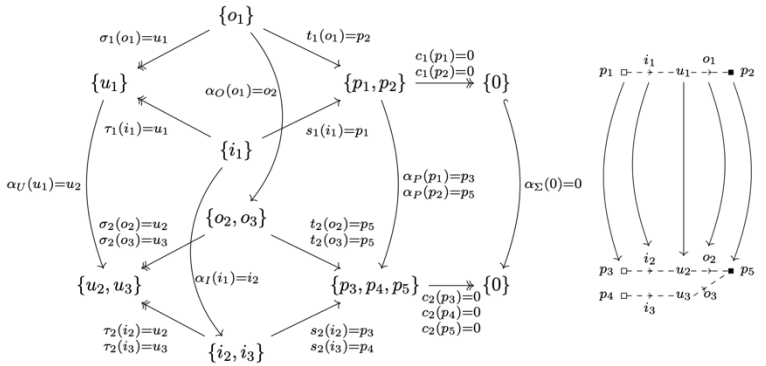
Theoretical Computer Science Group

Motivation

Emergence of Sequential Program Spaces (2021)



Computons (2023)

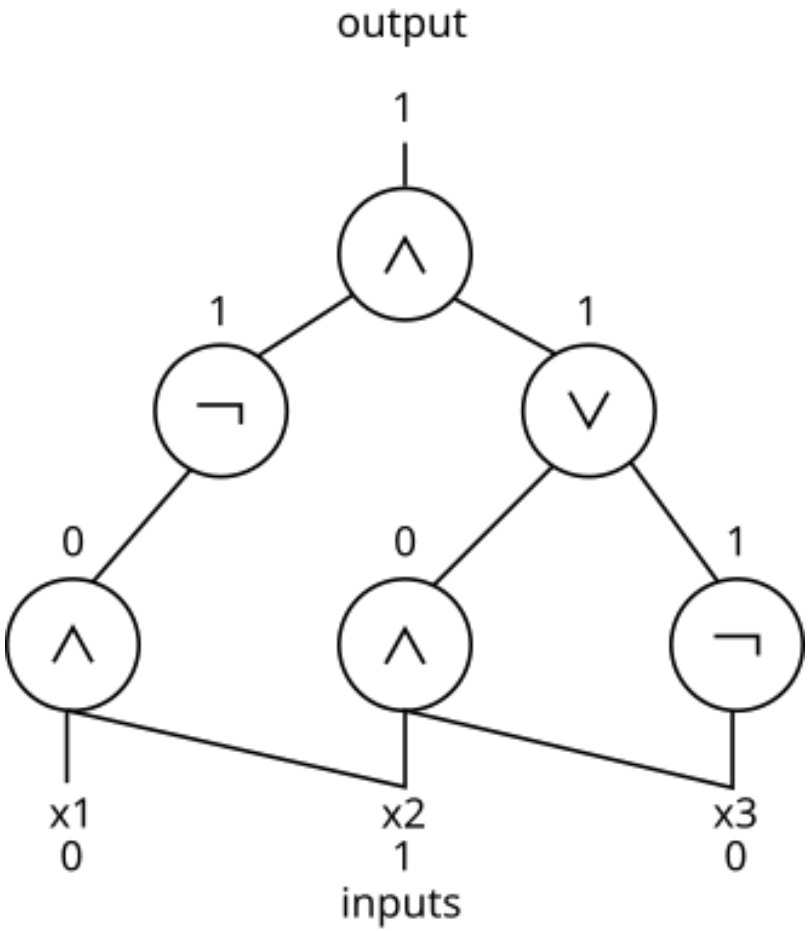


D. Arellanes. *Compositional Separation of Control Flow and Data Flow*. Journal of Logical and Algebraic Methods in Programming, vol. 150, 2026.

$$\begin{array}{ccccc} & & O & & \\ & \overset{\sigma}{\swarrow} & & \overset{t}{\searrow} & \\ U & & & & P \xrightarrow{\;c\;} \Sigma \\ & \underset{\tau}{\nwarrow} & & \underset{s}{\nearrow} & \\ & & I & & \end{array}$$

$$\begin{array}{ccc} P_1 & \xrightarrow{\;c_1\;} & \Sigma_1 \\ {\scriptstyle \alpha_P}\downarrow & & \downarrow{\scriptstyle \alpha_\Sigma} \\ P_2 & \xrightarrow[\;c_2\;]{} & \Sigma_2 \end{array} \qquad \begin{array}{ccccc} I_1 & \xrightarrow{\;\tau_1\;} & U_1 & \xleftarrow{\;\sigma_1\;} & O_1 \\ {\scriptstyle \alpha_I}\downarrow & & \downarrow{\scriptstyle \alpha_U} & & \downarrow{\scriptstyle \alpha_O} \\ I_2 & \xrightarrow[\;\tau_2\;]{} & U_2 & \xleftarrow[\;\sigma_2\;]{} & O_2 \end{array} \qquad \begin{array}{ccccc} I_1 & \xrightarrow{\;s_1\;} & P_1 & \xleftarrow{\;t_1\;} & O_1 \\ {\scriptstyle \alpha_I}\downarrow & & \downarrow{\scriptstyle \alpha_P} & & \downarrow{\scriptstyle \alpha_O} \\ I_2 & \xrightarrow[\;s_2\;]{} & P_2 & \xleftarrow[\;t_2\;]{} & O_2 \end{array}$$

Boolean Circuits



Source: https://en.wikipedia.org/wiki/Boolean_circuit

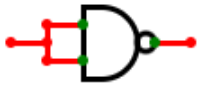
Functional Completeness

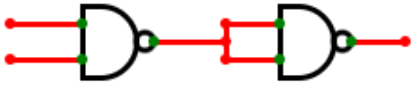
Examples of (minimal) functional complete sets

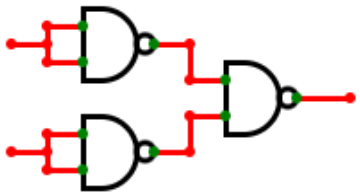
- $\{\wedge, \vee, \neg\}$
- $\{\rightarrow, \neg\}$
- $\{\uparrow\}$

