# A subexponential lower bound for Zadeh's pivoting rule for solving linear programs and games

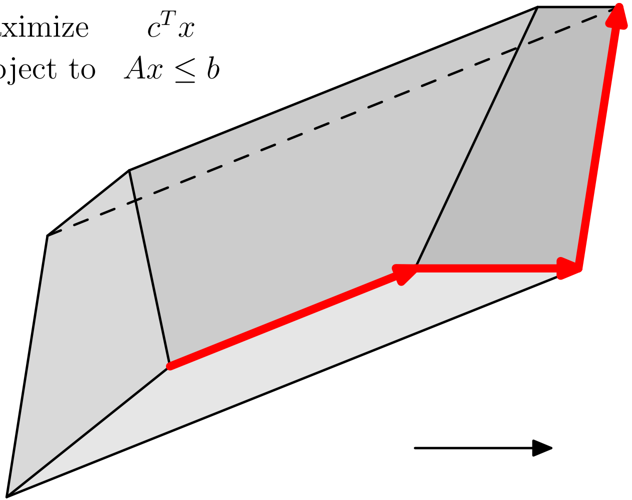
Oliver Friedmann

Department of Computer Science,
Ludwig-Maximilians-Universität Munich, Germany.

# Linear Programming and Simplex

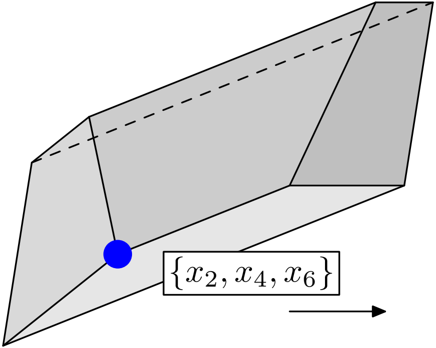
# The simplex algorithm, Dantzig (1947)

$$\begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & Ax \leq b \end{array}$$



# Basic feasible solutions and pivoting

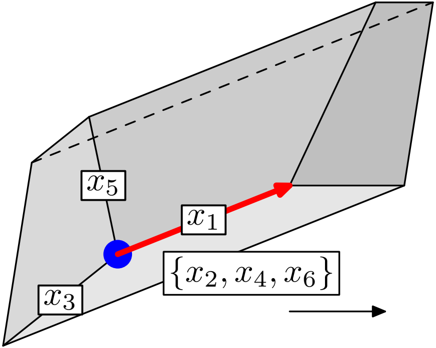
$$
\begin{array}{lrcl}
\max & 2x_1 - 2x_3 - 2x_5 - x_6 & & \\
\text{s.t.} & \frac{1}{3}x_1 + x_2 - \frac{2}{3}x_3 - \frac{2}{3}x_5 & = & 1 \\
& x_3 + x_4 - x_6 & = & 1 \\
& x_5 + x_6 & = & 1 \\
& x_1, x_2, x_3, x_4, x_5, x_6 & \geq & 0
\end{array}
$$




- The corners of the polytope correspond to **basic feasible solutions**: At most $n$, the number of equality constraints, variables are non-zero. The non-zero variables, or *basic* variables, form a *basis*.
- Moving along an edge corresponds to **pivoting**: Exchange a variable in the basis with a non-basic variable.

# Basic feasible solutions and pivoting

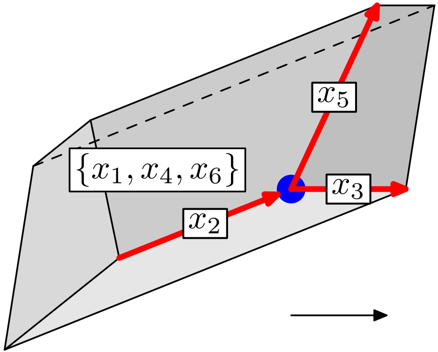
$$\begin{array}{lrcl} \max & \multicolumn{3}{l}{-1 + 2x_1 - 2x_3 - x_5} \\ \text{s.t.} & x_2 & = & 1 - \frac{1}{3}x_1 + \frac{2}{3}x_3 + \frac{2}{3}x_5 \\ & x_4 & = & 2 - x_3 - x_5 \\ & x_6 & = & 1 - x_5 \\ & \multicolumn{3}{l}{x_1, x_2, x_3, x_4, x_5, x_6 \geq 0} \end{array}$$




- The corners of the polytope correspond to **basic feasible solutions**: At most $n$, the number of equality constraints, variables are non-zero. The non-zero variables, or *basic* variables, form a *basis*.
- Moving along an edge corresponds to **pivoting**: Exchange a variable in the basis with a non-basic variable.

# Basic feasible solutions and pivoting

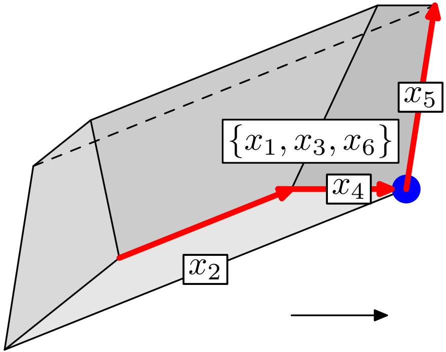
$$\begin{array}{lrcl} \max & \multicolumn{3}{l}{5 - 6x_2 + 2x_3 + 3x_5} \\ \text{s.t.} & x_1 & = & 3 - 3x_2 + 2x_3 + 2x_5 \\ & x_4 & = & 2 - x_3 - x_5 \\ & x_6 & = & 1 - x_5 \\ & \multicolumn{3}{l}{x_1, x_2, x_3, x_4, x_5, x_6 \geq 0} \end{array}$$




- The corners of the polytope correspond to **basic feasible solutions**: At most $n$, the number of equality constraints, variables are non-zero. The non-zero variables, or *basic* variables, form a *basis*.
- Moving along an edge corresponds to **pivoting**: Exchange a variable in the basis with a non-basic variable.

# Basic feasible solutions and pivoting

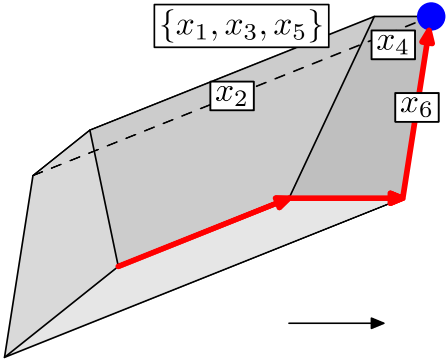
$$\begin{array}{lrcl} \max & \multicolumn{3}{l}{9 - 6x_2 - 2x_4 + x_5} \\ \text{s.t.} & x_1 & = & 7 - 3x_2 - 2x_4 \\ & x_3 & = & 2 - x_4 - x_5 \\ & x_6 & = & 1 - x_5 \\ & \multicolumn{3}{l}{x_1, x_2, x_3, x_4, x_5, x_6 \geq 0} \end{array}$$




- The corners of the polytope correspond to **basic feasible solutions**: At most $n$, the number of equality constraints, variables are non-zero. The non-zero variables, or *basic* variables, form a *basis*.
- Moving along an edge corresponds to **pivoting**: Exchange a variable in the basis with a non-basic variable.

# Basic feasible solutions and pivoting

$$\begin{array}{ll} \max & 10 - 6x_2 - 2x_4 - x_6 \\ \text{s.t.} & x_1 = 7 - 3x_2 - 2x_4 \\ & x_3 = 1 - x_4 + x_6 \\ & x_5 = 1 - x_6 \\ & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \end{array}$$




- The corners of the polytope correspond to **basic feasible solutions**: At most $n$, the number of equality constraints, variables are non-zero. The non-zero variables, or *basic* variables, form a *basis*.
- Moving along an edge corresponds to **pivoting**: Exchange a variable in the basis with a non-basic variable.

# Pivoting rules

Simplex method is parameterized by a pivoting rule: the method of chosing adjacent vertices with better objective

# Pivoting rules

Simplex method is parameterized by a pivoting rule: the method of chosing adjacent vertices with better objective

- Deterministic, oblivious rules: Almost all known natural oblivious deterministic pivoting rules are known to be exponential. See, e.g., Amenta and Ziegler (1996).

# Pivoting rules

Simplex method is parameterized by a pivoting rule: the method of chosing adjacent vertices with better objective

- Deterministic, oblivious rules: Almost all known natural oblivious deterministic pivoting rules are known to be exponential. See, e.g., Amenta and Ziegler (1996).
- Randomized rules: We (joint work with Thomas D. Hansen and Uri Zwick) have recently shown that Random-Edge and Random-Facet are subexponential.

# Pivoting rules

Simplex method is parameterized by a pivoting rule: the method of chosing adjacent vertices with better objective

- Deterministic, oblivious rules: Almost all known natural oblivious deterministic pivoting rules are known to be exponential. See, e.g., Amenta and Ziegler (1996).
- Randomized rules: We (joint work with Thomas D. Hansen and Uri Zwick) have recently shown that RANDOM-EDGE and RANDOM-FACET are subexponential.
- History-based rules (this talk): We show that Zadeh's LEAST-ENTERED rule is subexponential.

# Pivoting rules

Simplex method is parameterized by a pivoting rule: the method of chosing adjacent vertices with better objective

- Deterministic, oblivious rules: Almost all known natural oblivious deterministic pivoting rules are known to be exponential. See, e.g., Amenta and Ziegler (1996).
- Randomized rules: We (joint work with Thomas D. Hansen and Uri Zwick) have recently shown that RANDOM-EDGE and RANDOM-FACET are subexponential.
- History-based rules (this talk): We show that Zadeh's LEAST-ENTERED rule is subexponential.

Our results have been obtained by using a deep relation between algorithmic game theory and linear programming.

# On the diameter of polytopes

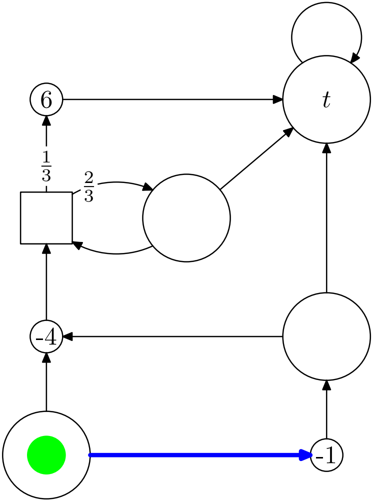
- Hirsch conjecture (1957): The diameter of any $n$-facet polytope in $d$-dimensional Euclidean space is at most $n - d$.

# On the diameter of polytopes

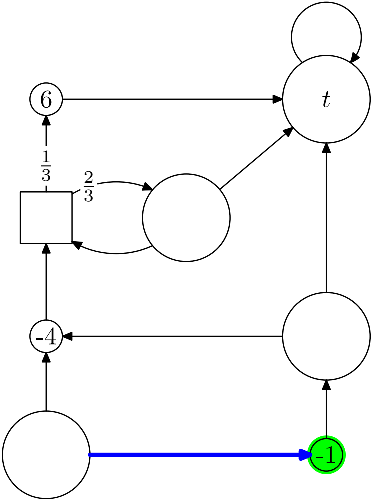
- Hirsch conjecture (1957): The diameter of any $n$-facet polytope in $d$-dimensional Euclidean space is at most $n - d$.
- Santos (2010): A counter-example to the Hirsch conjecture.
  - It remains open whether the diameter is polynomial, or even linear, in $n$ and $d$.

