Shannon Effect for BC-complexity of Finite Automata

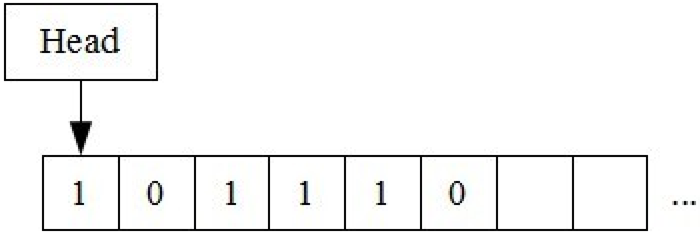
Māris Valdats
University of Latvia

2014.10.04

Outline of the talk

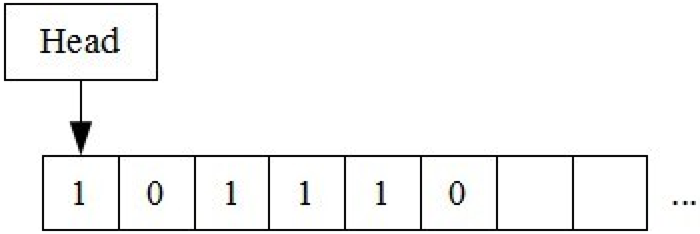
- Finite automata (DFA)
- Representation of automata with Boolean circuits
- BC-complexity
- Shannon effect for BC-complexity
- NFA, language operations
- Minimization

A finite automaton (DFA) consists of:

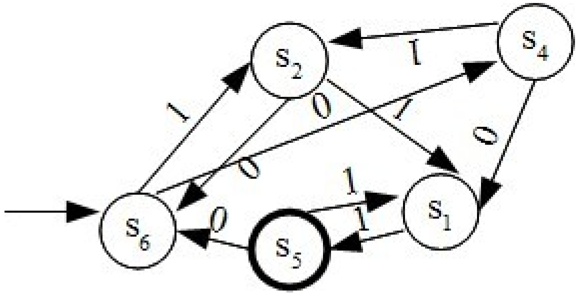


- Input tape
- Read-only head moving in only one direction
- On each step
  - -Read input symbol
  - -Change the state according to the transition function
  - -Move the head
- If there are no more input symbols
  - -If $q \in Q_F$ – accept word
  - -If $q \notin Q_F$ – reject word

A finite automaton (DFA) consists of:



- Input tape
- Read-only head moving in only one direction
- On each step
  - -Read input symbol
  - -Change the state according to the transition function
  - -Move the head
- If there are no more input symbols
  - -If $q \in Q_F$ – accept word
  - -If $q \notin Q_F$ – reject word



Finite Automata

A deterministic finite automaton (DFA) is a tuple $A(\Sigma, Q, Q_F, \delta, q_0)$ where

- Σ is the input alphabet
- Q is the state space
- $Q_F \subseteq Q$ is the set of final states
- $\delta : \Sigma \times Q \rightarrow Q$ is the transition function
- $q_0 \in Q$ is the start state

A deterministic finite automaton (DFA) is a tuple $A(\Sigma, Q, Q_F, \delta, q_0)$ where

- Σ is the input alphabet
- Q is the state space
- $Q_F \subseteq Q$ is the set of final states
- $\delta : \Sigma \times Q \rightarrow Q$ is the transition function
- $q_0 \in Q$ is the start state

Complexity measures of finite automata

A deterministic finite automaton (DFA) is a tuple $A(\Sigma, Q, Q_F, \delta, q_0)$ where

- Σ is the input alphabet
- Q is the state space
- $Q_F \subseteq Q$ is the set of final states
- $\delta : \Sigma \times Q \rightarrow Q$ is the transition function
- $q_0 \in Q$ is the start state

Complexity measures of finite automata

- State complexity $C_s(A)=|Q|$

Finite Automata

A deterministic finite automaton (DFA) is a tuple $A(\Sigma, Q, Q_F, \delta, q_0)$ where

- Σ is the input alphabet
- Q is the state space
- $Q_F \subseteq Q$ is the set of final states
- $\delta : \Sigma \times Q \rightarrow Q$ is the transition function
- $q_0 \in Q$ is the start state

Complexity measures of finite automata

- State complexity $C_s(A)=|Q|$
- ?

Motivation

What is the most complex automaton that we can build (model on a computer)?

Motivation

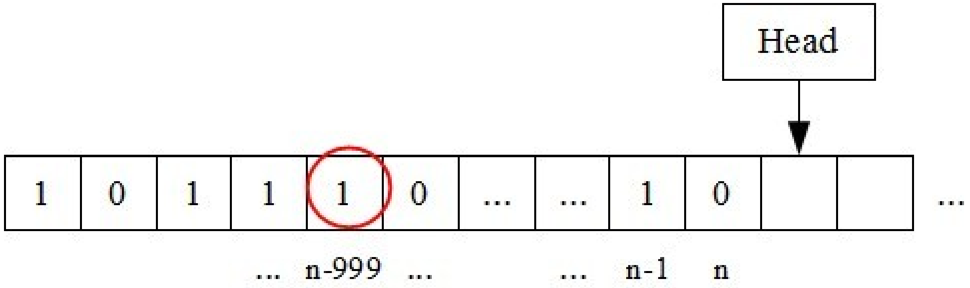
What is the most complex automaton that we can build (model on a computer)?

- 2^{10} states?
- 2^{100} states?
- 2^{1000} states?

Automata with 2^{1000} states

1. Automaton A_1 accepts language L_1 of words in a binary alphabet $\Sigma=\{0, 1\}$ for which 1000th digit from the end is "1".

$$\mathbf{x} \in L_1 \Leftrightarrow x_{|x|-999}=1$$

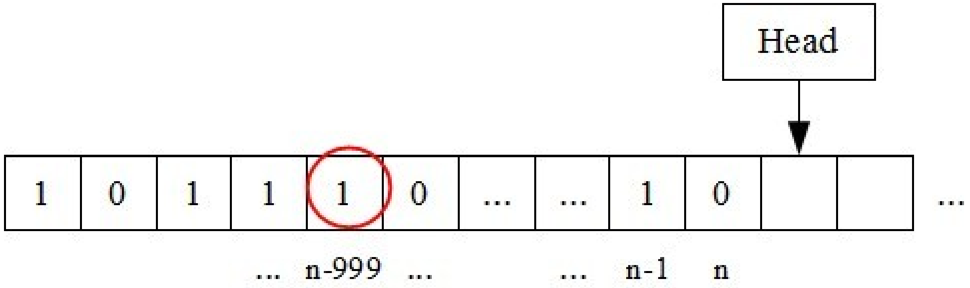


Automata with 2^{1000} states

1. Automaton A_1 accepts language L_1 of words in a binary alphabet $\Sigma=\{0, 1\}$ for which 1000th digit from the end is "1".

$$\mathbf{x}\in L_1 \Leftrightarrow x_{|x|-999}=1$$

- State complexity $C_S(A_1)=2^{1000}$
- Implementation – use 1000 bit LIFO register