Functional Completeness

We can use NAND gates to construct any possible Boolean circuit

NOT ≡ 

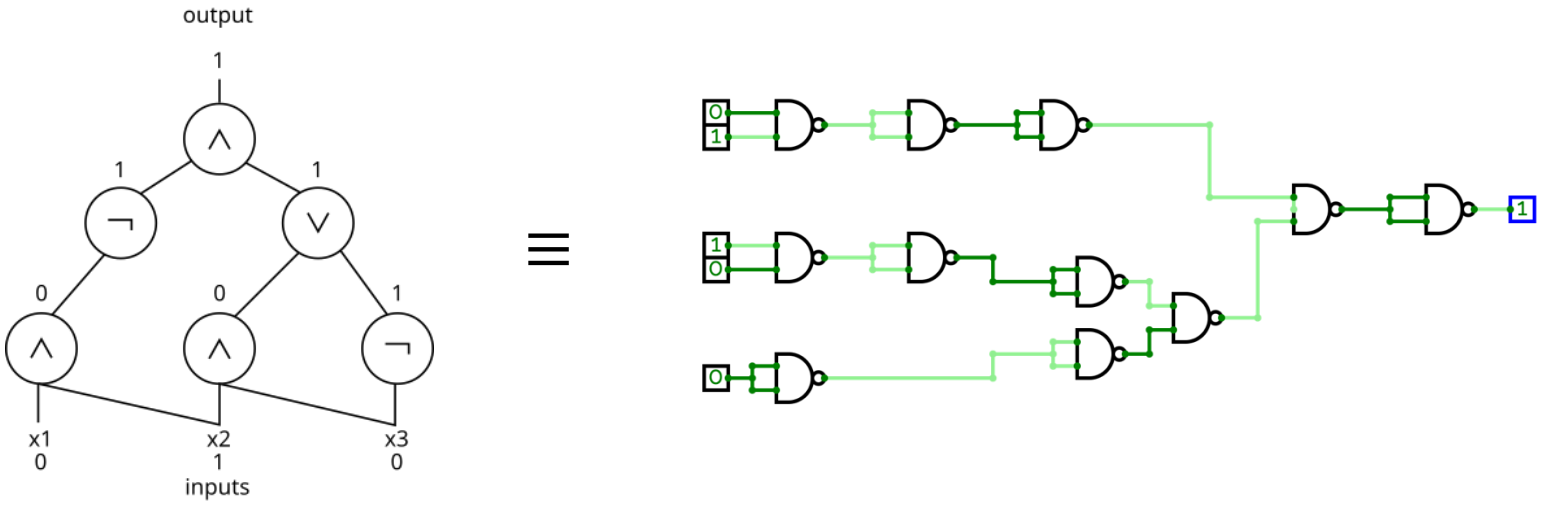
AND ≡ 

OR ≡ 

*Circuits created with CircuitVerse: https://circuitverse.org/

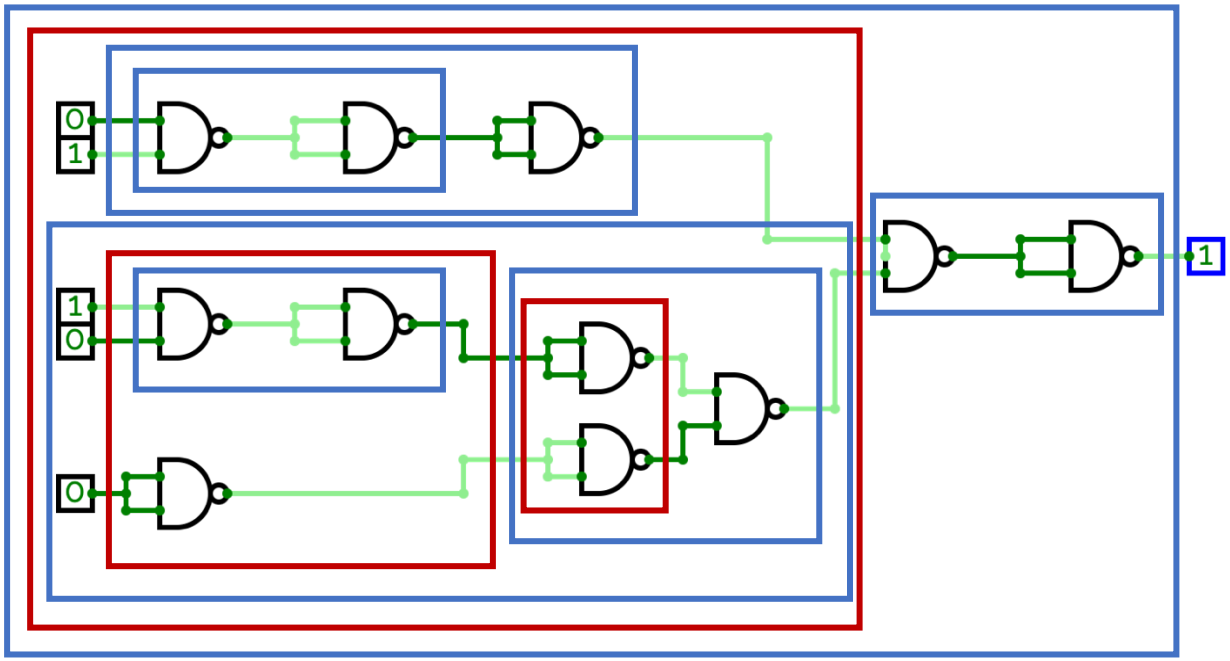
Functional Completeness

We can use NAND gates to construct any possible Boolean circuit



Functional Completeness

We can use NAND gates to construct any possible Boolean circuit

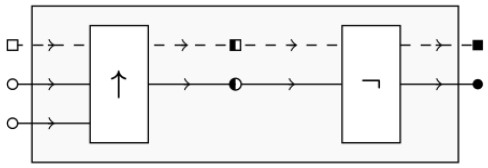


Insight: There is implicit control flow in every Boolean circuit

Problem: How to make it explicit?

Control-Driven Boolean Circuits

Control-Driven Circuits



Definition 1 (Computation Unit). *A computation unit is the indexed family* $(\uparrow_k)_{k\in\mathbb{N}}$ *where* $\uparrow_0 = 1$ *is a constant function and, for* $k > 0$, $\uparrow_k$ *is the* k*-ary NAND operator* $\mathbb{B}^k \to \mathbb{B}$ *given by* $\uparrow_k (b_1, \ldots, b_k) = \neg(b_1 \wedge \cdots \wedge b_k)$.

Definition 2 (Control-Driven Boolean Circuit). *A control-driven Boolean circuit* λ *is a 10-tuple* $(V, U, I, O, \Sigma, s, t, \sigma, \tau, c)$ *where:*

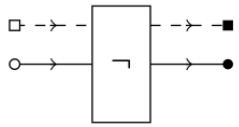
- V *is a non-empty finite set of variables,*
- U *is a finite set of computation units,*
- I *is a finite set of input flows linking variables to computation units,*
- O *is a finite set of output flows linking computation units to variables,*
- $\Sigma \subseteq \{\mathbb{1}, \mathbb{B}\}$ *is a non-empty finite set of types,*
- $s\colon I \to V$ *is a function that specifies the source variable of each input flow,*
- $\tau\colon I \twoheadrightarrow U$ *is a function that defines the target unit of each input flow,*
- $\sigma\colon O \twoheadrightarrow U$ *is a function that specifies the source unit of each output flow,*
- $t\colon O \to V$ *is a function defining the target variable of each output flow and*
- $c\colon V \twoheadrightarrow \Sigma$ *is a function that assigns a type to each variable,*

such that $\sigma\restriction_{(\mathrm{cot})^{-1}(\mathbb{1})}$ *and* $\tau\restriction_{(\mathrm{cos})^{-1}(\mathbb{1})}$ *are surjective and there is at least one variable* $v \in V \setminus s(I)$ *and at least one variable* $w \in V \setminus t(O)$ *with* $c(v) = \mathbb{1} = c(w)$.