# Games and Policy Iteration

# Markov decision processes (MDPs)

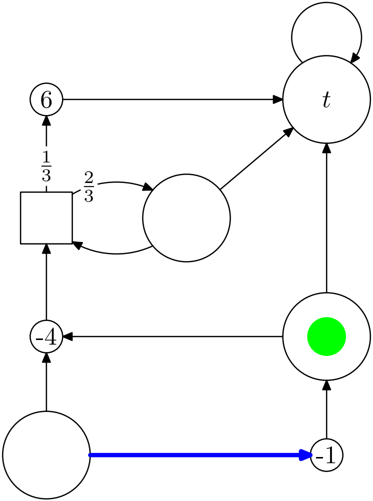


# Markov decision processes (MDPs)



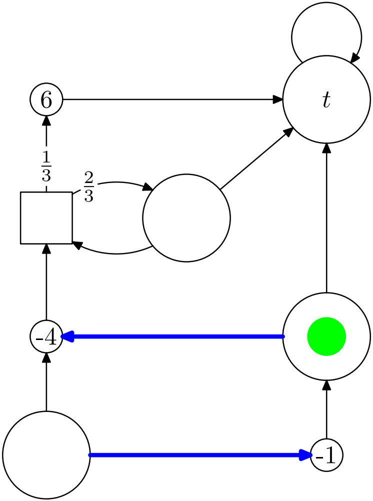

Reward: $-1$

# Markov decision processes (MDPs)



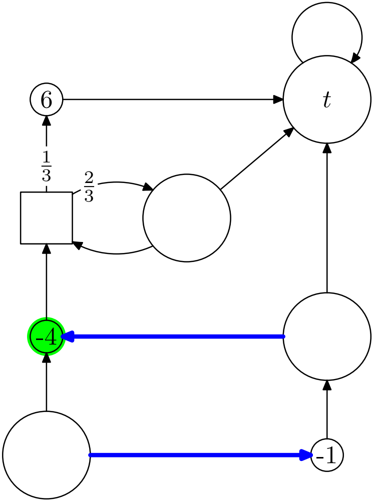

Reward: $-1$

# Markov decision processes (MDPs)



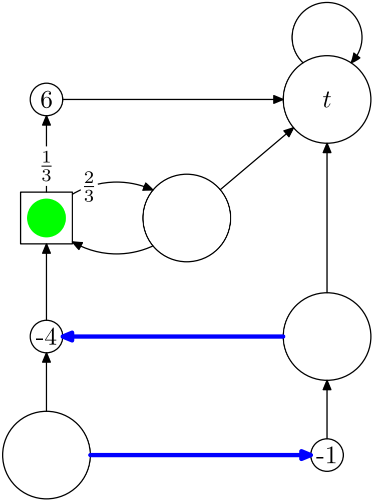

Reward: $-1$

# Markov decision processes (MDPs)



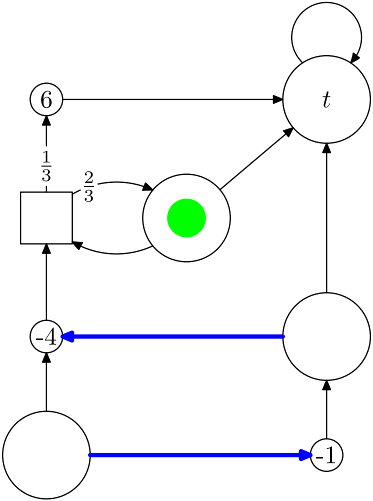

Reward: $-1 - 4$

# Markov decision processes (MDPs)



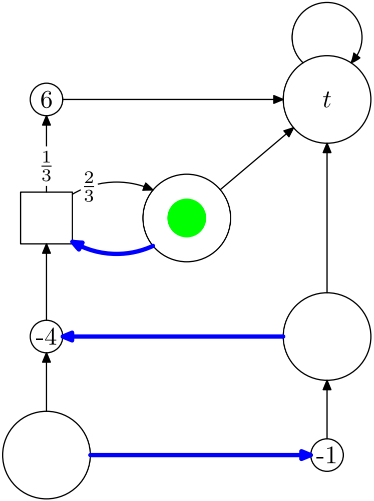

Reward: $-1 - 4$

# Markov decision processes (MDPs)



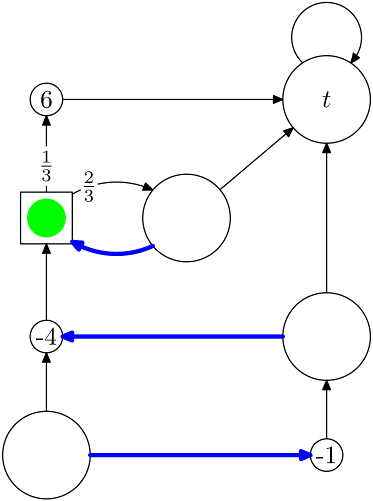

Reward: $-1 - 4$

# Markov decision processes (MDPs)



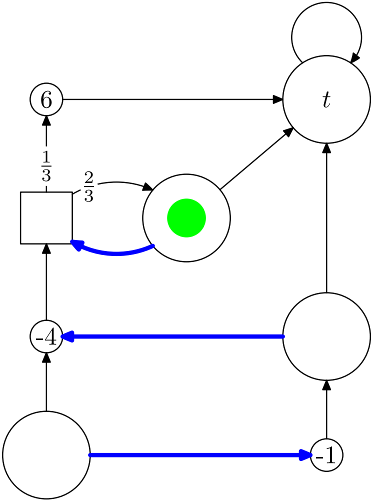

Reward: $-1 - 4$

# Markov decision processes (MDPs)



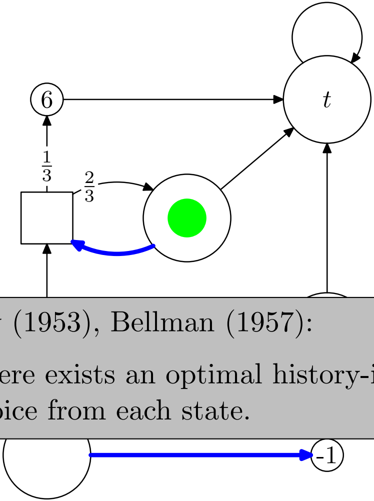

Reward: $-1 - 4$

# Markov decision processes (MDPs)




Reward: $-1 - 4$

# Markov decision processes (MDPs)

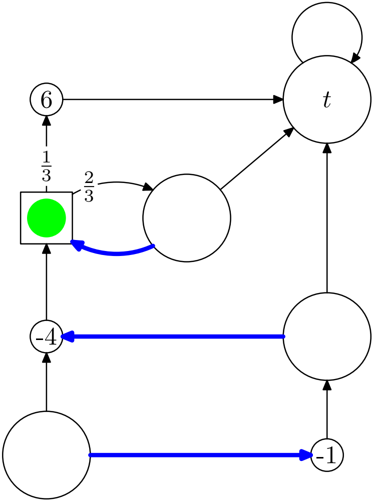



Shapley (1953), Bellman (1957):

> There exists an optimal history-independent choice from each state.

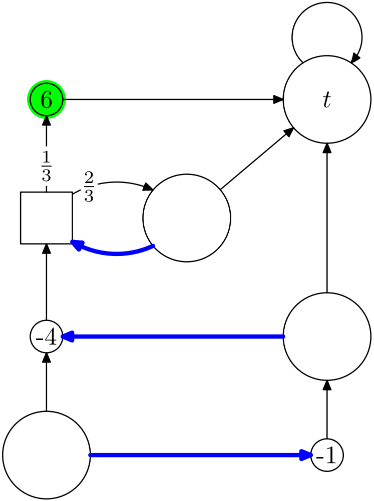
Reward: $-1 - 4$

# Markov decision processes (MDPs)



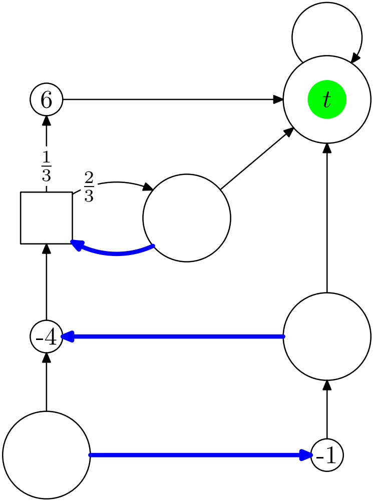

Reward: $-1 - 4$

# Markov decision processes (MDPs)




Reward: $-1 - 4 + 6$

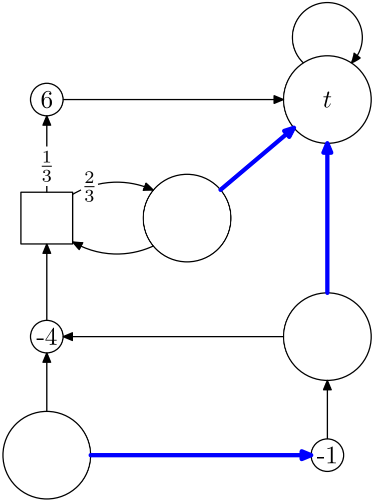
# Markov decision processes (MDPs)




Reward: $-1 - 4 + 6 \quad = \ 1$

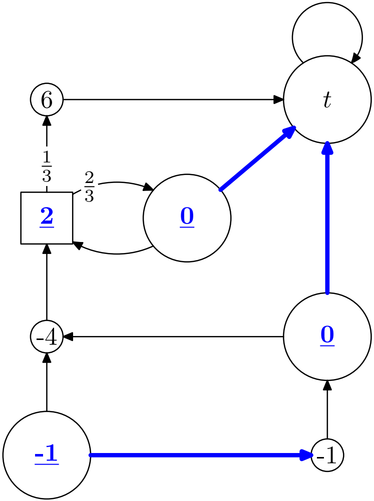
# Policies and corresponding values

- A policy $\pi$ is a choice of an action from each state.



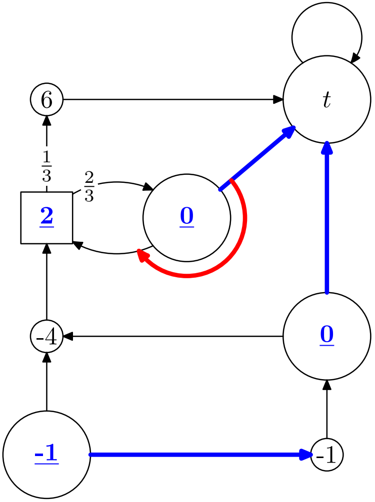
# Policies and corresponding values

- A policy $\pi$ is a choice of an action from each state.
- The value $\textsc{val}_\pi(i)$ of a state $i \in S$ for a policy $\pi$, is the expected sum of rewards obtained when moving according to $\pi$, starting from $i$.



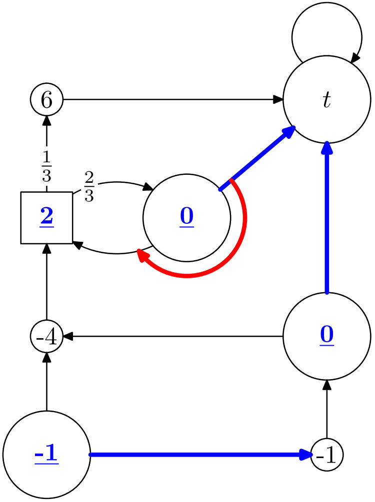
# Policies and corresponding values

- A policy $\pi$ is a choice of an action from each state.
- The value $\textsc{val}_\pi(i)$ of a state $i \in S$ for a policy $\pi$, is the expected sum of rewards obtained when moving according to $\pi$, starting from $i$.
- An action is an improving switch w.r.t. $\pi$ if it improves the values.



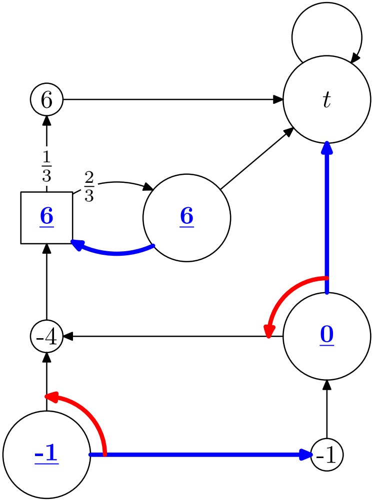
# Policies and corresponding values

- A policy $\pi$ is a choice of an action from each state.
- The value $\textsc{val}_\pi(i)$ of a state $i \in S$ for a policy $\pi$, is the expected sum of rewards obtained when moving according to $\pi$, starting from $i$.
- An action is an improving switch w.r.t. $\pi$ if it improves the values.
- It suffices to check whether an action is improving for one step w.r.t. the current values.



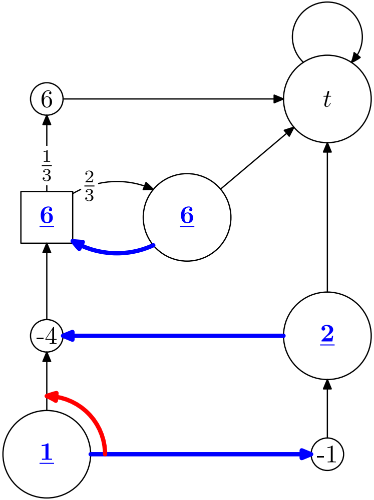
# Policies and corresponding values

- A policy $\pi$ is a choice of an action from each state.
- The value $\text{VAL}_\pi(i)$ of a state $i \in S$ for a policy $\pi$, is the expected sum of rewards obtained when moving according to $\pi$, starting from $i$.
- An action is an improving switch w.r.t. $\pi$ if it improves the values.
- It suffices to check whether an action is improving for one step w.r.t. the current values.