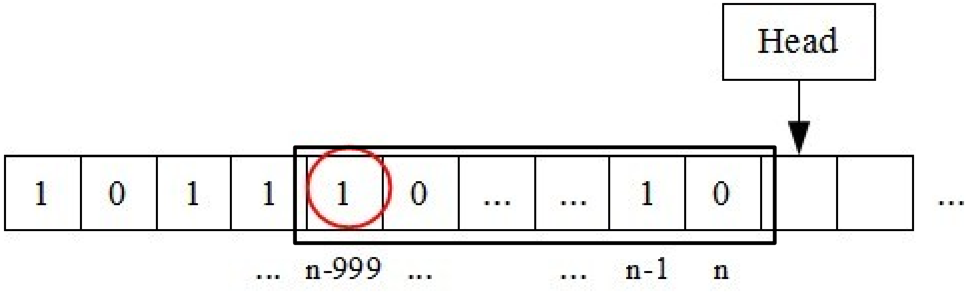


Automata with 2^{1000} states

1. Automaton A_1 accepts language L_1 of words in a binary alphabet $\Sigma=\{0, 1\}$ for which 1000th digit from the end is "1".

$$\mathbf{x}\in L_1 \Leftrightarrow x_{|x|-999}=1$$

- State complexity $C_S(A_1)=2^{1000}$
- Implementation – use 1000 bit LIFO register



Automata with 2^{1000} states

2. Automaton A_2 - a “random” 2^{1000} state automaton in a binary input alphabet.

Automata with 2^{1000} states

2. Automaton A_2 - a "random" 2^{1000} state automaton in a binary input alphabet.

- State complexity $C_S(A_2)=2^{1000}$
- Implementation – a table with 2^{1001} rows

Problem

How to show that the “random” automaton A_2 is more complex than A_1 if they have same state complexity?

Problem

How to show that the “random” automaton A_2 is more complex than A_1 if they have same state complexity?

Solution!

- Encode the state space into state register (bit vector)

Problem

How to show that the "random" automaton A_2 is more complex than A_1 if they have same state complexity?

Solution!

- Encode the state space into state register (bit vector)
- Measure not the complexity of the state space,

$$A(\Sigma, Q, Q_F, \delta, q_0)$$

Problem

How to show that the “random” automaton A_2 is more complex than A_1 if they have same state complexity?

Solution!

- Encode the state space into state register (bit vector)
- Measure not the complexity of the state space,

$$A(\Sigma, Q, Q_F, \delta, q_0)$$

Problem

How to show that the "random" automaton A_2 is more complex than A_1 if they have same state complexity?

Solution!

- Encode the state space into state register (bit vector)
- Measure not the complexity of the state space, but that of the transition function

$$A(\Sigma, Q, Q_F, \delta, q_0)$$

Problem

How to show that the “random” automaton A_2 is more complex than A_1 if they have same state complexity?

Solution!

- Encode the state space into state register (bit vector)
- Measure not the complexity of the state space, but that of the transition function

$$A(\Sigma, Q, Q_F, \delta, q_0)$$

Problem

How to show that the "random" automaton A_2 is more complex than A_1 if they have same state complexity?

Solution!

- Encode the state space into state register (bit vector)
- Measure not the complexity of the state space, but that of the transition function

$$A(\Sigma, Q, Q_F, \delta, q_0)$$

Representation of an automaton with a Boolean circuit

Representation of an automaton with a Boolean circuit

Encode:

Representation of an automaton with a Boolean circuit

Encode:

- State space Q → state register ($b_Q \geq \log|Q|$ state bits)

Representation of an automaton with a Boolean circuit

Encode:

- State space Q → state register ($b_Q \geq \log|Q|$ state bits)
- Input alphabet Σ → input bits ($b_\Sigma \geq \log|\Sigma|$ state bits)

Representation of an automaton with a Boolean circuit

Encode:

- State space Q → state register ($b_Q \geq \log|Q|$ state bits)
- Input alphabet Σ → input bits ($b_\Sigma \geq \log|\Sigma|$ state bits)

Represent:

Representation of an automaton with a Boolean circuit

Encode:

- State space Q → state register ($b_Q \geq \log|Q|$ state bits)
- Input alphabet Σ → input bits ($b_\Sigma \geq \log|\Sigma|$ state bits)

Represent:

- Transition function $\delta : \Sigma \times Q \rightarrow Q$ → Boolean circuit:
  - *Inputs* : *input bits and state bits*
  - *Outputs* : *state bits*

Representation of an automaton with a Boolean circuit

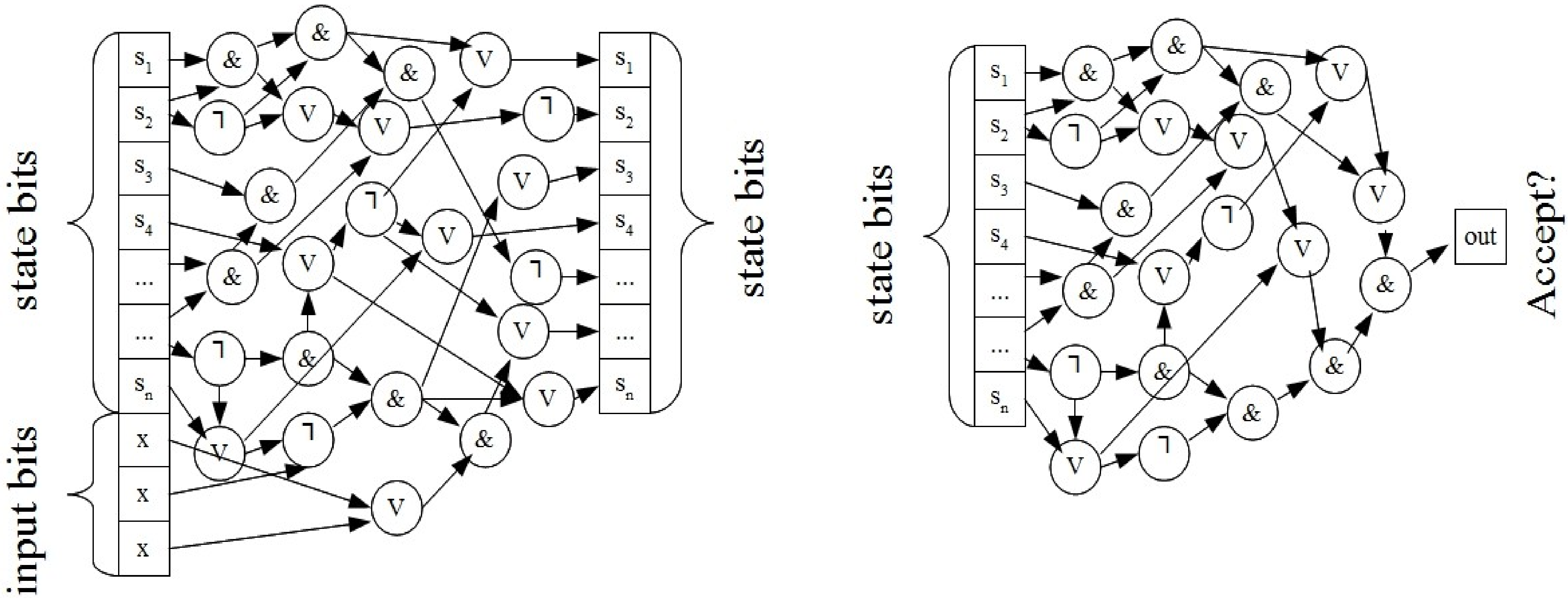
Encode:

- State space Q → state register ($b_Q \geq \log|Q|$ state bits)
- Input alphabet Σ → input bits ($b_\Sigma \geq \log|\Sigma|$ state bits)

Represent:

- Transition function $\delta : \Sigma \times Q \to Q$ → Boolean circuit:
  - *Inputs* : *input bits and state bits*
  - *Outputs* : *state bits*
- Set of final states $Q_F \subseteq Q$ → a Boolean circuit for the characteristic function of the set Q_F:
  - *Inputs* : *state bits*
  - *Outputs* : *one bit (accept/reject)*

Representation of an automaton with two Boolean circuits:



≥log|Q| state bits ≥log|Σ| input bits

Properties of circuit representation

Properties of circuit representation

- Each automaton can have infinitely many encodings

Properties of circuit representation

- Each automaton can have infinitely many encodings
- Each encoding can have infinitely many representations

Properties of circuit representation

- Each automaton can have infinitely many encodings
- Each encoding can have infinitely many representations
- (Number of state bits) $b_Q \geq \log_2(|Q|)$

Properties of circuit representation

- Each automaton can have infinitely many encodings
- Each encoding can have infinitely many representations
- (Number of state bits) $b_Q \geq \log_2(|Q|)$
- Two automata may have the same representation only if they are equivalent.

BC-complexity

DFA $\mathbf{A(\Sigma, Q, Q_F, \delta, q_0)}$ is represented by a pair of circuits **(F, G)**

BC-complexity

DFA **A(Σ, Q, Q_F , δ, q_0)** is represented by a pair of circuits **(F, G)**

BC-complexity of a representation

$C_{BC}((F, G)) = C(F)+C(G)+b_Q$

BC-complexity

DFA $\mathbf{A(\Sigma, Q, Q_F, \delta, q_0)}$ is represented by a pair of circuits **(F, G)**

BC-complexity of a representation

$C_{BC}((F, G)) = C(F)+C(G)+b_Q$

BC-complexity of an automaton

$C_{BC}(A)=\min\{C_{BC}(F, G): (F, G) \text{ represents } A\}$

BC-complexity

DFA $\mathbf{A(\Sigma, Q, Q_F, \delta, q_0)}$ is represented by a pair of circuits **(F, G)**

BC-complexity of a representation

$C_{BC}((F, G)) = C(F)+C(G)+b_Q$

BC-complexity of an automaton

$C_{BC}(A)=\min\{C_{BC}(F, G): (F, G) \text{ represents } A\}$

BC-complexity of a regular language

$C_{BC}(R)=\min\{C_{BC}(A): A \text{ recognizes } R\}$

Example 1

Automaton A_1 accepts language L_1 of words for which the *n* -th digit from the end is “1”:

- $\Sigma=\{0, 1\}$
- $C_s(A_1^n)=|Q|=2^n$

Example 1

Automaton A_1 accepts language L_1 of words for which the n -th digit from the end is “1”:

- $\Sigma=\{0, 1\}$
- $C_s(A_1^n)=|Q|=2^n$

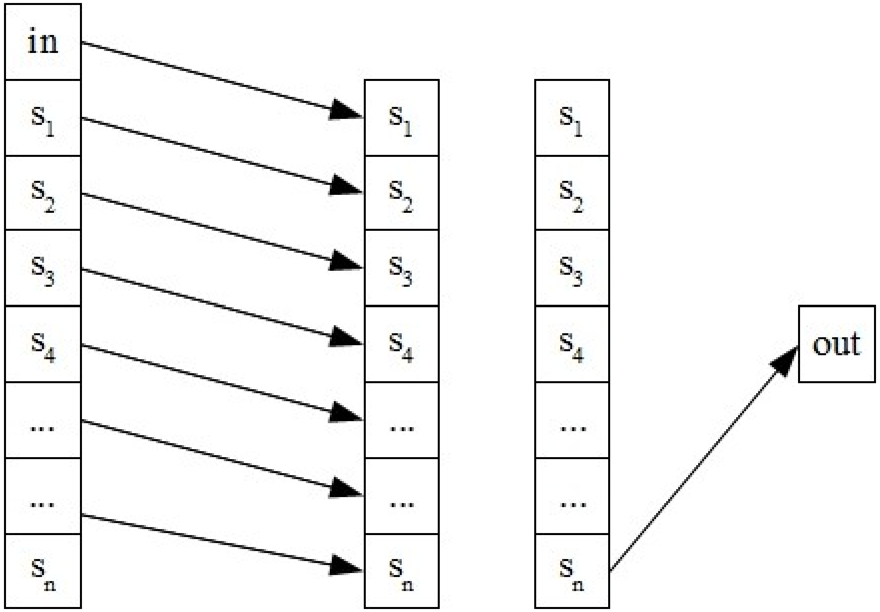
is represented by circuits:

Example 1

Automaton A_1 accepts language L_1 of words for which the n -th digit from the end is "1":

- $\Sigma=\{0, 1\}$
- $C_s(A_1^n)=|Q|=2^n$

is represented by circuits:

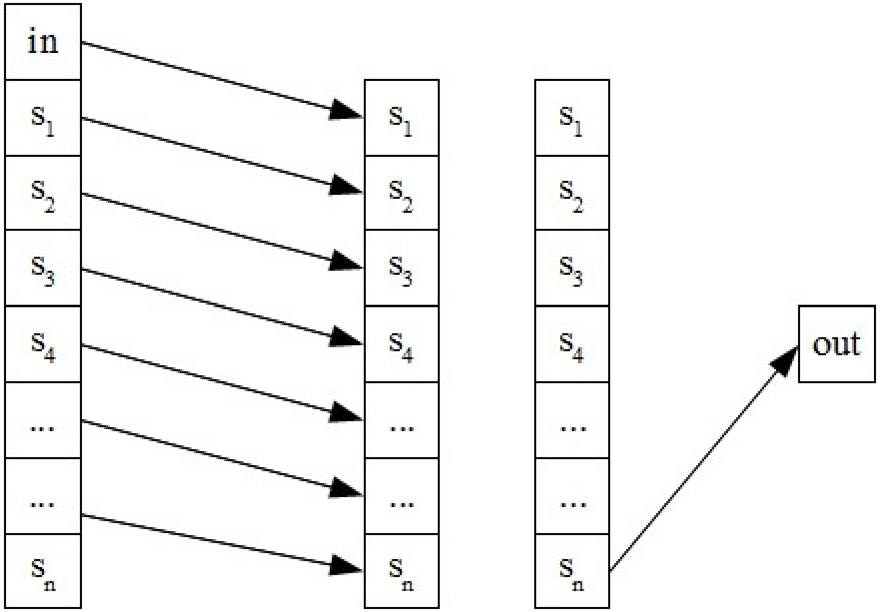


Example 1

Automaton A_1 accepts language L_1 of words for which the n -th digit from the end is “1”:

- $\Sigma=\{0, 1\}$
- $C_s(A_1^n)=|Q|=2^n$
- $C_{BC}(A)=0 + 0 + n = n$

is represented by circuits:



Example 2

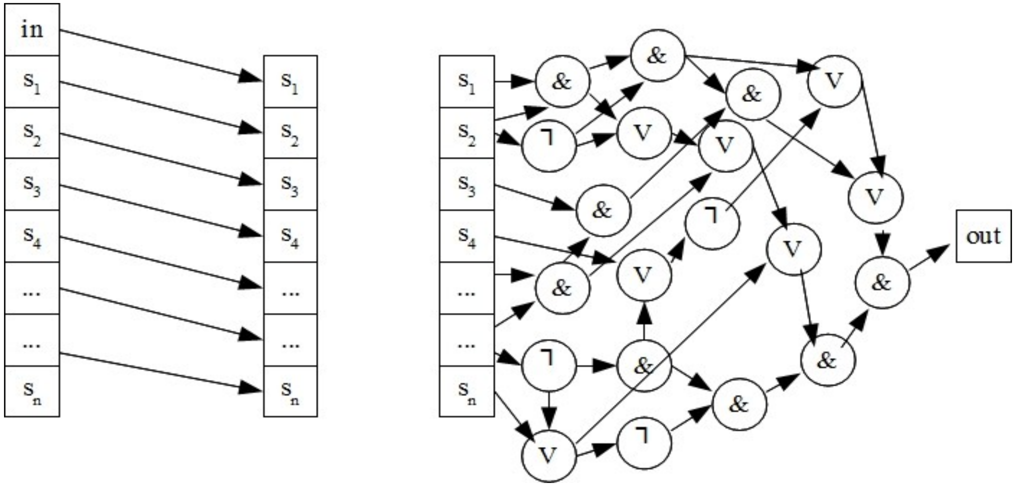
A finite automaton A_2^n, that accepts input iff the Shannon function of the last *n* input symbols is "1":

- $\Sigma=\{0, 1\}$
- $C_s(A_2^n)=|Q|=2^n$

Example 2

A finite automaton A_2^n, that accepts input iff the Shannon function of the last n input symbols is “1”:

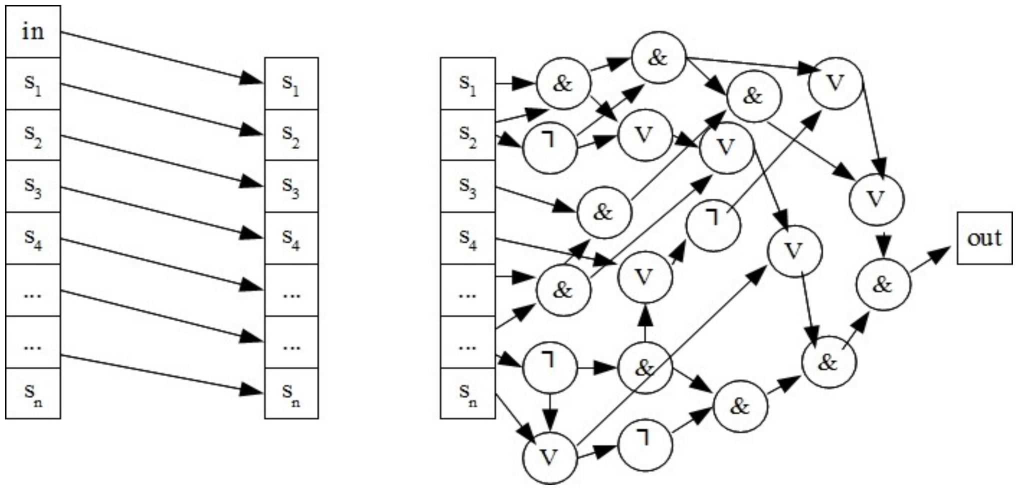
- $\Sigma=\{0, 1\}$
- $C_s(A_2^n)=|Q|=2^n$



Example 2

A finite automaton A_2^n, that accepts input iff the Shannon function of the last n input symbols is “1”:

- $\Sigma=\{0, 1\}$
- $C_s(A_2^n)=|Q|=2^n$
- $C_{BC}((F, G)) \geq 0 + 2^n/n + n \geq 2^n/n$

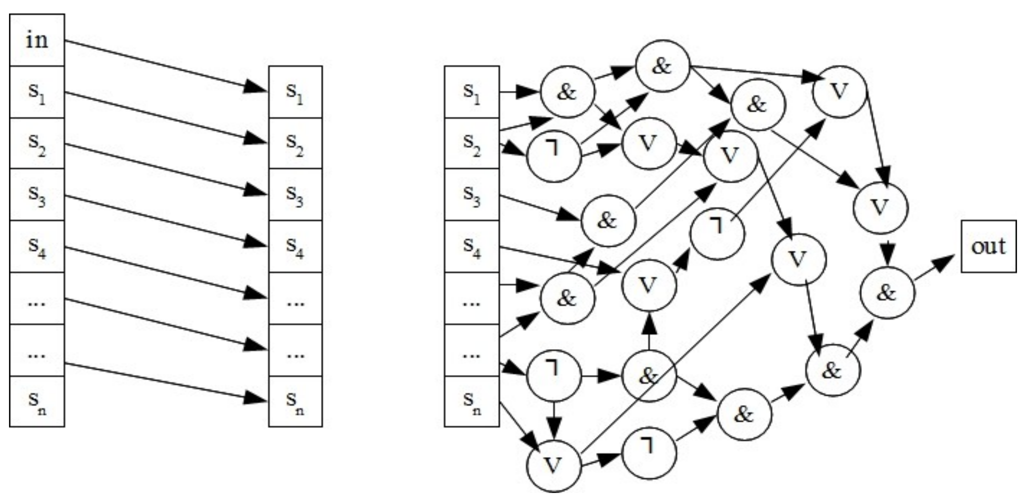


BC-complexity

Example 2

A finite automaton A_2^n, that accepts input iff the Shannon function of the last n input symbols is "1":

- $\Sigma=\{0, 1\}$
- $C_s(A_2^n)=|Q|=2^n$
- $C_{BC}((F, G)) \geq 0 + 2^n/n + n \geq 2^n/n$
- $C_{BC}(A_2^n)\geq 2^n/n^2$ (proof omitted)



Upper and lower bounds for BC-complexity

Upper and lower bounds for BC-complexity

- $C_{BC}(F, G) = C(F) + C(G) + b_Q$
- $|\Sigma|=k$ and $|Q|=2^n$:

Upper and lower bounds for BC-complexity

- $C_{BC}(F, G) = C(F) + C(G) + b_Q$
- $|\Sigma|=k$ and $|Q|=2^n$:

Lower bound: $n \leq C_{BC}(A)$

Upper and lower bounds for BC-complexity

- $C_{BC}(F, G) = C(F) + C(G) + b_Q$
- $|\Sigma|=k$ and $|Q|=2^n$:

  Lower bound: $n \leq C_{BC}(\mathcal{A})$

  Upper bound (simple) $C_{BC}(\mathcal{A}) \lesssim k2^n + 2^n/n+n \lesssim k2^n$

Upper and lower bounds for BC-complexity

- $C_{BC}(F, G) = C(F) + C(G) + b_Q$
- $|\Sigma|=k$ and $|Q|=2^n$:

Lower bound: $n \leq C_{BC}(A)$

Upper bound (simple) $C_{BC}(A) \lesssim k2^n + 2^n/n + n \lesssim k2^n$

Change the encoding (reorder states)

Upper and lower bounds for BC-complexity

- $C_{BC}(F, G) = C(F) + C(G) + b_Q$
- $|\Sigma|=k$ and $|Q|=2^n$:

Lower bound: $n \leq C_{BC}(\mathcal{A})$

Upper bound (simple) $C_{BC}(\mathcal{A}) \lesssim k2^n + 2^n/n + n \lesssim k2^n$

Change the encoding (reorder states)

Upper bound $C_{BC}(\mathcal{A}) \lesssim (k-1)2^n$

Upper and lower bounds for BC-complexity

- $C_{BC}(F, G) = C(F) + C(G) + b_Q$
- $|\Sigma|=k$ and $|Q|=2^n$:

  Lower bound: $n \leq C_{BC}(A)$

  Upper bound (simple) $C_{BC}(A) \lesssim k2^n + 2^n/n + n \lesssim k2^n$

  Change the encoding (reorder states)

  Upper bound $C_{BC}(A) \lesssim (k-1)2^n$

  Counting argument

Upper and lower bounds for BC-complexity

- $C_{BC}(F, G) = C(F) + C(G) + b_Q$
- $|\Sigma|=k$ and $|Q|=2^n$:

Lower bound: $n \leq C_{BC}(A)$

Upper bound (simple) $C_{BC}(A) \lesssim k2^n + 2^n/n + n \lesssim k2^n$

Change the encoding (reorder states)

Upper bound $C_{BC}(A) \lesssim (k-1)2^n$

Counting argument

For almost all *A* $C_{BC}(A) \gtrsim (k-1)2^n$

Upper and lower bounds for BC-complexity

- $C_{BC}(F, G) = C(F) + C(G) + b_Q$
- $|\Sigma|=k$ and $|Q|=2^n$:

Lower bound: $n \leq C_{BC}(A)$

Upper bound (simple) $C_{BC}(A) \lesssim k2^n + 2^n/n + n \lesssim k2^n$

Change the encoding (reorder states)

Upper bound $C_{BC}(A) \lesssim (k-1)2^n$

Counting argument

For almost all *A* $C_{BC}(A) \gtrsim (k-1)2^n$

$x \gtrsim f(n) \Leftrightarrow x > f(n)(1-o(1))$

BC-complexity

"Shannon effect" for the BC-complexity of DFA.

- $C_{BC}(F, G) = C(F) + C(G) + b_Q$
- $|\Sigma|=k$ and $|Q|=2^n$:

BC-complexity

“Shannon effect” for the BC-complexity of DFA.

- $C_{BC}(F, G) = C(F) + C(G) + b_Q$
- $|\Sigma|=k$ and $|Q|=2^n$:

For almost all *A* $C_{BC}(A) \approx (k-1)2^n$

BC-complexity

“Shannon effect” for the BC-complexity of DFA.

- $C_{BC}(F, G) = C(F) + C(G) + b_Q$
- $|\Sigma|=k$ and $|Q|=2^n$:

For almost all A $\quad C_{BC}(A) \approx (k-1)2^n$

- $C_{BC}(A_1^n) = n$
- $C_{BC}(A_2^n) \geq 2^n/n^2$

Some special cases

- Nondeterministic automata
- Language operations

Nondeterministic automata (NFA)

Nondeterministic automata (NFA)

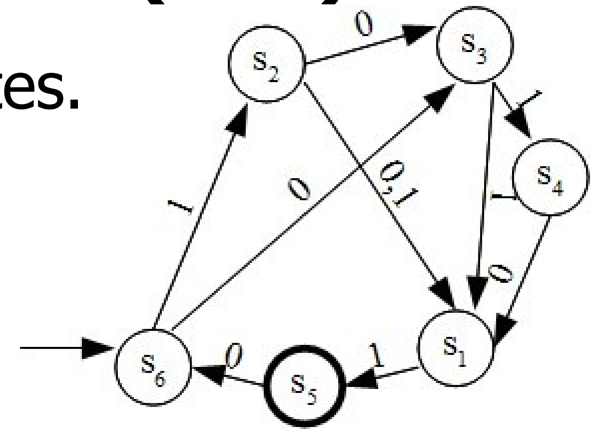
- NFA with *n* states → DFA *A* with 2^n states.

Nondeterministic automata (NFA)

- NFA with *n* states → DFA *A* with 2^n states.
- $C_{BC}(A) \lesssim (k-1)2^n$

Nondeterministic automata (NFA)

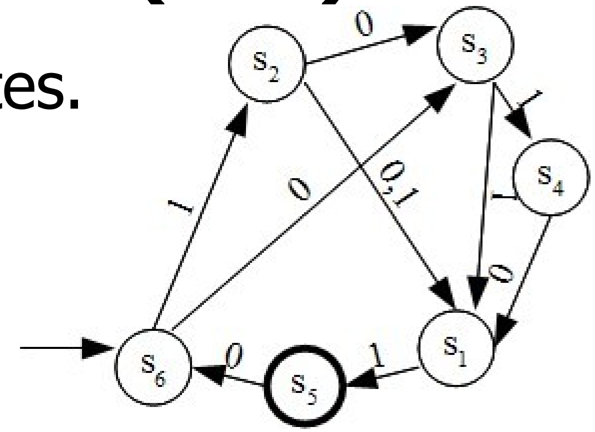
- NFA with *n* states → DFA *A* with 2^n states.
- $C_{BC}(A) \lesssim (k-1)2^n$



Nondeterministic automata (NFA)

- NFA with *n* states → DFA *A* with 2^n states.
- $C_{BC}(A) \lesssim (k-1)2^n$

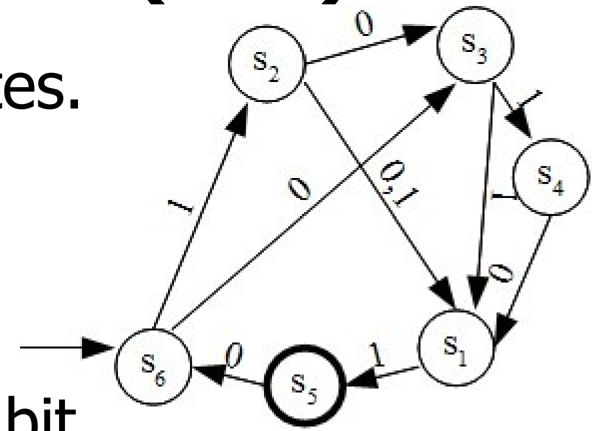
We can do much better!