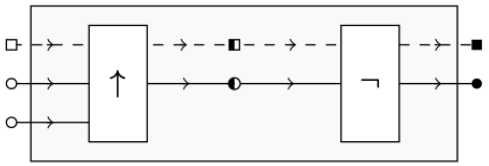
Control-Driven Circuits

Definition 2 (Control-Driven Boolean Circuit). *A control-driven Boolean circuit* λ *is a 10-tuple* $(V, U, I, O, \Sigma, s, t, \sigma, \tau, c)$

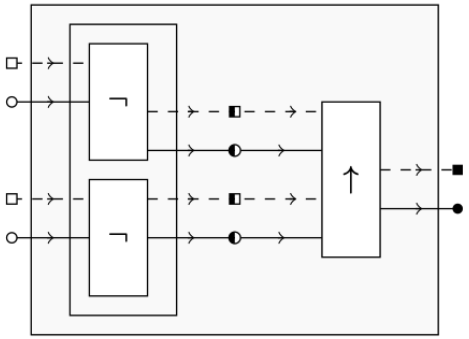
(a) NOT circuit

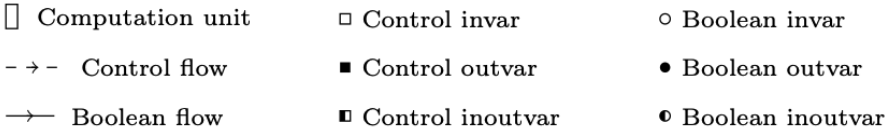


(b) AND circuit



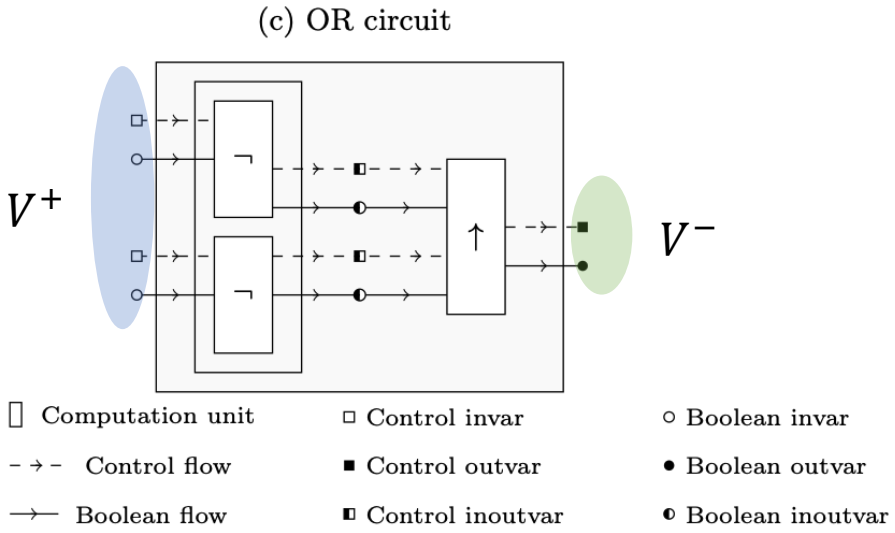
(c) OR circuit





Circuit Interfaces

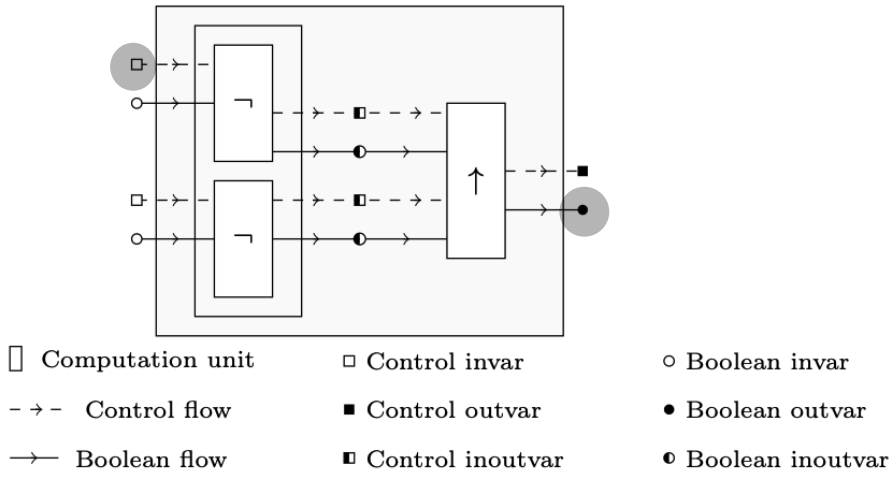
Definition 3 (Circuit Interface). *The interface of a circuit λ is a tuple (V^+, V^-) where V^+ and V^- are the sets $V \setminus t(O)$ and $V \setminus s(I)$, respectively.*



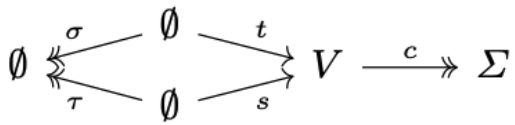
Soundness

Definition 4 (Sound Circuit). *A circuit is sound if every* $v \in s(I) \cup V^+$ *satisfies* $v \xrightarrow{\exists} w$ *for some* $w \in V^-$.

(c) OR circuit

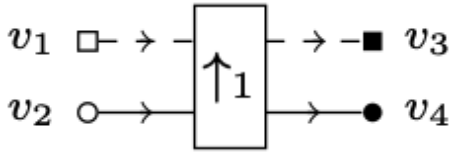


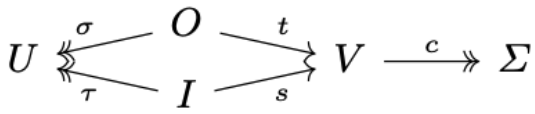
Primitive and trivial Circuits



(a)

Trivial Circuits





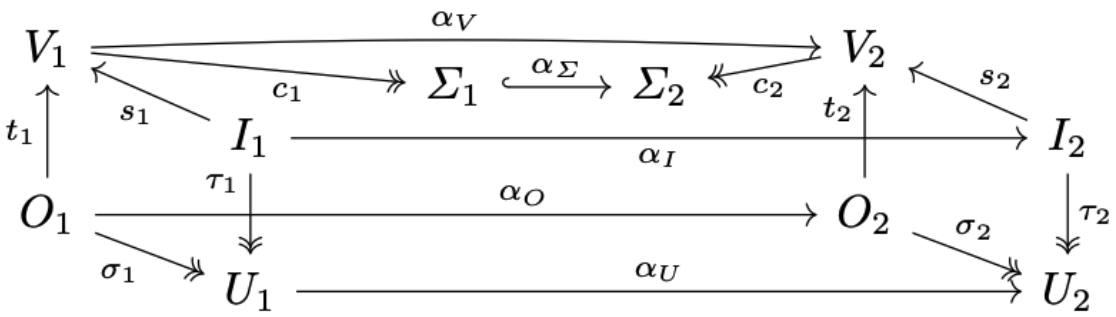
(b) where $V = s(I) \triangle t(O)$ and $|U| = 1$.

Primitive Circuits

Proposition 1. *Every primitive circuit is sound.*

Circuit Morphisms

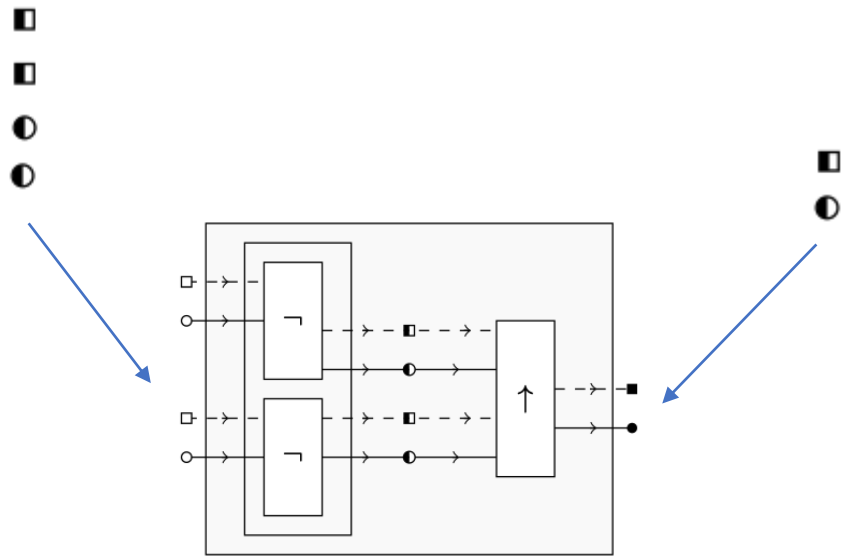
Definition 5 (Circuit Morphism). *A circuit morphism* $\alpha\colon \lambda_1 \to \lambda_2$ *is a tuple* $(\alpha_V, \alpha_U, \alpha_I, \alpha_O, \alpha_\Sigma)$ *of total functions* $\alpha_V : V_1 \to V_2$, $\alpha_U : U_1 \to U_2$, $\alpha_I : I_1 \to I_2$, $\alpha_O : O_1 \to O_2$ *and* $\alpha_\Sigma : \Sigma_1 \hookrightarrow \Sigma_2$, *satisfying the condition* $\vec{i}(\alpha) \cup \vec{o}(\alpha) \subseteq V_1^+ \cup V_1^-$ *with* $\vec{i}(\alpha) := \{v \in V_1 \mid \bullet\alpha_V(v) \setminus \alpha_V(\bullet v) \neq \emptyset\}$ *and* $\vec{o}(\alpha) := \{v \in V_1 \mid \alpha_V(v)\bullet \setminus \alpha_V(v\bullet) \neq \emptyset\}$, *and making the following diagram commute:*