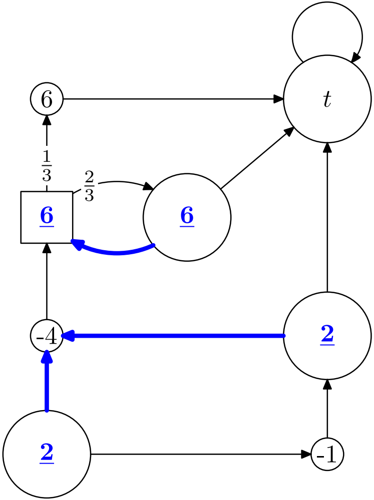
# Policies and corresponding values

- A policy $\pi$ is a choice of an action from each state.
- The value $\textsc{val}_\pi(i)$ of a state $i \in S$ for a policy $\pi$, is the expected sum of rewards obtained when moving according to $\pi$, starting from $i$.
- An action is an improving switch w.r.t. $\pi$ if it improves the values.
- It suffices to check whether an action is improving for one step w.r.t. the current values.



# Policies and corresponding values

- A policy $\pi$ is a choice of an action from each state.
- The value $\text{VAL}_\pi(i)$ of a state $i \in S$ for a policy $\pi$, is the expected sum of rewards obtained when moving according to $\pi$, starting from $i$.
- An action is an improving switch w.r.t. $\pi$ if it improves the values.
- It suffices to check whether an action is improving for one step w.r.t. the current values.
- A policy $\pi^*$ is optimal iff there are no improving switches. Optimal policies simultaneously maximize the values of all states.



# MDPs and linear programming

- No improving switches for optimal policy $\pi^*$:

$$\forall i \in S: \ \text{VAL}_{\pi^*}(i) = \max_{a \in A_i} \ r_a + \sum_{j \in S} p_{a,j} \text{VAL}_{\pi^*}(j)$$

  where $A_i$ is the set of actions from state $i$, $r_a$ is the expected reward of using action $a$, and $p_{a,j}$ is the probability of moving to state $j$ when using action $a$.

# MDPs and linear programming

- No improving switches for optimal policy $\pi^*$:

$$\forall i \in S: \ \mathrm{VAL}_{\pi^*}(i) = \max_{a \in A_i} \ r_a + \sum_{j \in S} p_{a,j} \mathrm{VAL}_{\pi^*}(j)$$

  where $A_i$ is the set of actions from state $i$, $r_a$ is the expected reward of using action $a$, and $p_{a,j}$ is the probability of moving to state $j$ when using action $a$.

- This can be used to formulate an LP for solving the MDP:

$$\begin{aligned} &\text{minimize} \quad \sum_{i \in S} v_i \\ &s.t. \ \ \forall i \in S \ \forall a \in A_i : \ v_i \geq r_a + \sum_{j \in S} p_{a,j} v_j \end{aligned}$$

# Primal and dual LPs for MDPs

$$\text{minimize} \quad \sum_{i \in S} v_i$$

$$s.t. \;\; \forall i \in S \; \forall a \in A_i : \;\; v_i \geq r_a + \sum_{j \in S} p_{a,j} v_j$$

$$\text{maximize} \quad \sum_{i \in S} \sum_{a \in A_i} r_a x_a$$

$$s.t. \;\; \forall i \in S : \quad \sum_{a \in A_i} x_a = 1 + \sum_{j \in S} \sum_{a \in A_j} p_{a,i} x_a$$

# Primal and dual LPs for MDPs

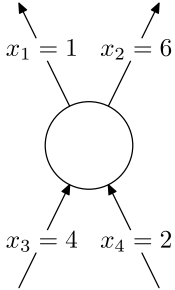
$$\text{minimize} \quad \sum_{i \in S} v_i$$

$$s.t. \;\; \forall i \in S \; \forall a \in A_i : \;\; v_i \geq r_a + \sum_{j \in S} p_{a,j} v_j$$

$$\text{maximize} \quad \sum_{i \in S} \sum_{a \in A_i} r_a x_a$$

$$s.t. \;\; \forall i \in S : \quad \sum_{a \in A_i} x_a = 1 + \sum_{j \in S} \sum_{a \in A_j} p_{a,i} x_a$$

Flow conservation:




$$x_1 + x_2 = 1 + x_3 + x_4$$

# Primal and dual LPs for MDPs

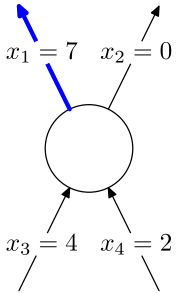
$$\text{minimize} \quad \sum_{i \in S} v_i$$

$$s.t. \;\; \forall i \in S \; \forall a \in A_i : \;\; v_i \geq r_a + \sum_{j \in S} p_{a,j} v_j$$

$$\text{maximize} \quad \sum_{i \in S} \sum_{a \in A_i} r_a x_a$$

$$s.t. \;\; \forall i \in S : \quad \sum_{a \in A_i} x_a = 1 + \sum_{j \in S} \sum_{a \in A_j} p_{a,i} x_a$$

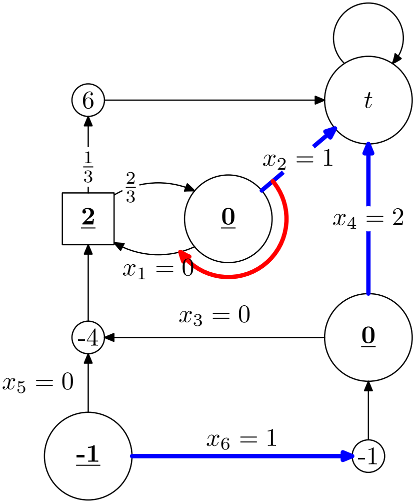
Flow conservation:




$$x_1 + x_2 = 1 + x_3 + x_4$$

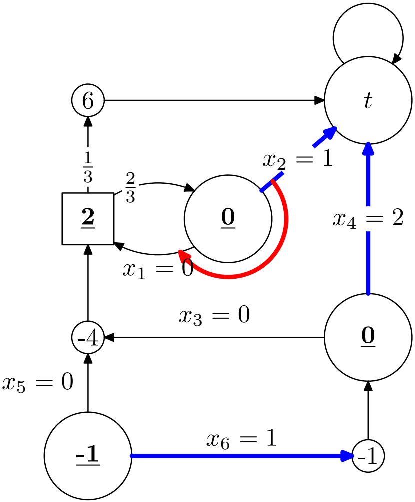
- Every basic feasible solution corresponds to a policy $\pi$.

# Variables of the primal LP




- $x_a$ is the expected number of times action $a$ is used, summed over all starting states.

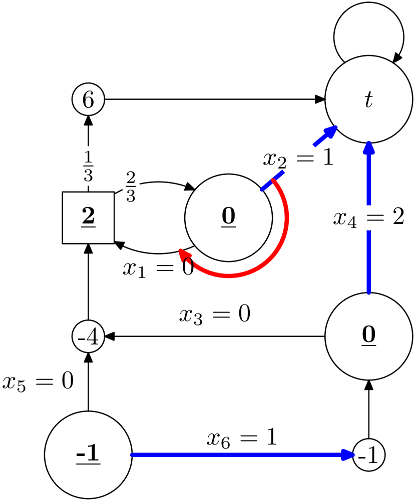
# Variables of the primal LP



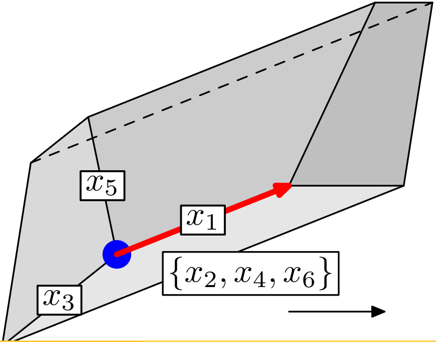

- $x_a$ is the expected number of times action $a$ is used, summed over all starting states.
- We have:

$$\sum_{i \in S} \mathrm{VAL}_{\pi}(i) = \sum_{a \in \pi} r_a x_a^{\pi}$$

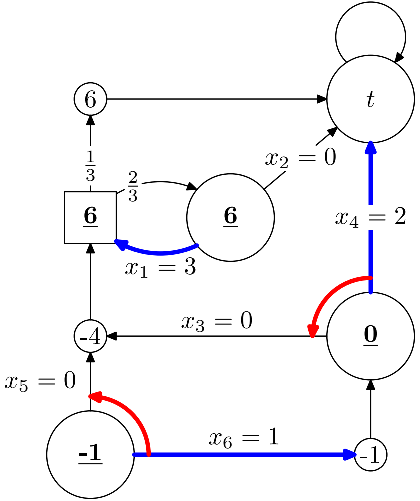
# From MDP to LP



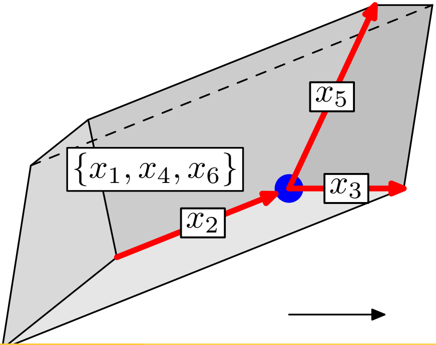

$$
\begin{array}{lrcl}
\max & \multicolumn{3}{l}{-1 + 2x_1 - 2x_3 - x_5} \\
\text{s.t.} & x_2 & = & 1 - \frac{1}{3}x_1 + \frac{2}{3}x_3 + \frac{2}{3}x_5 \\
& x_4 & = & 2 - x_3 - x_5 \\
& x_6 & = & 1 - x_5 \\
& \multicolumn{3}{l}{x_1, x_2, x_3, x_4, x_5, x_6 \geq 0}
\end{array}
$$



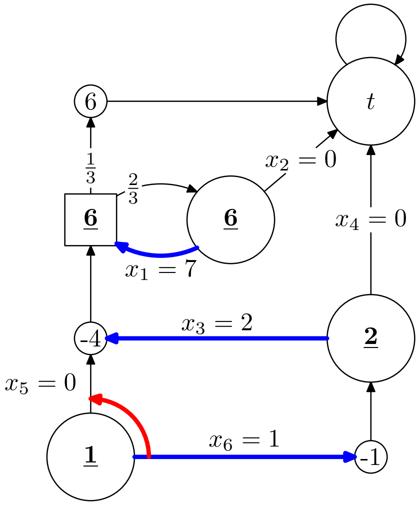
# From MDP to LP



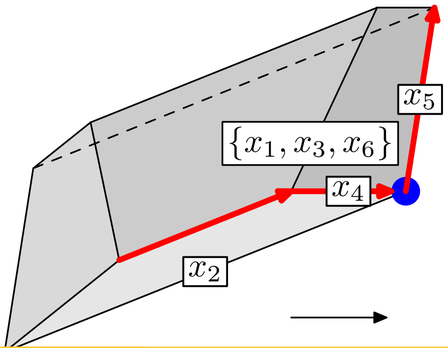

$$\begin{array}{lrcl} \max & \multicolumn{3}{l}{5 - 6x_2 + 2x_3 + 3x_5} \\ \text{s.t.} & x_1 & = & 3 - 3x_2 + 2x_3 + 2x_5 \\ & x_4 & = & 2 - x_3 - x_5 \\ & x_6 & = & 1 - x_5 \\ & \multicolumn{3}{l}{x_1, x_2, x_3, x_4, x_5, x_6 \geq 0} \end{array}$$



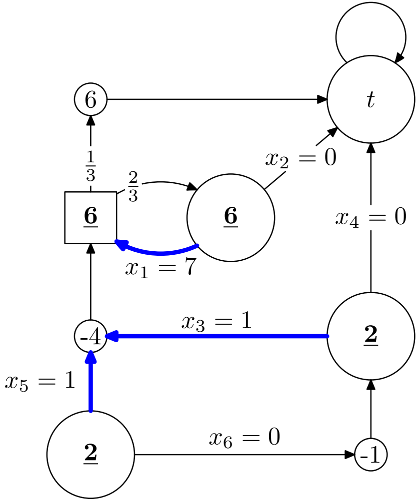
# From MDP to LP



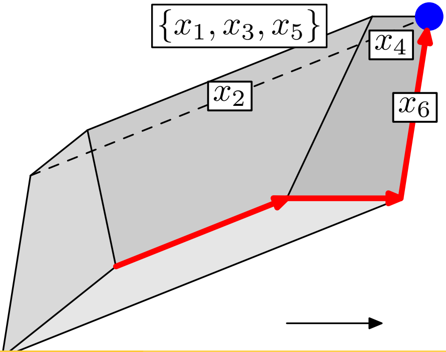

$$
\begin{aligned}
\max \quad & 9 - 6x_2 - 2x_4 + x_5 \\
\text{s.t.} \quad & x_1 = 7 - 3x_2 - 2x_4 \\
& x_3 = 2 - x_4 - x_5 \\
& x_6 = 1 - x_5 \\
& x_1, x_2, x_3, x_4, x_5, x_6 \geq 0
\end{aligned}
$$



# From MDP to LP




$$
\begin{array}{lrcl}
\max & \multicolumn{3}{l}{10 - 6x_2 - 2x_4 - x_6} \\
\text{s.t.} & x_1 & = & 7 - 3x_2 - 2x_4 \\
 & x_3 & = & 1 - x_4 + x_6 \\
 & x_5 & = & 1 - x_6 \\
 & \multicolumn{3}{l}{x_1, x_2, x_3, x_4, x_5, x_6 \geq 0}
\end{array}
$$



# Diameter

Question: theoretically possible to have polynomially many iterations?