Nondeterministic automata (NFA)

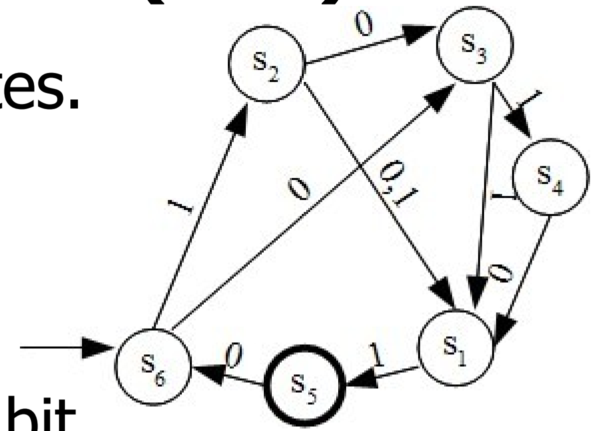
- NFA with *n* states → DFA *A* with 2^n states.
- $C_{BC}(A) \lesssim (k-1)2^n$

We can do much better!



- Encode one state of NFA as one state bit.

Nondeterministic automata (NFA)

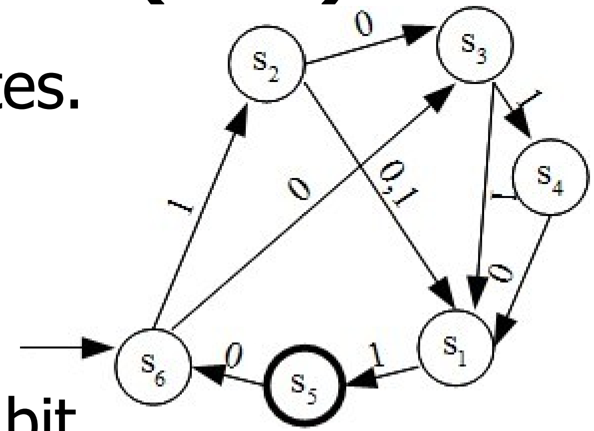


- NFA with *n* states → DFA *A* with 2^n states.
- $C_{BC}(A) \lesssim (k-1)2^n$

We can do much better!

- Encode one state of NFA as one state bit.
- $s_1' = 1 \Leftrightarrow (x=1 \ \& \ (s_2=1 \lor s_3=1)) \lor (x=0 \ \& \ (s_2=1 \lor s_4=1))$

Nondeterministic automata (NFA)



- NFA with *n* states → DFA *A* with 2^n states.
- $C_{BC}(A) \lesssim (k-1)2^n$

We can do much better!

- Encode one state of NFA as one state bit.
- $s_1' = 1 \Leftrightarrow (x=1 \;\&\; (s_2=1 \vee s_3=1)) \vee (x=0 \;\&\; (s_2=1 \vee s_4=1))$

Much simpler than a general function on *n* arguments!

Nondeterministic automata (NFA)

- NFA with n states → DFA A with 2^n states.
- $C_{BC}(A) \lesssim (k-1)2^n$ (from the upper bound)

Nondeterministic automata (NFA)

- NFA with *n* states → DFA *A* with 2^n states.
- $C_{BC}(A) \lesssim (k-1)2^n$ (from the upper bound)
- $C_{BC}(A) < kn^2$ (simple construction)

Nondeterministic automata (NFA)

- NFA with n states → DFA A with 2^n states.
- $C_{BC}(A) \lesssim (k-1)2^n$ (from the upper bound)
- $C_{BC}(A) < kn^2$ (simple construction)
- $C_{BC}(A) \lesssim kn^2/\log n$ (nearly optimal construction)

Nondeterministic automata (NFA)

- NFA with n states → DFA A with 2^n states.
- $C_{BC}(A) \lesssim (k-1)2^n$ (from the upper bound)
- $C_{BC}(A) < kn^2$ (simple construction)
- $C_{BC}(A) \lesssim kn^2/\log n$ (nearly optimal construction)

Counting argument

For almost all A $\quad C_{BC}(A) \gtrsim (k-1)n^2/2\log n$

Nondeterministic automata (NFA)

- NFA with n states → DFA A with 2^n states.
- $C_{BC}(A) \lesssim (k-1)2^n$ (from the upper bound)
- $C_{BC}(A) < kn^2$ (simple construction)
- $C_{BC}(A) \lesssim kn^2/\log n$ (nearly optimal construction)

Counting argument

For almost all A $C_{BC}(A) \gtrsim (k-1)n^2/2\log n$

No Shannon effect right now for NFA (but coming close)

Upper bounds of BC-complexity for Language operations

- Given two languages L_1 and L_2 with state complexities *m* and *n* and BC-complexities *a* and *b*.

Operation	State comp.	BC-complexity
$L_1 \cup L_2$	mn	a+b+1
$L_1 \cap L_2$	mn	a+b+1
Σ^*-L_1	m	a+1
L_1^R	2^m	2m(m+1)
L_1L_2	$(2m-1)2^{n-1}$	2a+2n(n+1)
L_1^*	$2^{m-1}+2^{m-2}$	2m(m+1)

A computer

A computer

- Is computer a Turing machine?

A computer

- Is computer a Turing machine?
- Is computer a finite automaton?

A computer

- Is computer a Turing machine?
- Is computer a finite automaton?

On each step it:

- reads input (may be nothing),
- transforms its registers and memory according to some simple function,
- has state space with around $2^{2^{40}}$ states (2^{40} state bits)

A computer

- Is computer a Turing machine?
- Is computer a finite automaton?

On each step it:

- reads input (may be nothing),
- transforms its registers and memory according to some simple function,
- has state space with around $2^{2^{40}}$ states (2^{40} state bits)

- Is computer (an efficient) representation of a finite automaton?

Minimization

Minimization

- For every automaton *A* one can find its minimal (with the respect to the number of states) automaton M(*A*)

Minimization

- For every automaton *A* one can find its minimal (with the respect to the number of states) automaton M(*A*)
  - Is the BC-complexity of M(*A*) also minimal?

Minimization

- For every automaton *A* one can find its minimal (with the respect to the number of states) automaton M(*A*)
  - Is the BC-complexity of M(*A*) also minimal?
  - Can it be that for some automaton *A*: $C_{BC}(M(A)) \gg C_{BC}(A)$?

Minimization

Theorem:
If there exists a polynomial $p(x)$ such that
$C_{BC}(M(A)) < p(C_{BC}(A))$
for all automata A then PSPACE $\subseteq$ P/Poly

Minimization

Theorem:
If there exists a polynomial $p(x)$ such that
$C_{BC}(M(A))<p(C_{BC}(A))$
for all automata A then PSPACE $\subseteq$ P/Poly

For every n we can build an automaton B_n with

- Poly(n) state bits ($2^{Poly(n)}$ states)
- $C_{BC}(B_n)=Poly(n)$
- $C_{BC}(M(B_n))\notin Poly(n)$

Kolmogorov complexity of automata

Kolmogorov complexity of automata

$C_K(A)$=minimal size of a program that outputs state transition table of automaton A

Kolmogorov complexity of automata

$C_K(A)$=minimal size of a program that outputs state transition table of automaton A

- $C_K(A)$: efficient description of A
- $C_{BC}(A)$: efficient execution of A

Kolmogorov complexity of automata

$C_K(A)$=minimal size of a program that outputs state transition table of automaton A

- $C_K(A)$: efficient description of A
- $C_{BC}(A)$: efficient execution of A

There is a constant c such that
$C_K(M(A)) \leq C_K(A)+c$
for all automata A.