$$(\alpha_V, \alpha_U, \alpha_I, \alpha_O, \alpha_\Sigma) \circ (\beta_V, \beta_U, \beta_I, \beta_O, \beta_\Sigma) := (\alpha_V \circ \beta_V, \alpha_U \circ \beta_U, \alpha_I \circ \beta_I, \alpha_O \circ \beta_O, \alpha_\Sigma \circ \beta_\Sigma)$$

Adjoint Morphisms

Definition 6 (Adjoint Morphism). *An adjoint* $\lambda^{\square}$ *of a circuit* λ *is a circuit monomorphism* $\lambda_0 \to \lambda$ *where* λ_0 *is a trivial circuit with* $\lambda^{\square}(V_0) = V^{\square}$ *and* $\square = \{+, -\}$. *If* $\square = +$, *it is called* in-adjoint; *otherwise, it is called* out-adjoint.



Proposition 2. *For any circuit morphism* $\alpha\colon \lambda_1 \to \lambda_2$, *we have* $\alpha_V^{-1}(V_2^+) \subseteq V_1^+$ *and* $\alpha_V^{-1}(V_2^-) \subseteq V_1^-$.

Colimit Constructions

Definition 7 (Pushout). *The pushout* $\lambda_1 +_{\lambda_0} \lambda_2$ *of a span* $\rho := \lambda_1 \xleftarrow{\alpha} \lambda_0 \xrightarrow{\beta} \lambda_2$ *of circuit morphisms is obtained by computing quotient sets componentwise. For example, the set of variables of* $\lambda_1 +_{\lambda_0} \lambda_2$ *is given by* $V_1 \sqcup V_2/\sim$ *where* $\sim$ *is the finest equivalence relation given by* $\alpha_V(v) \sim \beta_V(v)$ *for all* $v \in V_0$*. The pushout of* ρ *exists only if* $\alpha(\vec{i}(\beta)) \cup \alpha(\vec{o}(\beta)) \subseteq P_1^+ \cup P_1^-$ *and* $\beta(\vec{i}(\alpha)) \cup \beta(\vec{o}(\alpha)) \subseteq P_2^+ \cup P_2^-$*.*

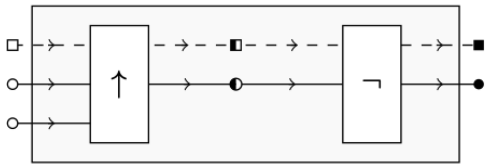
Definition 8 (Coproduct). *The coproduct* $\lambda_1 + \lambda_2$ *of circuits* λ_1 *and* λ_2 *is obtained by computing disjoint union componentwise such that the two canonical coproduct morphisms associated with* $\lambda_1 + \lambda_2$ *are circuit monomorphisms.*

Composition Operators

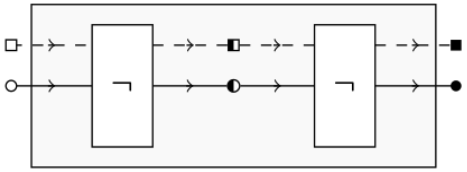
Sequential Circuits

Definition 9 (Sequentiable Span). *A span* $\lambda_1 \xleftarrow{\alpha} \lambda_0 \xrightarrow{\beta} \lambda_2$ *is sequentiable if* λ_0 *is a trivial circuit, and* α *and* β *are monomorphisms satisfying* $\alpha_V(V_0) \subseteq V_1^-$ *and* $\beta_V(V_0) \subseteq V_2^+$*, respectively. If* $\alpha_V(V_0) = V_1^-$ *and* $\beta_V(V_0) = V_2^+$*, the span is totally sequentiable; otherwise, it is partially sequentiable.*

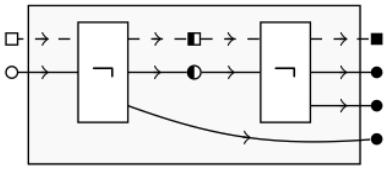
Definition 10 (Sequential Circuit). *Let* $\rho := \lambda_1 \xleftarrow{\alpha} \lambda_0 \xrightarrow{\beta} \lambda_2$ *be a sequentiable span of circuit morphisms. If* ρ *is totally sequentiable, its pushout is called a total sequential circuit* $\lambda_1 \trianglerighteq_\rho \lambda_2$*. Otherwise, its pushout is a called a partial sequential circuit* $\lambda_1 \triangleright_\rho \lambda_2$*. The operations to form total and partial sequential circuits are called total sequencing and partial sequencing, respectively.*



AND Circuit



Buffer Circuit



Partial Buffer

Sequential Circuits

Proposition 3. *If* $\lambda_1 \xrightarrow{\gamma} \lambda_3 \xleftarrow{\theta} \lambda_2$ *is the cospan induced by the pushout of a sequentiable span* $\lambda_1 \xleftarrow{\alpha} \lambda_0 \xrightarrow{\beta} \lambda_2$*, then* γ *and* θ *are circuit monomorphisms.*

Proposition 4. *Assume* $\rho := \lambda_1 \xleftarrow{\alpha} \lambda_0 \xrightarrow{\beta} \lambda_2$ *is a sequentiable span of circuit morphisms. If* $\lambda_1 \xrightarrow{\gamma} \lambda_3 \xleftarrow{\theta} \lambda_2$ *is the cospan induced by the pushout of* ρ*, then* $\gamma_V(V_1^+) \subseteq V_3^+$ *and* $\theta_V(V_2^-) \subseteq V_3^-$*.*

Proposition 5. *Assume* $\rho := \lambda_1 \xleftarrow{\alpha} \lambda_0 \xrightarrow{\beta} \lambda_2$ *is a totally sequentiable span of circuit morphisms. If* $\lambda_1 \xrightarrow{\gamma} \lambda_3 \xleftarrow{\theta} \lambda_2$ *is the cospan induced by the pushout of* ρ*, then* $\gamma_V(V_1^+) = V_3^+$ *and* $\theta_V(V_2^-) = V_3^-$*.*

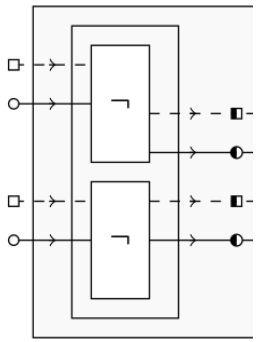
Theorem 1. *Up to isomorphism, the unit circuit* ◧ *serves as the left- and right-identity for (total or partial) sequencing.*