Let $G$ be a Markov decision process and $n$ be the number of nodes.

Definition: the diameter of $G$ is the least number of iterations required to solve $G$

**Small Diameter Theorem**

The diameter of $G$ is less or equal to $n$.

# Lower Bound for Zadeh's Rule

# Lower bound construction

- We define a family of lower bound MDPs $G_n$ such that the Least-Entered pivoting rule will simulate an $n$-bit binary counter.

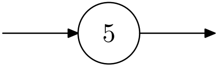
# Lower bound construction

- We define a family of lower bound MDPs $G_n$ such that the LEAST-ENTERED pivoting rule will simulate an $n$-bit binary counter.
- We make use of exponentially growing rewards (and penalties): To get a higher reward the MDP is willing to sacrifice everything that has been built up so far.

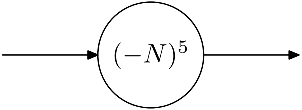
# Lower bound construction

- We define a family of lower bound MDPs $G_n$ such that the LEAST-ENTERED pivoting rule will simulate an $n$-bit binary counter.
- We make use of exponentially growing rewards (and penalties): To get a higher reward the MDP is willing to sacrifice everything that has been built up so far.
- Notation: Integer priority $p$ corresponds to reward $(-N)^p$, where $N = 7n + 1$.

$$\ldots < 5 < 3 < 1 < 2 < 4 < 6 < \ldots$$




for



# Background

- The use of priorities is inspired by *parity games*.

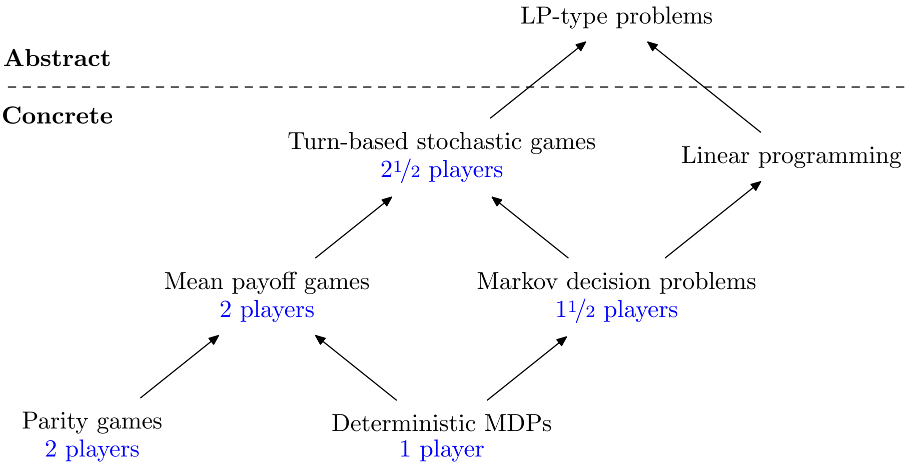
# Background

- The use of priorities is inspired by *parity games*.
- Friedmann (2009): The strategy iteration algorithm may require exponentially many iterations to solve parity games.
- Fearnley (2010): The strategy iteration algorithm may require exponentially many iterations to solve MDPs.

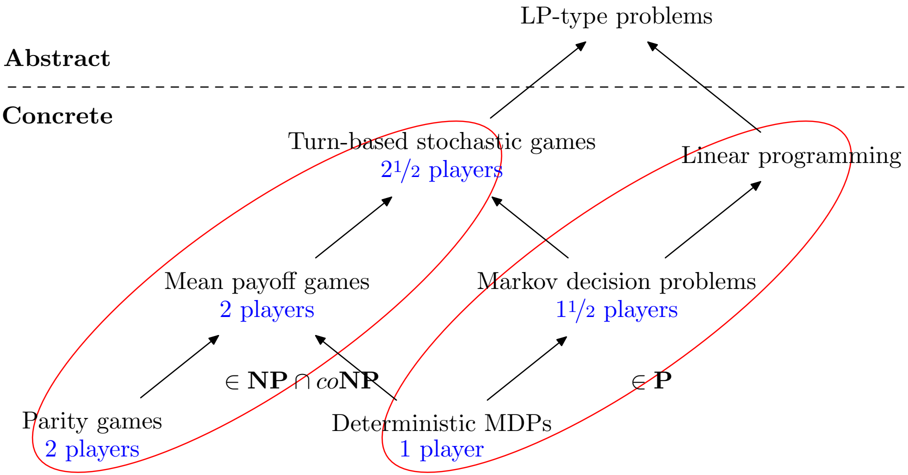
# Background

- The use of priorities is inspired by *parity games*.
- Friedmann (2009): The strategy iteration algorithm may require exponentially many iterations to solve parity games.
- Fearnley (2010): The strategy iteration algorithm may require exponentially many iterations to solve MDPs.
- We also first proved a lower bound for parity games and then transferred the result to MDPs and linear programs.

# Related game-theoretic settings



# Related game-theoretic settings



# Zadeh's pivoting rule

**Zadeh's Least-Entered rule**

Perform single switch that has been applied least often.

Dear Victor,

Please post this offer of \$1000 to the first person who can find a counterexample to the least entered rule or prove it to be polynomial. The least entered rule enters the improving variable which has been entered least often.

Sincerely,

Norman Zadeh

(taken from David Avis' paper)

# Tie-Breaking Rule

Tie-Breaking Rule = method of selecting a switch in case of a tie (w.r.t. the occurrence record)

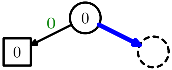
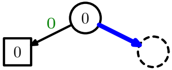
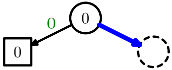
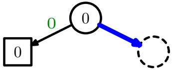
Proof of Small Diameter Theorem implies:

**Corollary**

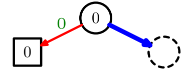
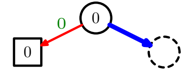
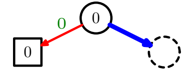
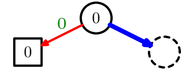
There is a tie-breaking rule s.t. Zadeh's rule requires linearly many iterations in the worst-case.

Consequence: lower bound construction is equipped with particular tie-breaking rule

# Binary Counting

| 0 | 0 | 0 | 0 |
|---|---|---|---|
|  |  |  |  |

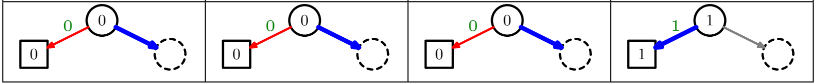
# Binary Counting

| 0 | 0 | 0 | 0 |
|---|---|---|---|
|  |  |  |  |

Principle: If a bit can be set, then all bits can be set.

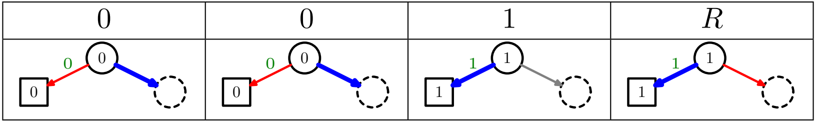
# Binary Counting

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 1 |



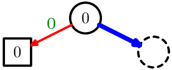
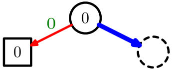
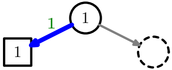
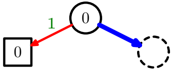

Tie-Breaking: We decide to set the first bit.

# Binary Counting



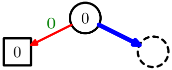
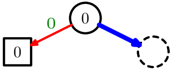
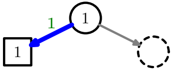
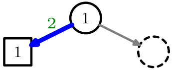

Set the second bit and reset the first bit.

# Binary Counting

| 0 | 0 | 1 | 0 |
|---|---|---|---|
|   |   |   |   |

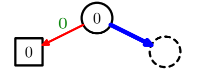
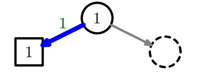
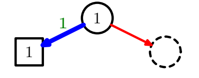
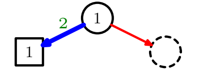
Set the first bit again.

# Binary Counting

| 0 | 0 | 1 | 1 |
|---|---|---|---|
|   |   |   |   |

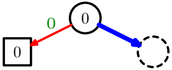
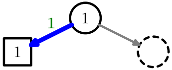
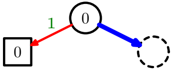
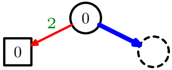
Continue...

# Binary Counting

| 0 | 1 | $R$ | $R$ |
|---|---|---|---|
|  |  |  |  |

Continue...

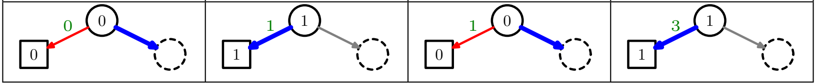
# Binary Counting

| 0 | 1 | 0 | 0 |
|---|---|---|---|
|  |  |  |  |

Continue...

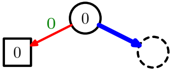
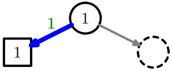
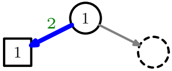
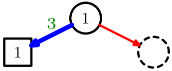
# Binary Counting

| 0 | 1 | 0 | 1 |
|---|---|---|---|




Continue...

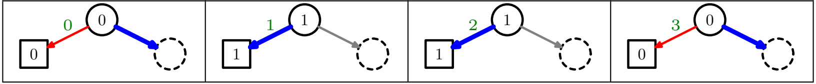
# Binary Counting

| 0 | 1 | 1 | $R$ |
|---|---|---|---|
|  |  |  |  |

Continue...

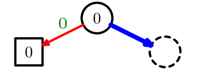
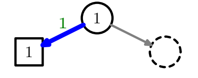
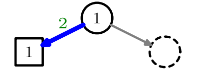
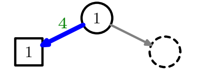
# Binary Counting

| 0 | 1 | 1 | 0 |
|---|---|---|---|



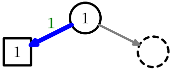
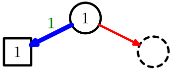
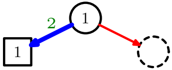
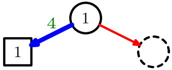

Continue...

# Binary Counting

| 0 | 1 | 1 | 1 |
|---|---|---|---|
|  |  |  |  |

Continue...

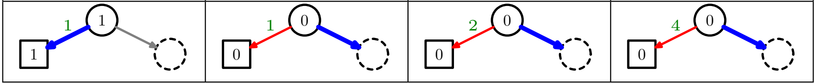
# Binary Counting

<table>
<tr><td>1</td><td>$R$</td><td>$R$</td><td>$R$</td></tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</table>

Continue...

# Binary Counting

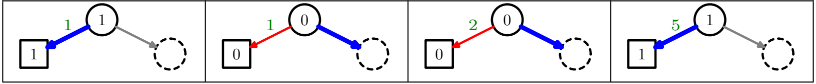
| 1 | 0 | 0 | 0 |
|---|---|---|---|




Continue...