Kolmogorov complexity of automata

$C_K(A)$=minimal size of a program that outputs state transition table of automaton A

- $C_K(A)$: efficient description of A
- $C_{BC}(A)$: efficient execution of A

There is a constant c such that
$C_K(M(A)) \leq C_K(A)+c$
for all automata A.

If there exists a polynomial $p(x)$ such that
$C_{BC}(M(A))<p(C_{BC}(A))$
for all automata A then PSPACE $\subseteq$ P/Poly

Conclusions

Conclusions

"What is the most complex automaton that we can build (or model on a computer)"?

Conclusions

"What is the most complex automaton that we can build (or model on a computer)"?

- We can model any automaton with a reasonable BC-complexity
- Many "naturally generated" DFAs have large state complexity but low BC complexity
- Sometimes minimizing the number of states leads to (a large) increase in BC-complexity

Open questions

Open questions

Minimizing the number of states is not always optimal for achieving minimal BC-complexity.

Open questions

Minimizing the number of states is not always optimal for achieving minimal BC-complexity.

- Is it always true that representation with minimal (log(|Q|) number of state bits is optimal?

Open questions

Minimizing the number of states is not always optimal for achieving minimal BC-complexity.

- Is it always true that representation with minimal (log(|Q|) number of state bits is optimal?
- Can the upper and lower bounds for NFA be improved?

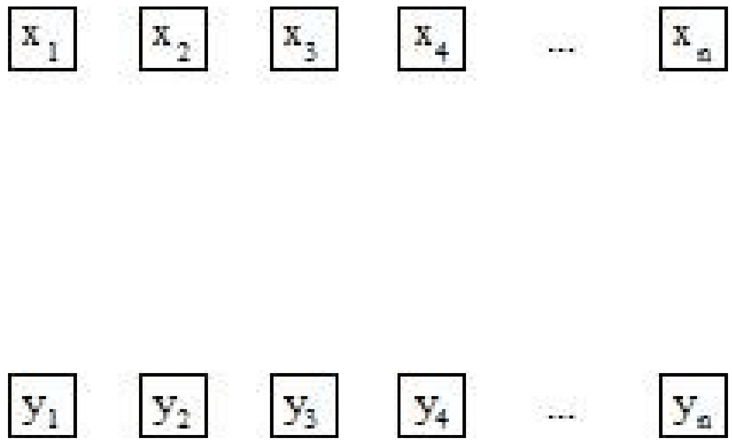
Open questions

Minimizing the number of states is not always optimal for achieving minimal BC-complexity.

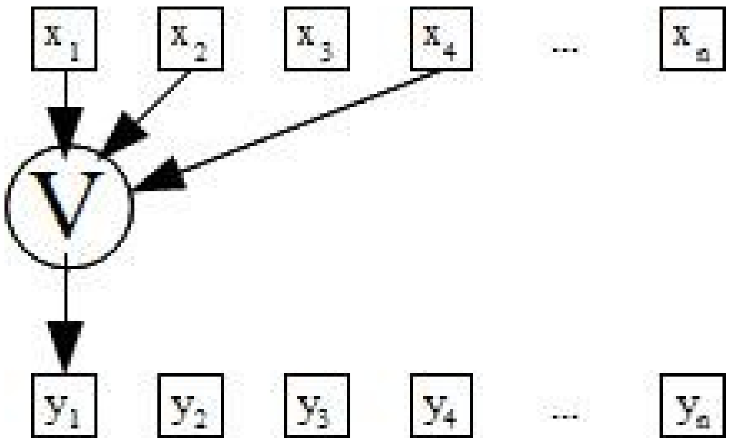
- Is it always true that representation with minimal (log(|Q|) number of state bits is optimal?
- Can the upper and lower bounds for NFA be improved?
- How to estimate the lower bounds for language operations?