Theorem 2. *Total sequencing is associative up to isomorphism.*

Parallel Circuits

Definition 11. *A parallel circuit $\lambda_1 + \lambda_2$ is a coproduct of λ_1 and λ_2.*

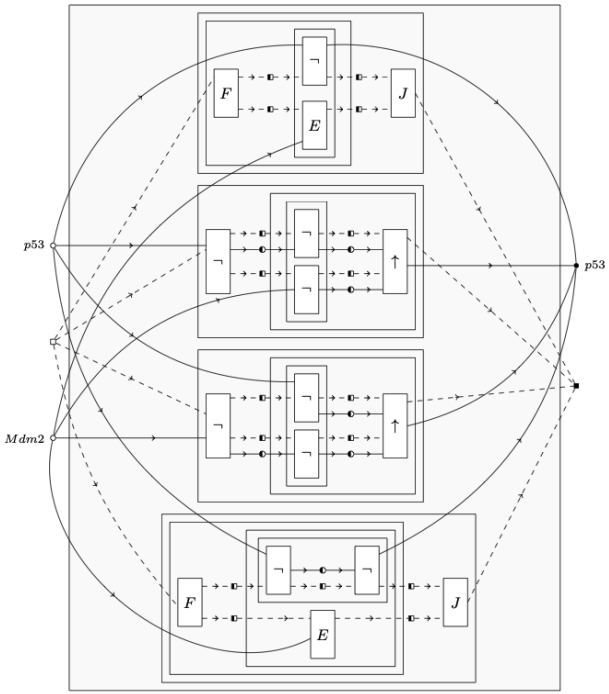
Theorem 3. *Parallelising is associative and commutative up to isomorphism.*



Parallel NOT circuit

Branching Circuits

Definition 12 (Branching Circuit). *Let ρ be the diagram formed by the spans $\lambda_2 \xleftarrow{\lambda_2^+} \lambda_0 \xrightarrow{\lambda_3^+} \lambda_3$ and $\lambda_2 \xleftarrow{\lambda_2^-} \lambda_1 \xrightarrow{\lambda_3^-} \lambda_3$ of adjoint morphisms. A branching circuit $\lambda_2 ?_\rho \lambda_3$ is the colimit of ρ, computed as $\lambda_2 +_{\lambda_0+\lambda_1} \lambda_3$. By Definition 6, λ_0 and λ_1 are necessarily trivial circuits.*



Oscillatory dynamics of a p53-Mdm2

Branching Circuits

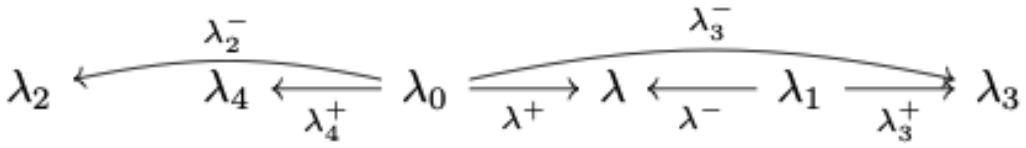
Proposition 6. *If ρ is the diagram formed by the spans $\lambda_2 \xleftarrow{\lambda_2^+} \lambda_0 \xrightarrow{\lambda_3^+} \lambda_3$ and $\lambda_2 \xleftarrow{\lambda_2^-} \lambda_1 \xrightarrow{\lambda_3^-} \lambda_3$, λ_0 and λ_1 are the domains of the in- and out-adjoints of $\lambda_2 ?_\rho \lambda_3$, respectively.*

Theorem 4. *Branching is commutative up to isomorphism.*

Theorem 5. *Branching is associative up to isomorphism.*

Iterative Circuits

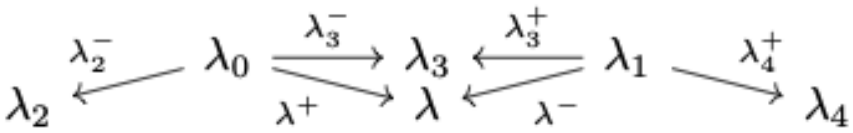
Definition 13 (Head-Iterative Circuit). *A head-iterative circuit $*_\rho\lambda$ is the colimit of a diagram ρ of adjoint morphisms:*



where λ, λ_2, λ_3 and λ_4 are sound circuits, and λ_0 and λ_1 are trivial (see Definition 6). The colimit of ρ is computed as $(\lambda_4 +_{\lambda_0} \lambda) +_{(\lambda_0+\lambda_1)} (\lambda_2 +_{\lambda_0} \lambda_3)$.

Tail-Iterative Circuits

Definition 14 (Tail-Iterative Circuit). *A tail-iterative circuit $\lambda*_\rho$ is the colimit of a diagram ρ of adjoint morphisms:*



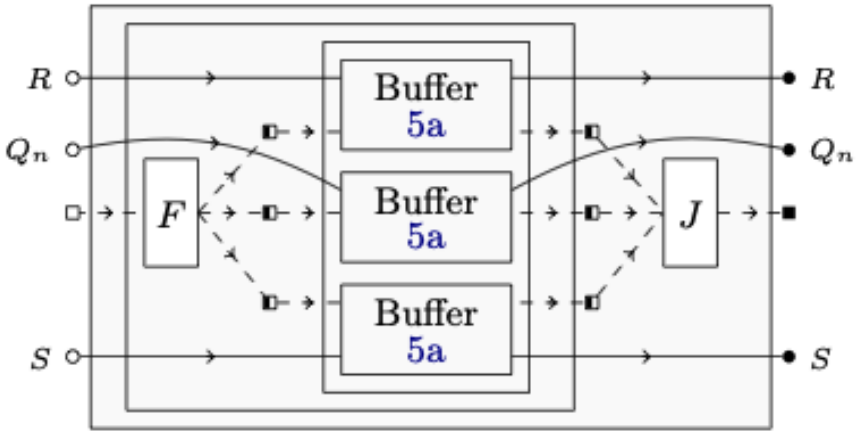
where λ, λ_2, λ_3 and λ_4 are sound circuits, and λ_0 and λ_1 are trivial (see Definition 6). The colimit of ρ is computed as $((\lambda_2 +_{\lambda_0} \lambda_3) +_{\lambda_3} (\lambda_3 +_{\lambda_1} \lambda_4)) +_{(\lambda_0+\lambda_1)} \lambda$.

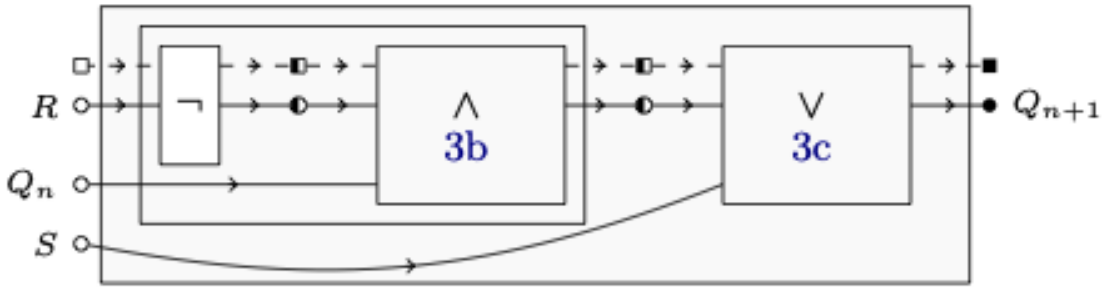
Constructing a Tail-Iterative Circuit

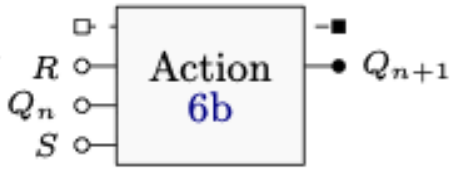
Toggle action of a clocked set-reset flip-flop