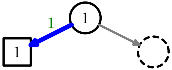
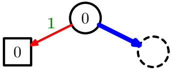
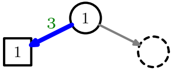
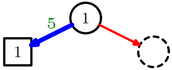
# Binary Counting

| 1 | 0 | 0 | 1 |
|---|---|---|---|



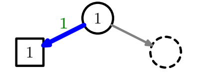
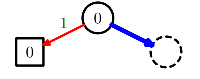
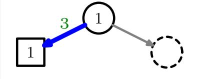
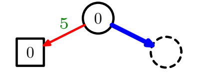

Continue...

# Binary Counting

<table>
<tr><th>1</th><th>0</th><th>1</th><th>$R$</th></tr>
<tr>
<td>
</td>
<td>
</td>
<td>
</td>
<td>
</td>
</tr>
</table>

Continue...

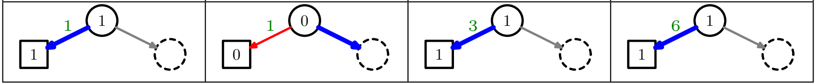
# Binary Counting

| 1 | 0 | 1 | 0 |
|---|---|---|---|
|  |  |  |  |

Continue...

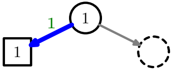
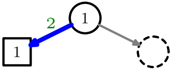
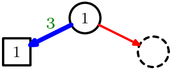
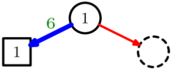
# Binary Counting

| 1 | 0 | 1 | 1 |
|---|---|---|---|



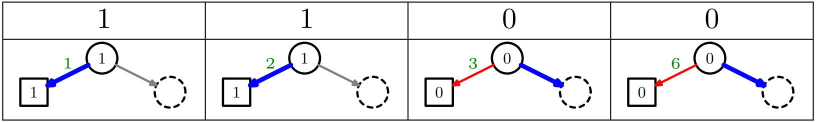

Continue...

# Binary Counting

| $1$ | $1$ | $R$ | $R$ |
|---|---|---|---|
|   |   |   |   |

Continue...

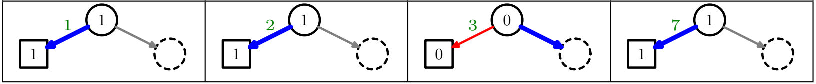
# Binary Counting




Continue...

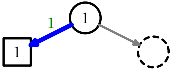
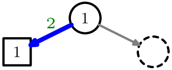
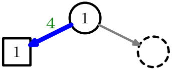
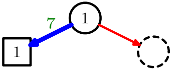
# Binary Counting

| 1 | 1 | 0 | 1 |
|---|---|---|---|



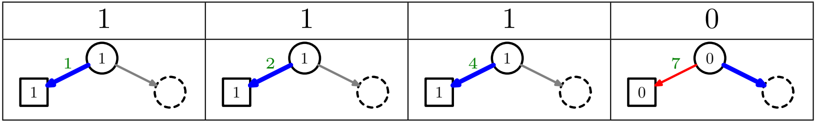

Continue...

# Binary Counting

| 1 | 1 | 1 | $R$ |
|---|---|---|---|
|  |  |  |  |

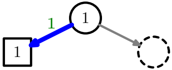
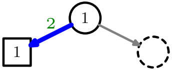
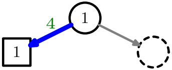
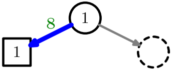
Continue...

# Binary Counting



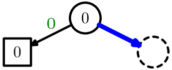
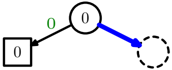
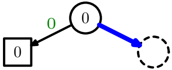
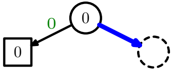

Continue...

# Binary Counting

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
|   |   |   |   |

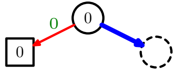
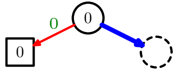
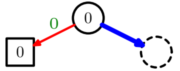
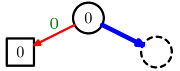
Problem: Occurrence record unbalanced!

# Binary Counting (... again!)

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
|  |  |  |  |

Let's do it again - watch the occurrence record this time!

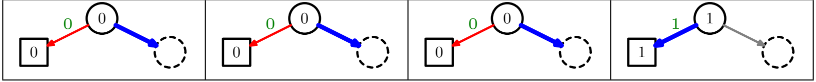
# Binary Counting (... again!)

| 0 | 0 | 0 | 0 |
|---|---|---|---|
|  |  |  |  |

Everything okay so far...

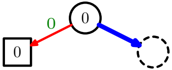
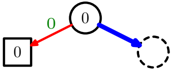
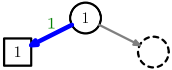
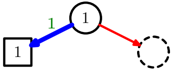
# Binary Counting (... again!)

| 0 | 0 | 0 | 1 |
|---|---|---|---|



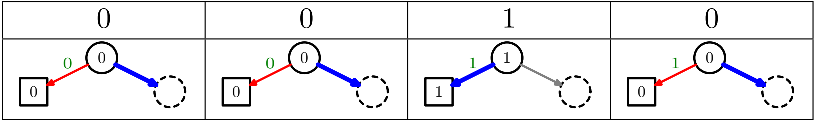

Everything okay so far...

# Binary Counting (... again!)

| 0 | 0 | 1 | $R$ |
|---|---|---|---|
|  |  |  |  |

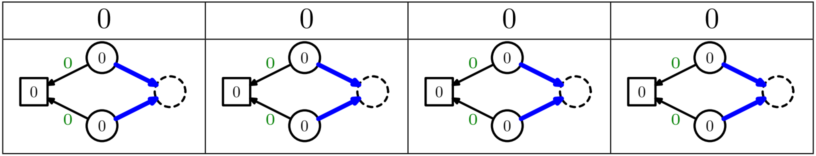
Everything okay so far...

# Binary Counting (... again!)




Problem: We have to set one of the higher bits now!

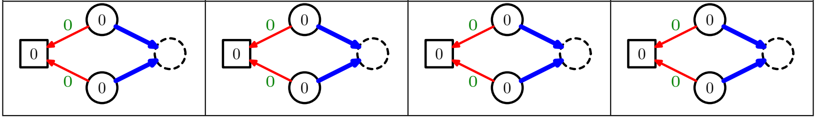
# Binary Counting with conjunctive bits




Replace gadget by two-bit, conjunctive structure.

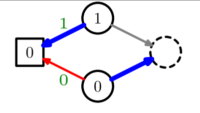
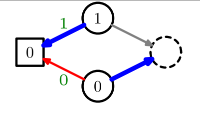
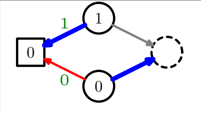
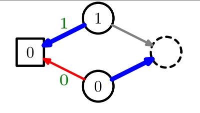
# Binary Counting with conjunctive bits

| 0 | 0 | 0 | 0 |
|---|---|---|---|




Gadget is set iff both edges are going in.

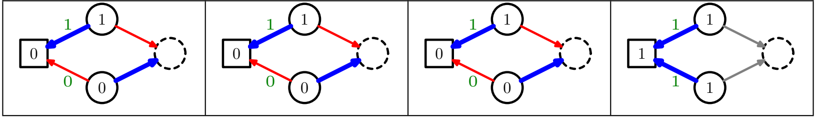
# Binary Counting with conjunctive bits

| 0 | 0 | 0 | 0 |
|---|---|---|---|
|  |  |  |  |

Set one improving edge of every gadget.

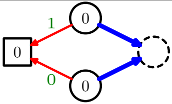
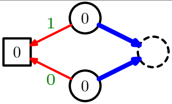
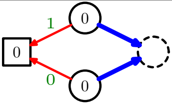
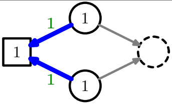
# Binary Counting with conjunctive bits

| 0 | 0 | 0 | 1 |
|---|---|---|---|




Set other improving edge of first gadget.

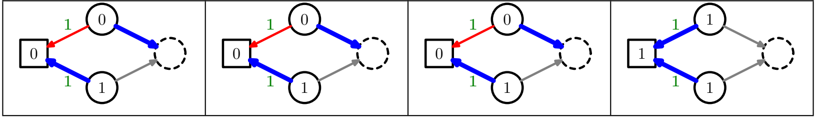
# Binary Counting with conjunctive bits

| 0 | 0 | 0 | 1 |
|---|---|---|---|
|  |  |  |  |

Other gadgets have updated to their old setting.

# Binary Counting with conjunctive bits

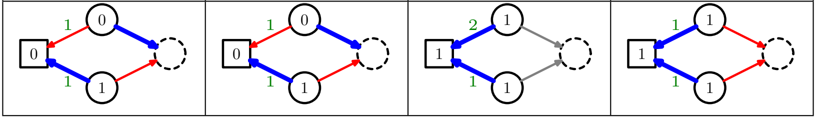
| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 1 |




Set one improving edge of every gadget again.

# Binary Counting with conjunctive bits

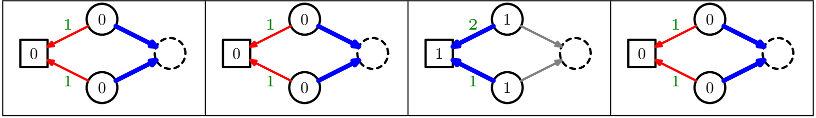
| 0 | 0 | 1 | $R$ |
|---|---|---|---|




Set other improving edge of second gadget.

# Binary Counting with conjunctive bits

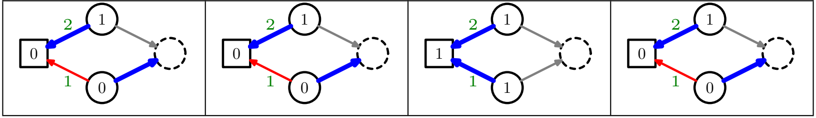
| 0 | 0 | 1 | 0 |
|---|---|---|---|




Reset all other gadgets.

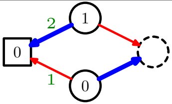
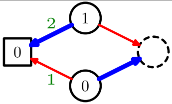
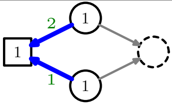
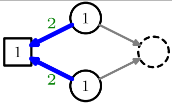
# Binary Counting with conjunctive bits

| 0 | 0 | 1 | 0 |
|---|---|---|---|



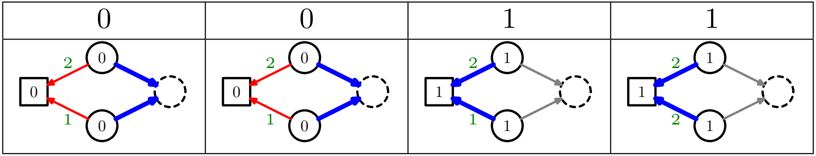

Everything okay so far, continue...

# Binary Counting with conjunctive bits

| 0 | 0 | 1 | 1 |
|---|---|---|---|
|  |  |  |  |

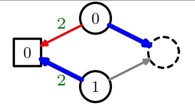
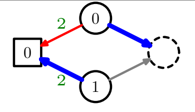
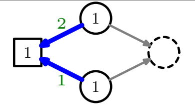
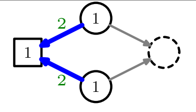
Everything okay so far, continue...

# Binary Counting with conjunctive bits



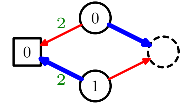
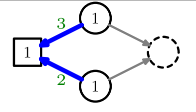
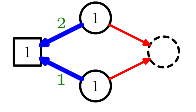
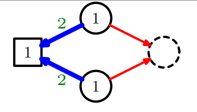

Everything okay so far, continue...

# Binary Counting with conjunctive bits

| 0 | 0 | 1 | 1 |
|---|---|---|---|
|  |  |  |  |

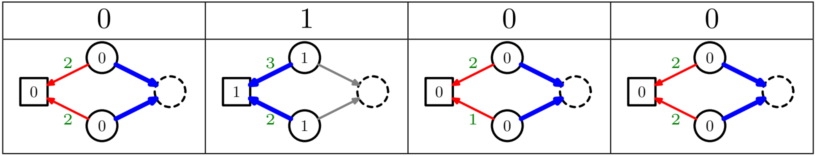
Everything okay so far, continue...

# Binary Counting with conjunctive bits

| 0 | 1 | $R$ | $R$ |
|---|---|---|---|
|  |  |  |  |

Everything okay so far, continue...