Open questions

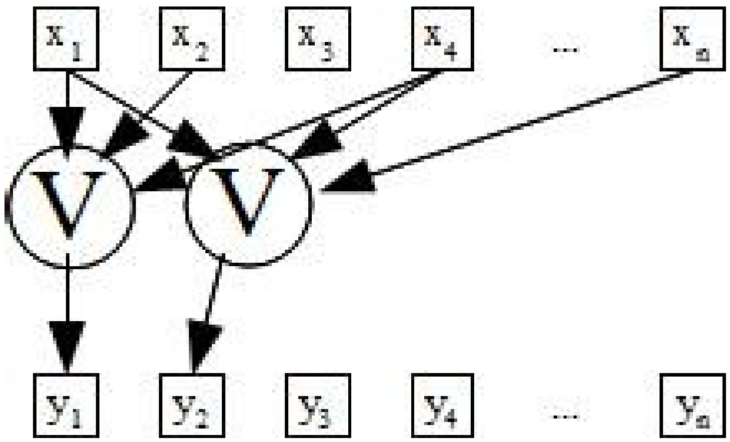
Open questions



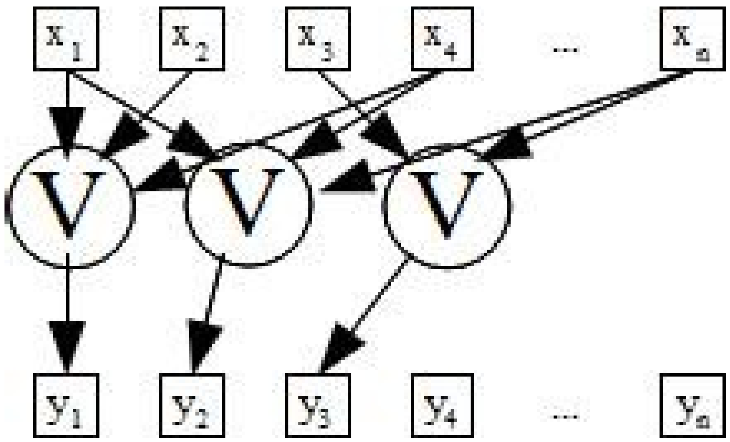
Open questions



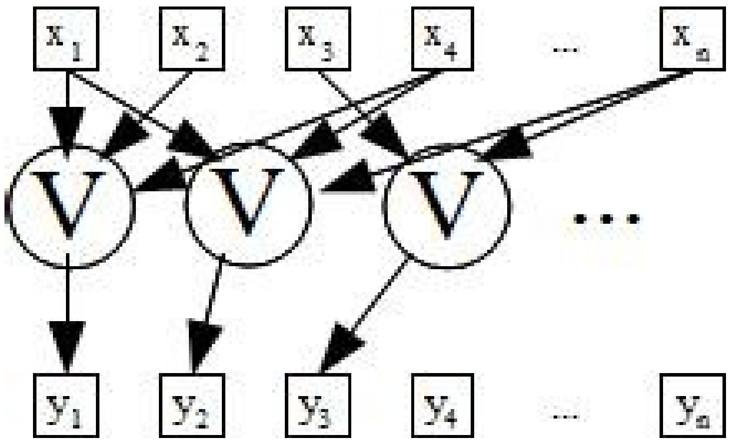
Open questions



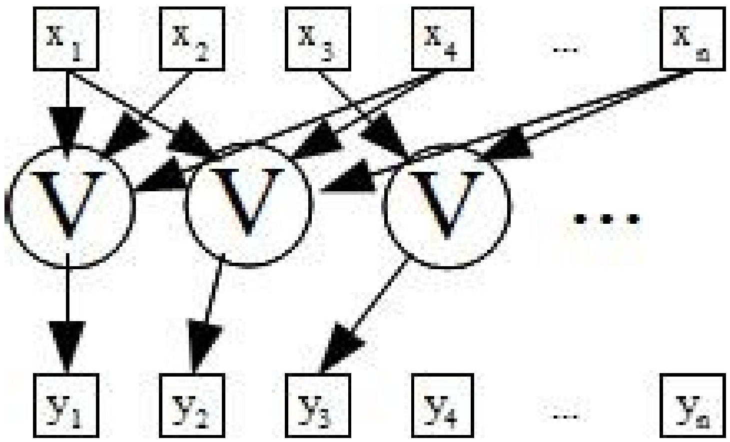
Open questions



Open questions

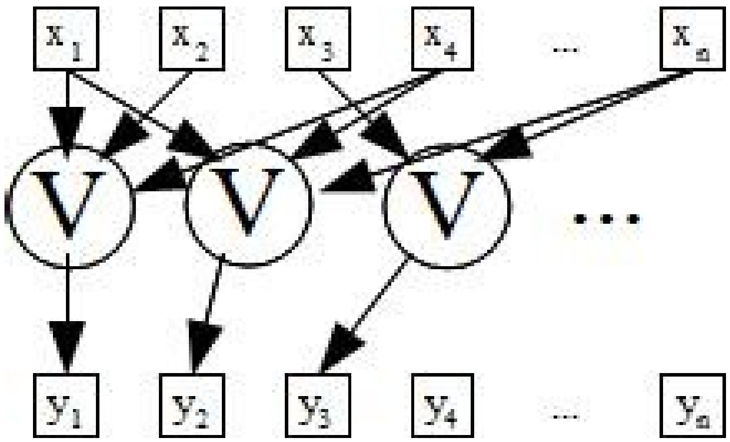


Open questions



$y_i = x_{j1} \vee x_{j2} \vee \ldots \vee x_{jk}$

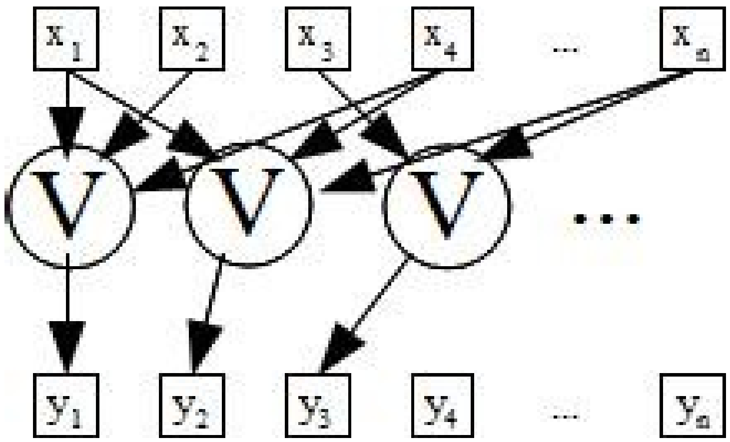
Open questions



$y_i = x_{j1} \vee x_{j2} \vee \ldots \vee x_{jk}$

- Simple contruction needs on average $n^2/2$ gates

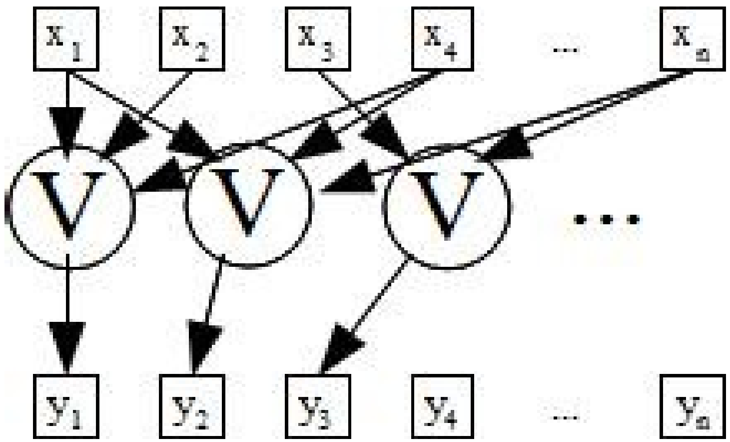
Open questions



$y_i = x_{j1} \vee x_{j2} \vee \ldots \vee x_{jk}$

- Simple contruction needs on average $n^2/2$ gates
- More efficient contruction needs asymptotically $n^2/\log n$ gates

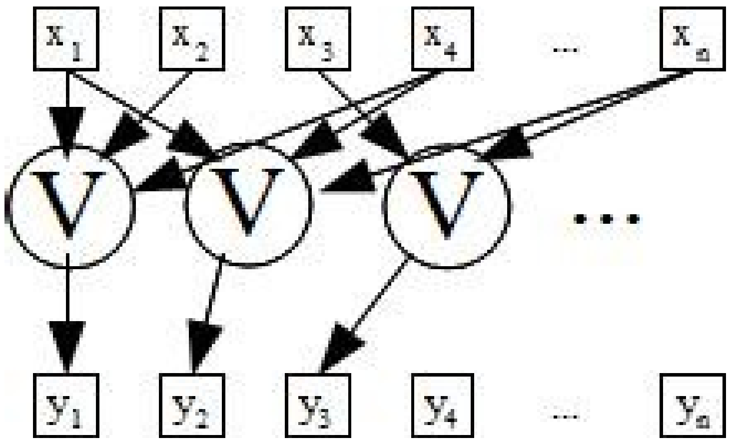
Open questions



$y_i = x_{j1} \vee x_{j2} \vee \ldots \vee x_{jk}$

- Simple contruction needs on average $n^2/2$ gates
- More efficient contruction needs asymptotically $n^2/\log n$ gates
- Can you do better?

Open questions



$y_i = x_{j1} \vee x_{j2} \vee \ldots \vee x_{jk}$

- Simple contruction needs on average $n^2/2$ gates
- More efficient contruction needs asymptotically $n^2/\log n$ gates
- Can you do better?
- Lower bounds is $n^2/2\log n$ gates

Thank you!