S_n	R_n	Q_n	$S_{n+1} = \neg Q_{n+1}$	$R_{n+1} = Q_{n+1}$	$Q_{n+1} = S \vee (\neg R \wedge Q_n)$	
0	0	0	1	0	0	Hold
0	0	1	0	1	1	
0	1	0	1	0	0	Reset
0	1	1	1	0	0	
1	0	0	0	1	1	Set
1	0	1	0	1	1	
1	1	0	0	1	1	Invalid
1	1	1	0	1	1	

Constructing a Tail-Iterative Circuit

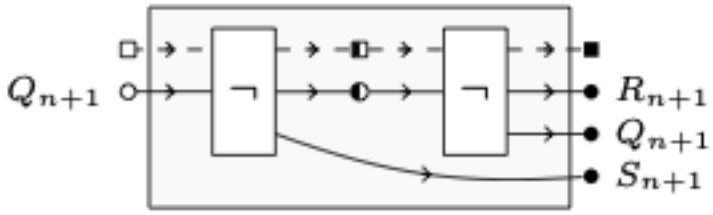


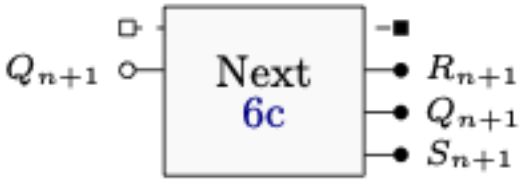




$$Q_{n+1} = S \vee (\neg R \wedge Q_n)$$

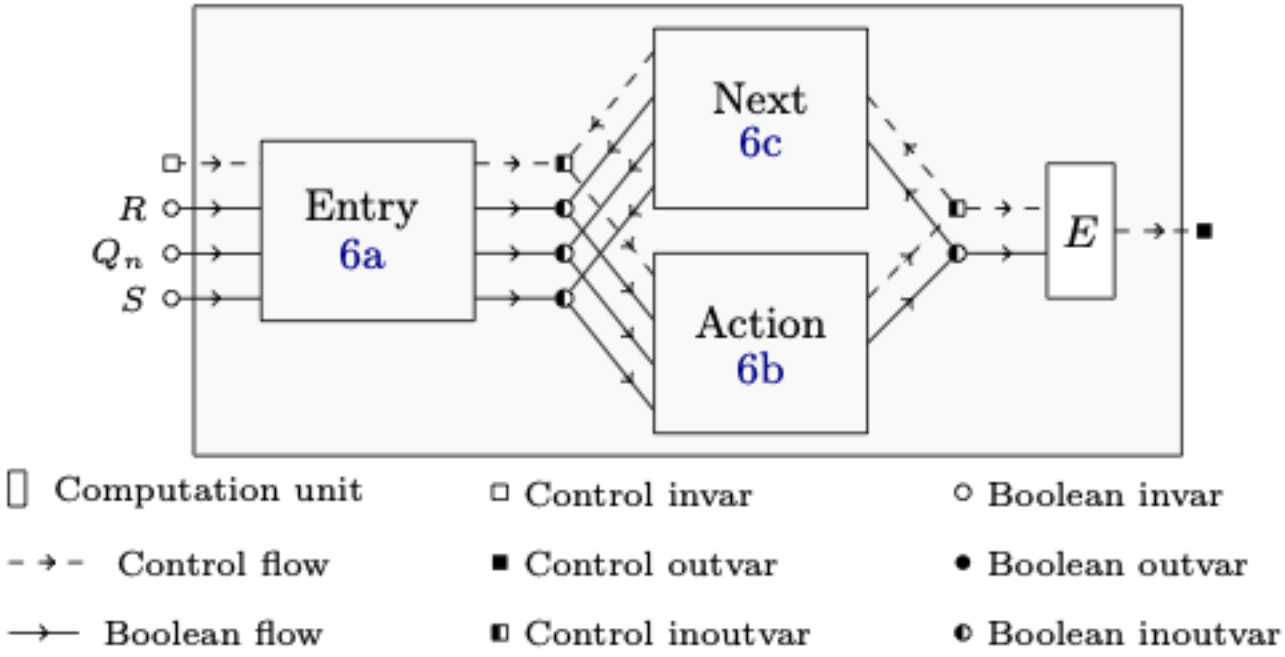
Constructing a Tail-Iterative Circuit





S_n	R_n	Q_n	$S_{n+1} = \neg Q_{n+1}$	$R_{n+1} = Q_{n+1}$	$Q_{n+1} = S \vee (\neg R \wedge Q_n)$

Constructing a Tail-Iterative Circuit



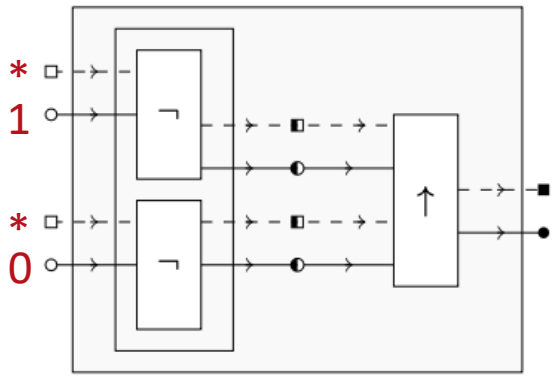
Circuit Dynamics

Circuit State

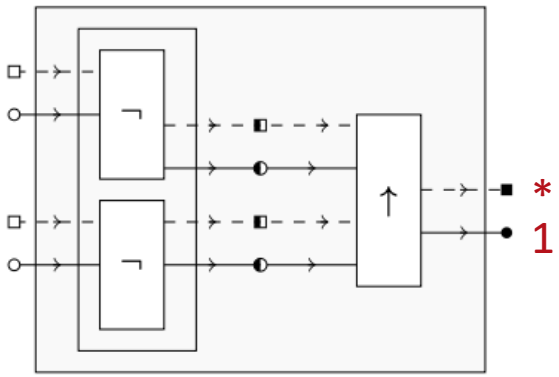
Definition 15 (Circuit State). *The state of a circuit λ at time j is a partial function $\delta^j : V \to \{*, 0, 1\}$ where for all $v \in Dom(\delta^j)$:*

$$c(v) = \mathbb{1} \iff \delta^j(v) \in \{*\} \text{ and}$$
$$c(v) = \mathbb{B} \iff \delta^j(v) \in \{0, 1\}$$

A state δ^j is initial if $Dom(\delta^j) = V^+$ or final if $Dom(\delta^j) = V^-$.