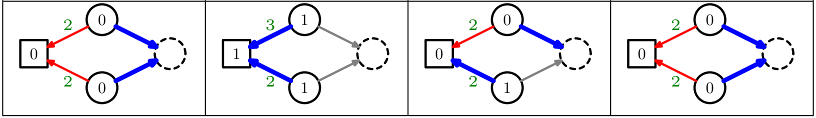
# Binary Counting with conjunctive bits




Everything okay so far, continue...

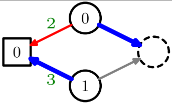
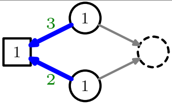
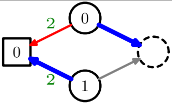
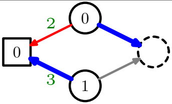
# Binary Counting with conjunctive bits

| 0 | 1 | 0 | 0 |
|---|---|---|---|



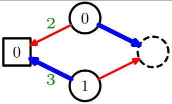
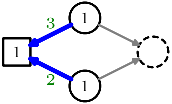
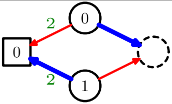
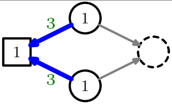

Everything okay so far, continue...

# Binary Counting with conjunctive bits

| 0 | 1 | 0 | 0 |
|---|---|---|---|
|  |  |  |  |

Everything okay so far, continue...

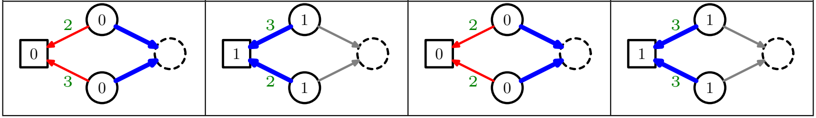
# Binary Counting with conjunctive bits

<table>
<tr><th>0</th><th>1</th><th>0</th><th>1</th></tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</table>

Everything okay so far, continue...

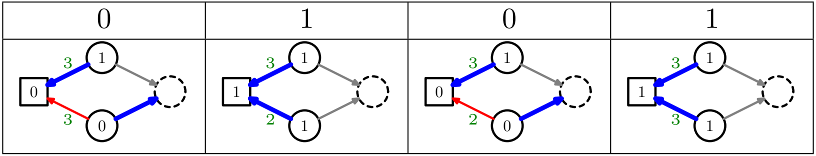
# Binary Counting with conjunctive bits

| 0 | 1 | 0 | 1 |
|---|---|---|---|



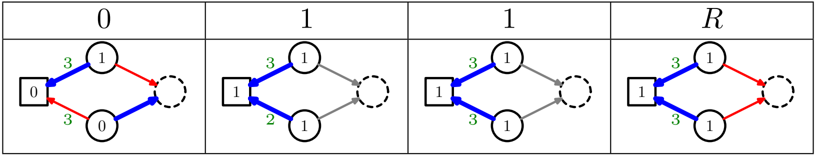

Everything okay so far, continue...

# Binary Counting with conjunctive bits




Everything okay so far, continue...

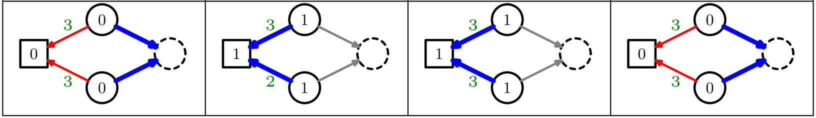
# Binary Counting with conjunctive bits




Everything okay so far, continue...

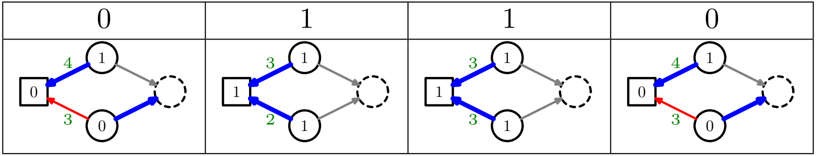
# Binary Counting with conjunctive bits

| 0 | 1 | 1 | 0 |
|---|---|---|---|




Everything okay so far, continue...

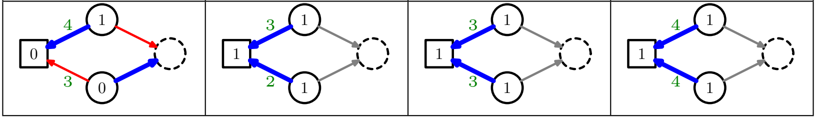
# Binary Counting with conjunctive bits




Everything okay so far, continue...

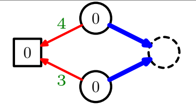
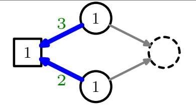
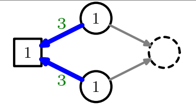
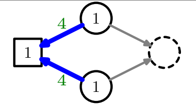
# Binary Counting with conjunctive bits

| 0 | 1 | 1 | 1 |
|---|---|---|---|




Everything okay so far, continue...

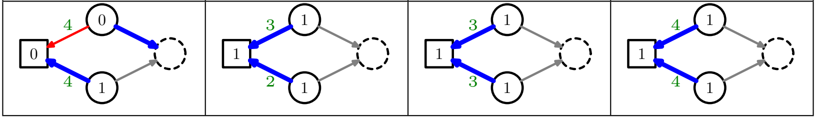
# Binary Counting with conjunctive bits

| 0 | 1 | 1 | 1 |
|---|---|---|---|
|   |   |   |   |

Everything okay so far, continue...

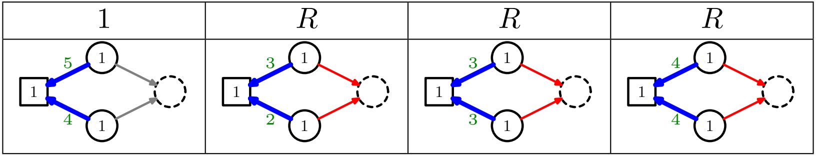
# Binary Counting with conjunctive bits

| | | | |
|---|---|---|---|
| 0 | 1 | 1 | 1 |



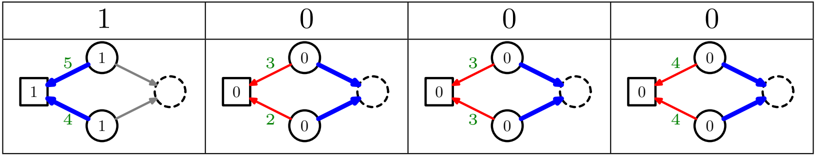

Everything okay so far, continue...

# Binary Counting with conjunctive bits



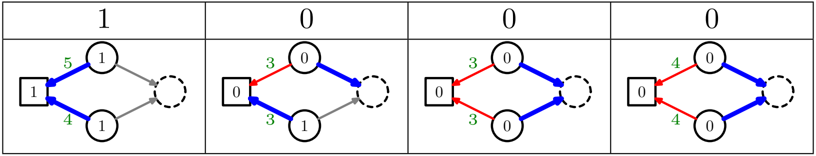

Reset all three lower gadgets.

# Binary Counting with conjunctive bits



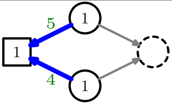
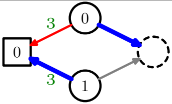
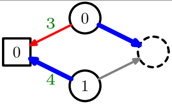
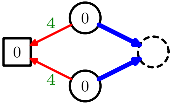

Occurrence record of gadget #3 is pretty low...

# Binary Counting with conjunctive bits



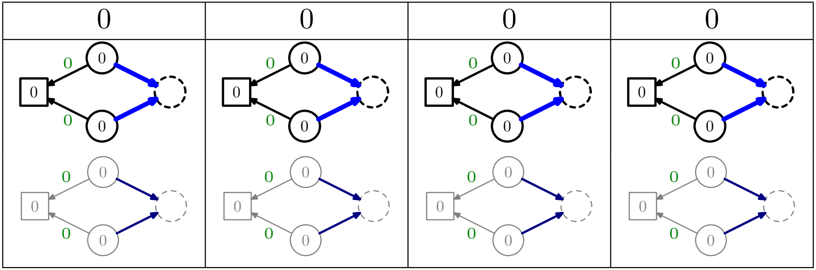

Occurrence record of gadget #3 is pretty low...

# Binary Counting with conjunctive bits

| 1 | 0 | 0 | 0 |
|---|---|---|---|
|  |  |  |  |

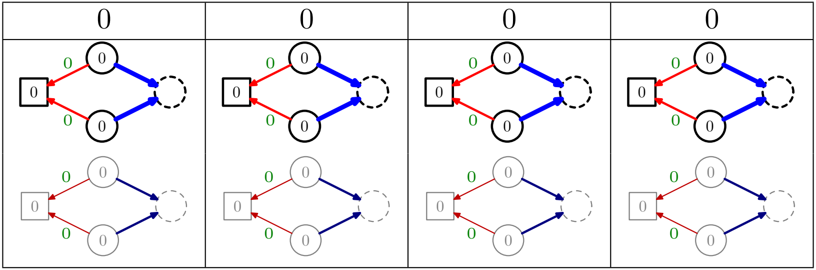
Problem: We have to set gadget #2 or #3 now!

# Binary Counting with conjunctive representatives



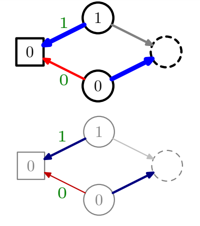
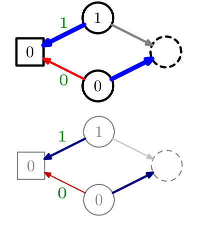
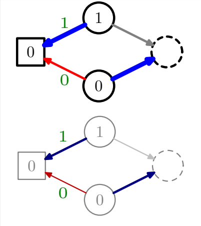
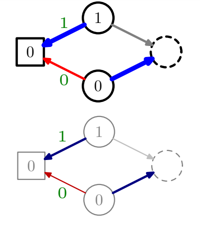

Replace gadget by two conjunctive structures.

# Binary Counting with conjunctive representatives




Only one representative subgadget is active. The upper representative is active iff the next higher bit is not set

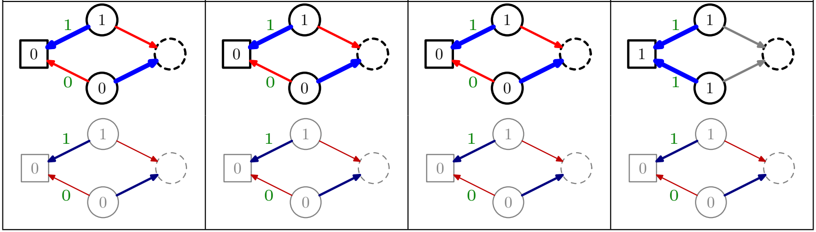
# Binary Counting with conjunctive representatives

| 0 | 0 | 0 | 0 |
|---|---|---|---|
|  |  |  |  |

Only one representative subgadget is active. The upper representative is active iff the next higher bit is not set

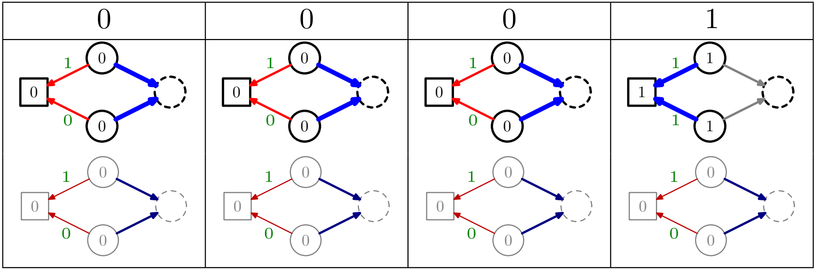
# Binary Counting with conjunctive representatives

| 0 | 0 | 0 | 1 |
|---|---|---|---|



Only one representative subgadget is active. The upper representative is active iff the next higher bit is not set

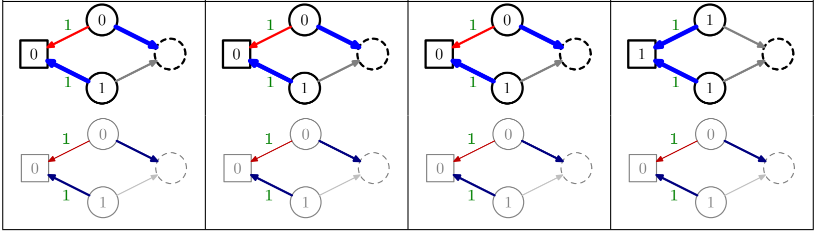
# Binary Counting with conjunctive representatives




Only one representative subgadget is active. The upper representative is active iff the next higher bit is not set

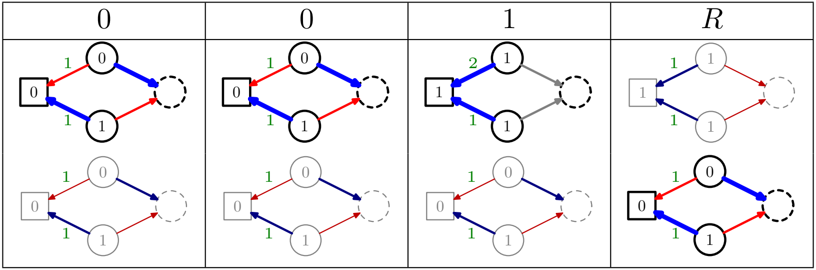
# Binary Counting with conjunctive representatives

| 0 | 0 | 0 | 1 |
|---|---|---|---|



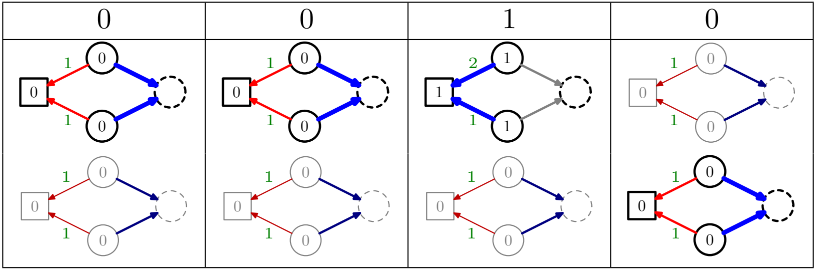
Only one representative subgadget is active. The upper representative is active iff the next higher bit is not set

# Binary Counting with conjunctive representatives




Only one representative subgadget is active. The upper representative is active iff the next higher bit is not set

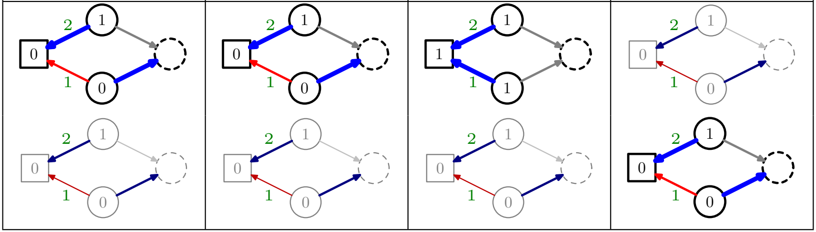
# Binary Counting with conjunctive representatives




Only one representative subgadget is active. The upper representative is active iff the next higher bit is not set

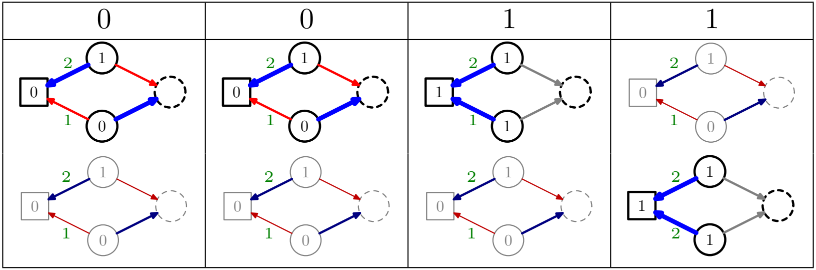
# Binary Counting with conjunctive representatives

| 0 | 0 | 1 | 0 |
|---|---|---|---|



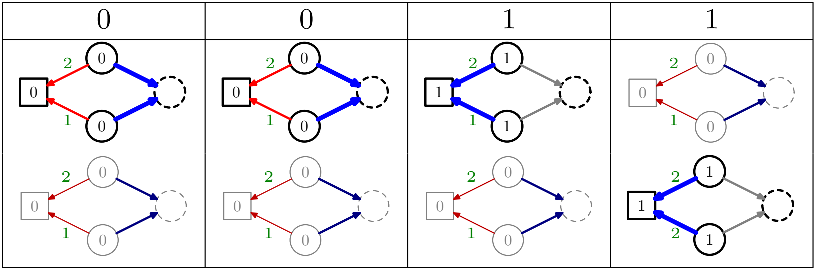
Only one representative subgadget is active. The upper representative is active iff the next higher bit is not set

# Binary Counting with conjunctive representatives




Only one representative subgadget is active. The upper representative is active iff the next higher bit is not set

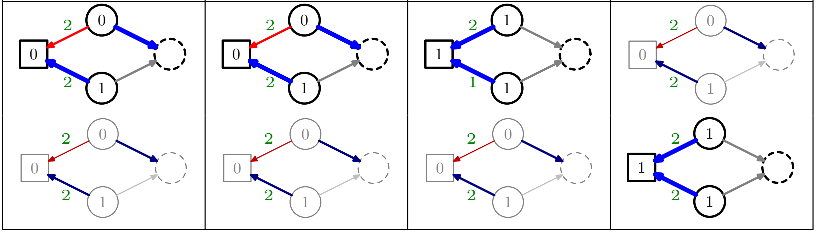
# Binary Counting with conjunctive representatives




Only one representative subgadget is active. The upper representative is active iff the next higher bit is not set

# Binary Counting with conjunctive representatives

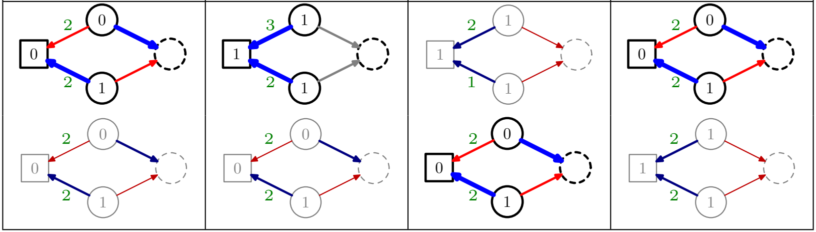
| 0 | 0 | 1 | 1 |
|---|---|---|---|



Only one representative subgadget is active. The upper representative is active iff the next higher bit is not set

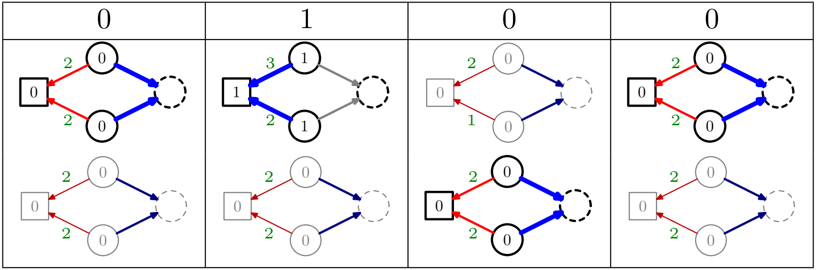
# Binary Counting with conjunctive representatives

| 0 | 1 | $R$ | $R$ |
|---|---|---|---|



Only one representative subgadget is active. The upper representative is active iff the next higher bit is not set

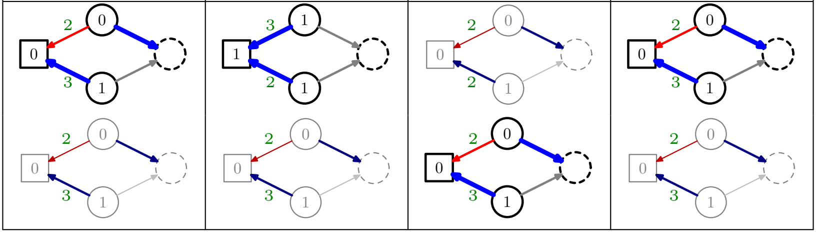
# Binary Counting with conjunctive representatives




Only one representative subgadget is active. The upper representative is active iff the next higher bit is not set

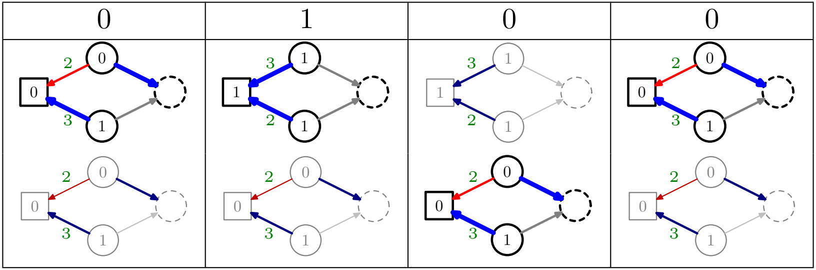
# Binary Counting with conjunctive representatives

| 0 | 1 | 0 | 0 |
|---|---|---|---|



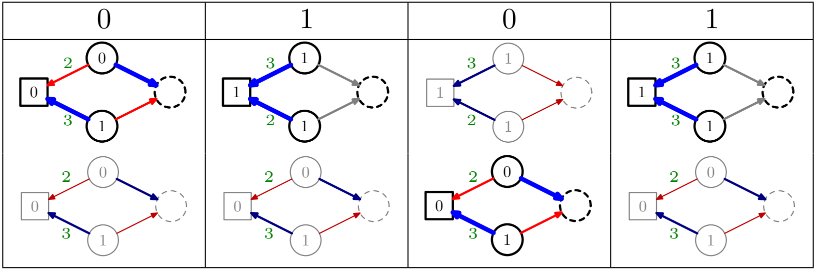
Only one representative subgadget is active. The upper representative is active iff the next higher bit is not set

# Binary Counting with conjunctive representatives




Only one representative subgadget is active. The upper representative is active iff the next higher bit is not set

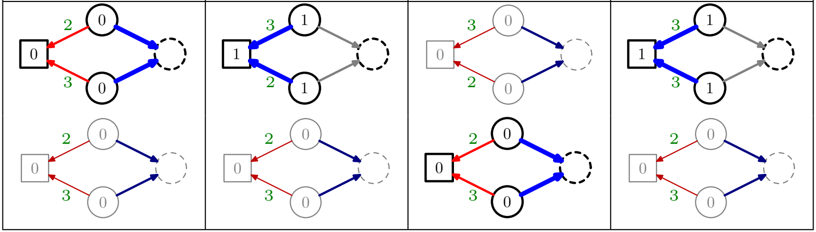
# Binary Counting with conjunctive representatives




Only one representative subgadget is active. The upper representative is active iff the next higher bit is not set

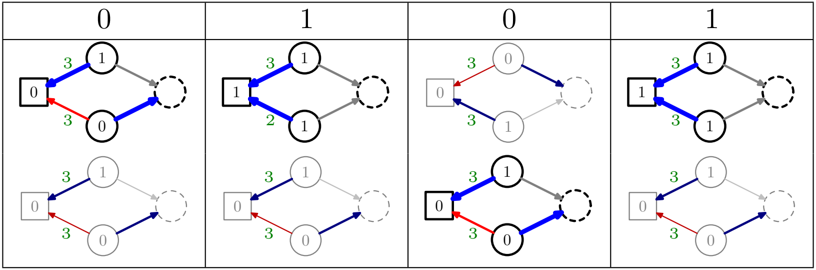
# Binary Counting with conjunctive representatives

| 0 | 1 | 0 | 1 |
|---|---|---|---|



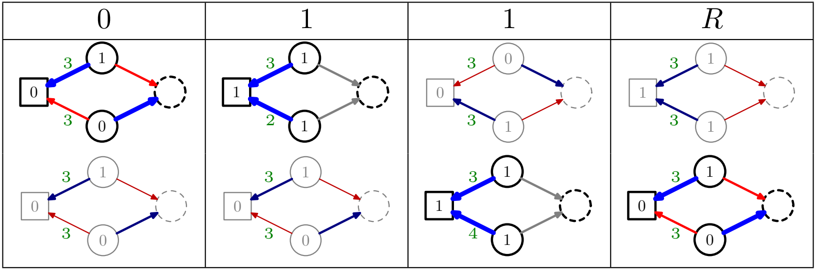

Only one representative subgadget is active. The upper representative is active iff the next higher bit is not set

# Binary Counting with conjunctive representatives




Only one representative subgadget is active. The upper representative is active iff the next higher bit is not set