An Initial State



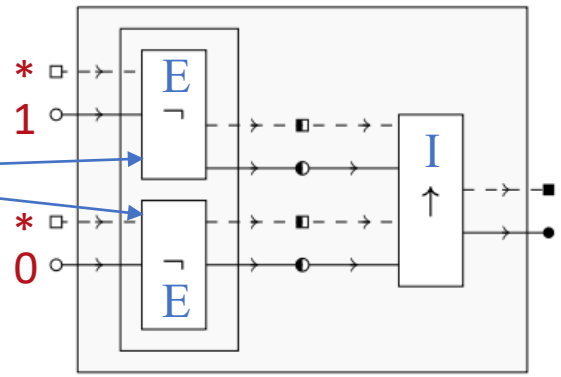
A Final State

Choosing Units for Reduction

Definition 16 (Unit Status). *Given a circuit λ and a state δ^j, a computation unit $u \in U$ is enabled under δ^j if $\bullet u \subseteq Dom(\delta^j)$; otherwise, u is idle under δ^j.*

Definition 17 (Ready Units). *Let E^j be the finite set of computation units enabled under δ^j and $\sim$ the equivalence relation $\{(u_1, u_2) \in E^j \times E^j \mid \bullet u_1 = \bullet u_2\}$. If A is the partition induced by $\sim$ and f is the random choice function on A, the set R^j of computation units ready to be reduced under δ^j is $\{f(E) \mid E \in A\}$.*

These two units are in R^0



Unit Reduction and State Transition

Definition 18 (Unit Reduction). *If λ is a circuit with $u \in R^j$ and B is the set $\{v \in \bullet u \mid c(v) = \mathbb{B}\}$, the result $[\![u]\!]^j$ of reducing u under δ^j is given by:*

$$[\![u]\!]^j = u(|B|)(\delta^j(b_1), \ldots, \delta^j(b_{|B|})) = \uparrow_{|B|} (\delta^j(b_1), \ldots, \delta^j(b_{|B|}))$$

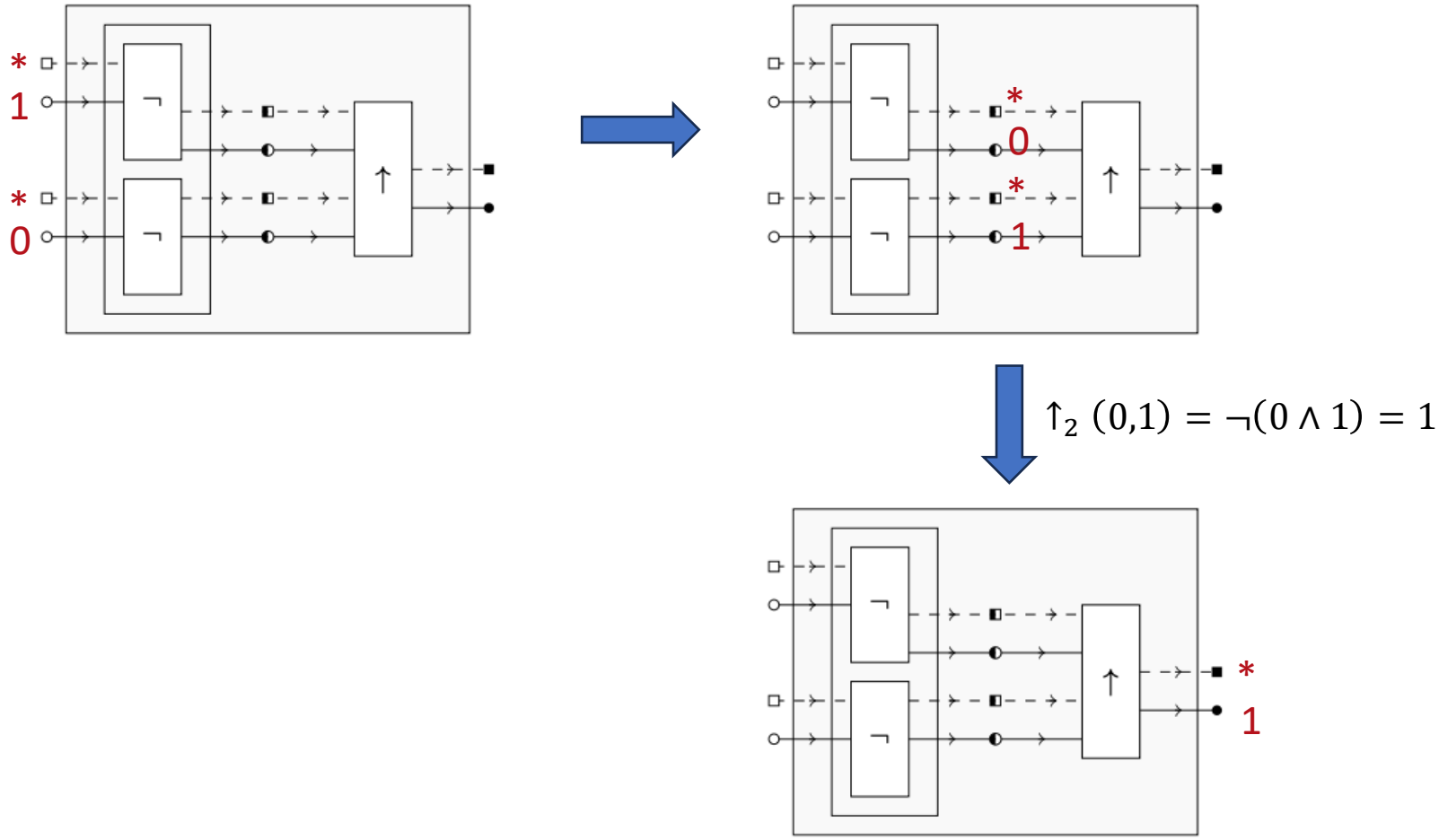
$$= \begin{cases} 1 & |B| = 0 \\ \neg(\delta^j(b_1) \wedge \cdots \wedge \delta^j(b_{|B|})) & \text{otherwise} \end{cases} \qquad \text{where } \bigcup_{k=1}^{|B|} \{b_k\} = B$$

Definition 19 (State Transition). *Given the state δ^j of a circuit λ for $j \geq 0$, δ^{j+1} is computed as follows for each $v \in V$:*

$$\delta^{j+1}(v) = \begin{cases} * & (\exists u \in R^j)[v \in u\bullet \text{ and } c(v) = \mathbb{1}] \\ [\![u]\!]^j & (\exists u \in R^j)[v \in u\bullet \text{ and } c(v) = \mathbb{B}] \\ \delta^j(v) & v \in Dom(\delta^j) \text{ and } (\nexists u \in R^j)[v \in \bullet u \cup u\bullet] \end{cases}$$

Definition 20 (Circuit Termination). *A circuit λ terminates if and only if there is a finite orbit of states from its initial state to its final state.*