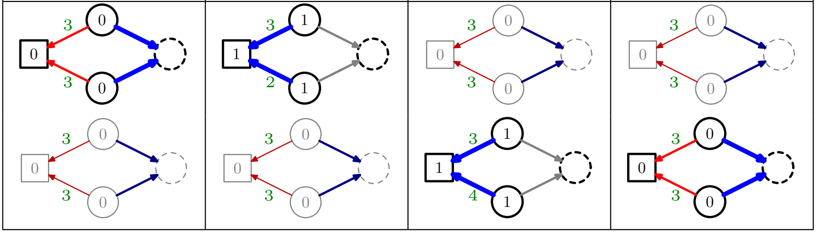
# Binary Counting with conjunctive representatives




Only one representative subgadget is active. The upper representative is active iff the next higher bit is not set

# Binary Counting with conjunctive representatives

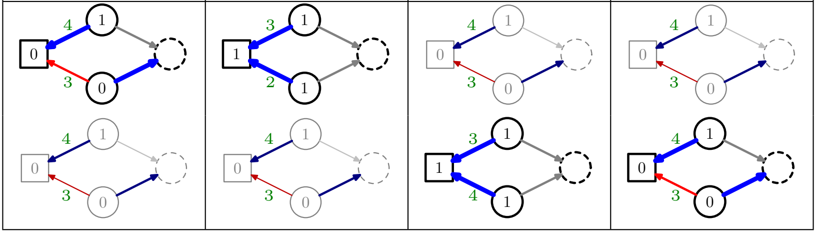
| 0 | 1 | 1 | 0 |
|---|---|---|---|




Only one representative subgadget is active. The upper representative is active iff the next higher bit is not set

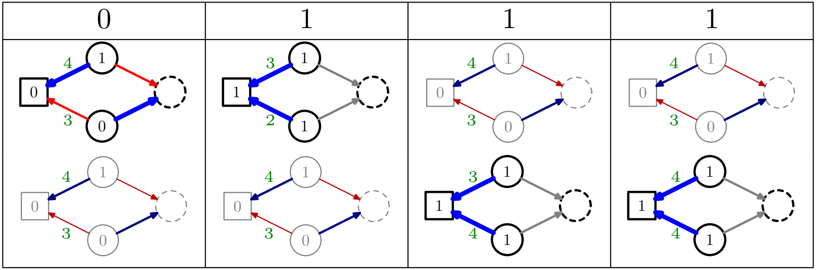
# Binary Counting with conjunctive representatives

| 0 | 1 | 1 | 0 |
|---|---|---|---|



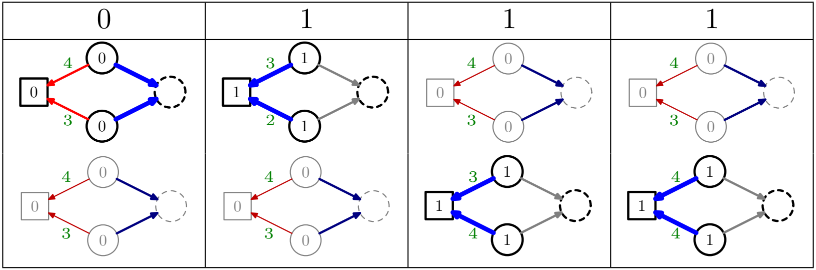
Only one representative subgadget is active. The upper representative is active iff the next higher bit is not set

# Binary Counting with conjunctive representatives



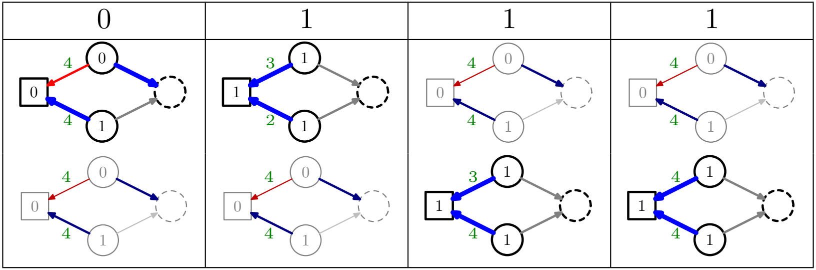

Only one representative subgadget is active. The upper representative is active iff the next higher bit is not set

# Binary Counting with conjunctive representatives



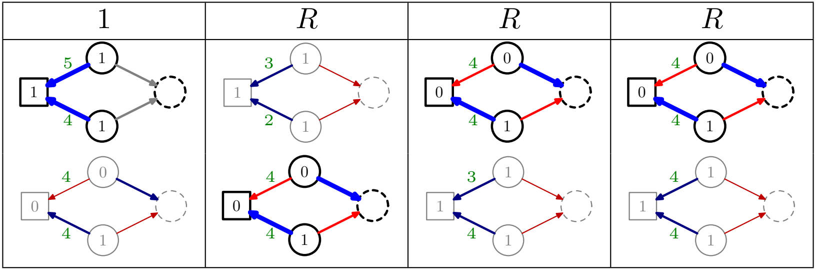

Only one representative subgadget is active. The upper representative is active iff the next higher bit is not set

# Binary Counting with conjunctive representatives



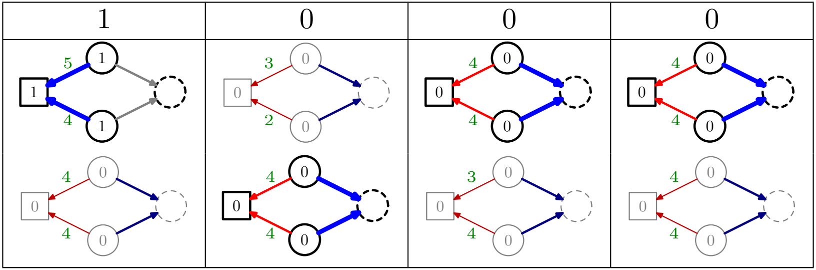

Only one representative subgadget is active. The upper representative is active iff the next higher bit is not set

# Binary Counting with conjunctive representatives



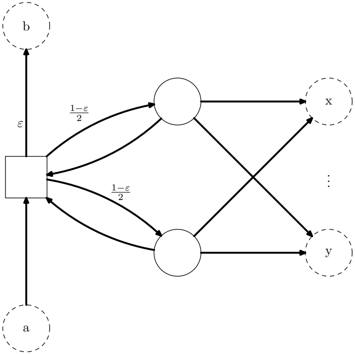

Only one representative subgadget is active. The upper representative is active iff the next higher bit is not set

# Binary Counting with conjunctive representatives

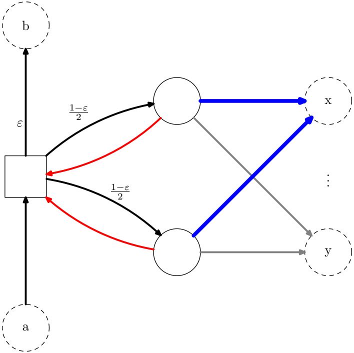



Only one representative subgadget is active. The upper representative is active iff the next higher bit is not set

# Bit Gadget



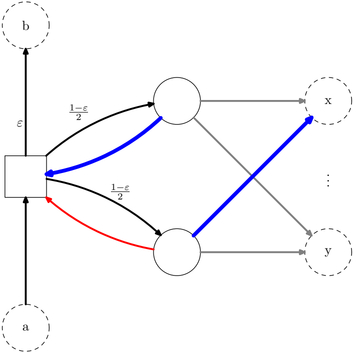
# Bit Gadget




- Bit unset
- Improving to go in

$$\mathrm{VAL}_\sigma(a) = \underbrace{\varepsilon \cdot \mathrm{VAL}_\sigma(b)}_{\approx 0} + \underbrace{(1-\varepsilon)}_{\approx 1} \cdot \mathrm{VAL}_\sigma(x) \approx \mathrm{VAL}_\sigma(x)$$

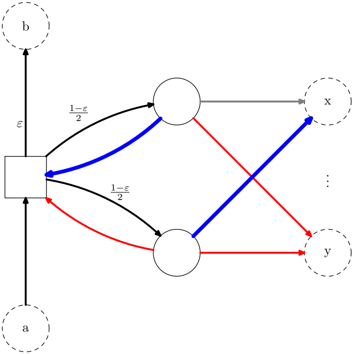
# Bit Gadget




- Bit still unset (only one edge going in)
- Still improving to go in with the other edge

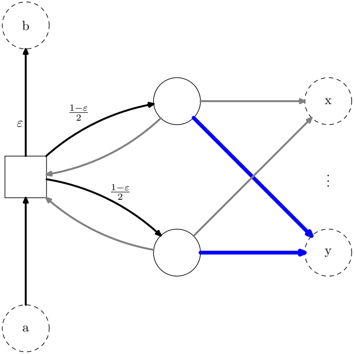
$$\mathrm{VAL}_\sigma(a) = \underbrace{\frac{2\varepsilon}{1+\varepsilon} \cdot \mathrm{VAL}_\sigma(b)}_{\approx 0} + \underbrace{\frac{1-\varepsilon}{1+\varepsilon}}_{\approx 1} \cdot \mathrm{VAL}_\sigma(x) \approx \mathrm{VAL}_\sigma(x)$$

# Bit Gadget



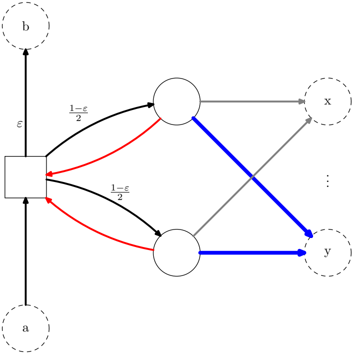

- $y$ has now better valuation than $x$
- Gadget could close, but also open completely again

# Bit Gadget



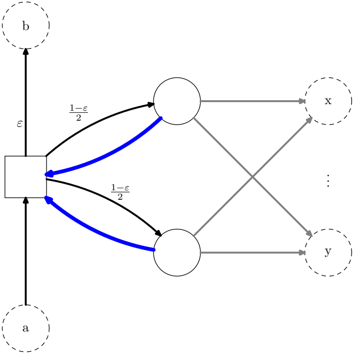

- Bit unset
- No improvements

# Bit Gadget




- Bit unset
- Improving to go in

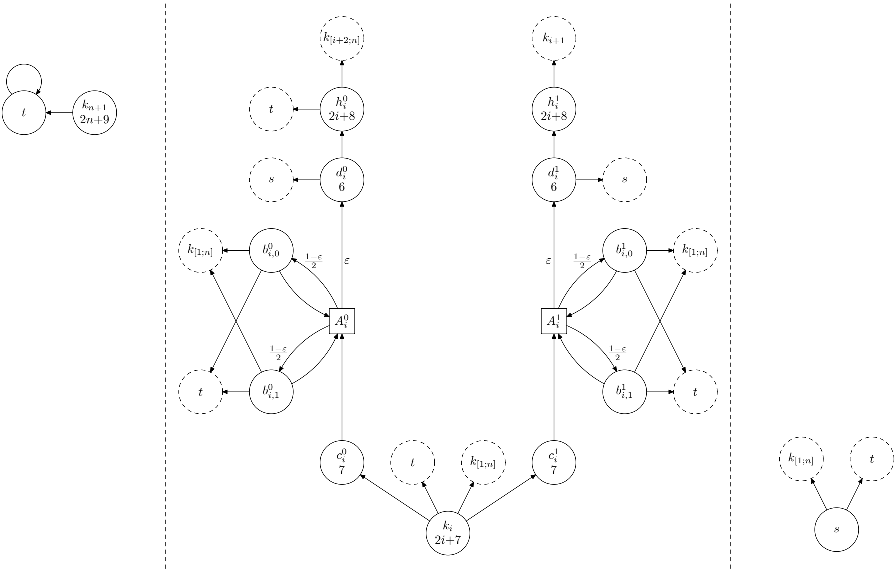
# Bit Gadget




- Bit set
- Now $b$ can be "observed" from $a$

$$\text{VAL}_\sigma(a) = \text{VAL}_\sigma(b)$$

# Full Construction



# Concluding Remarks

# Open problems

- Obtain lower bounds for related history-based pivoting rules
  - Least-recently considered: subexponential lower bound
  - Least-recently basic, Least-recently entered, Least basic iterations: work in progress
- Polytime algorithm for two-player games and the like
- Strongly polytime algorithm for LPs (and MDPs)
- Resolving the Hirsch conjecture
- Find game-theoretic model with unresolved diameter bounds

# The slide usually called “the end”.

Thank you for listening!