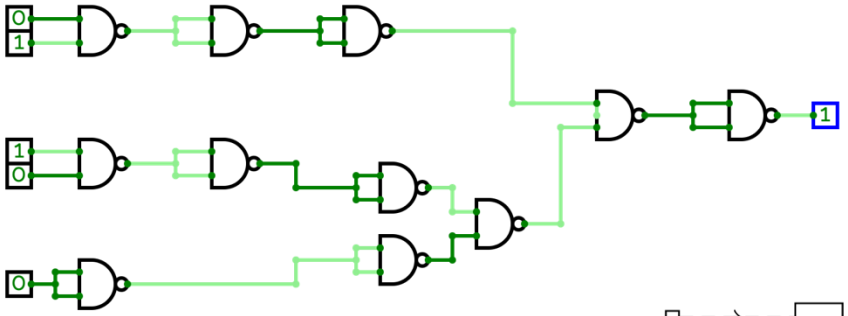
Unit Reduction and State Transition

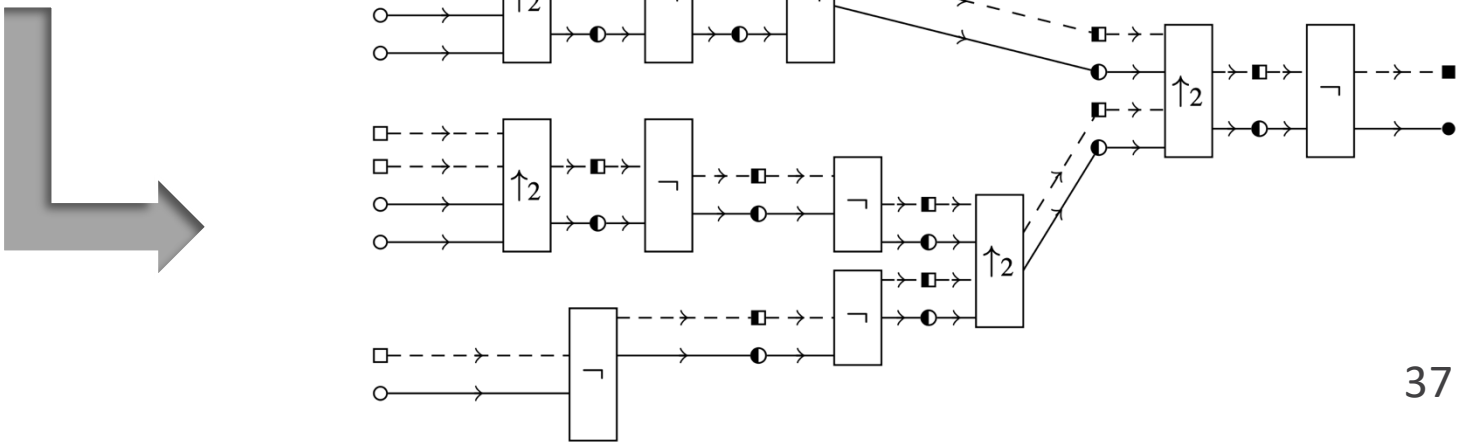


Control-Driven vs Classical

NAND Circuits are Control-Driven Circuits!

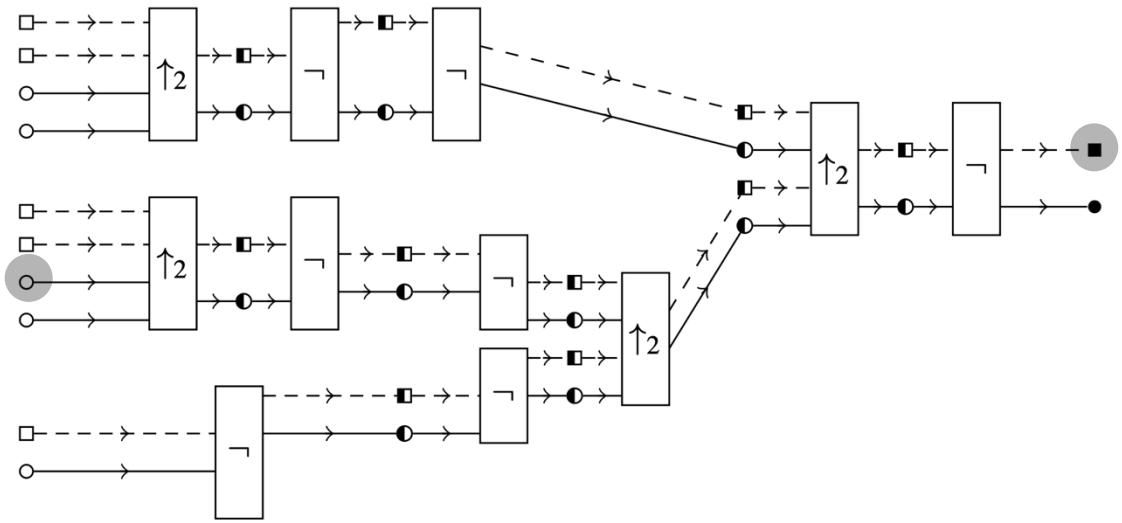
Theorem 6. *Every NAND circuit is a control-driven Boolean circuit.*





NAND Circuits are Control-Driven Circuits!

Proposition 7. *The control-driven circuit λ of a NAND circuit C is sound.*

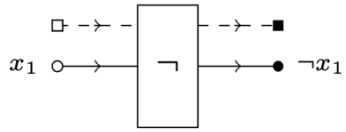


Non-Uniform Computation

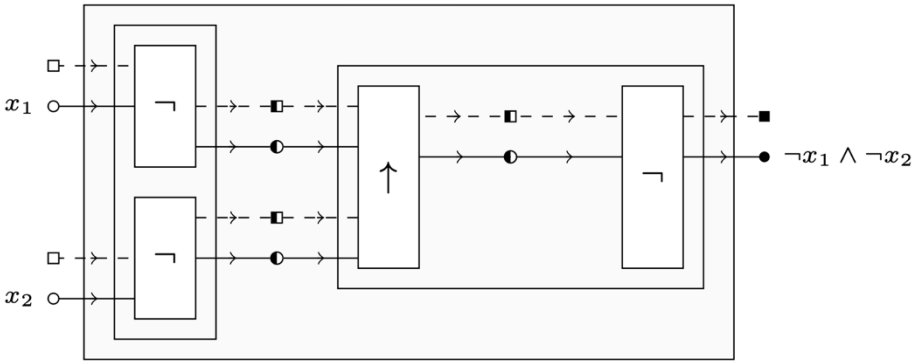
Theorem 7. *For any Boolean function* $\{0,1\}^* \to \{0,1\}$ *there is a family of control-driven Boolean circuits able to compute it.*

Non-Uniform Computation

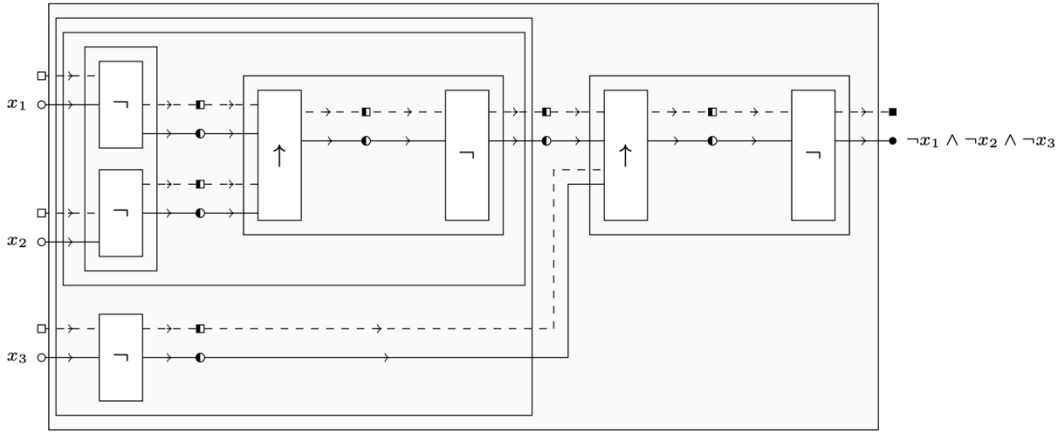
$$L = \{x \in \{0,1\}^n \mid x \text{ is all zeros}\}$$



for 1-bit inputs



for 2-bit inputs



for 3-bit inputs

Conclusions

- Families of control-driven Boolean circuits
  - Non-uniform model of computation
- Colimit-based composition operations:
  - Partial sequencing (identity)
  - Total sequencing (identity, associative)
  - Parallelising (commutative, associative)
  - Branching (commutative, associative)
  - Iteration
- Showed that every NAND circuit is a control-driven one
- Other functional complete sets can be used for extra expressivity!

Future Work

1. Decomposition operators
2. Less complex circuits in terms of size upon transformation
3. Characterisation of Boolean functions that can be represented via iteration
4. More expressive branching structures via:
   - Pre/Post-conditions
   - Probabilistic choice

Thank you for attending!