Pancake Flipping Is Hard

Laurent BULTEAU, Guillaume FERTIN, Irena RUSU
LINA, Université de Nantes

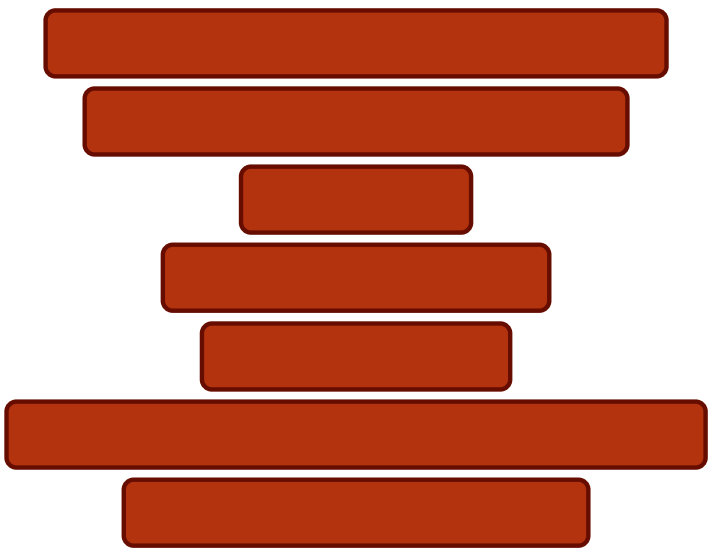
Algorithms & Permutations 2012

Feb. 21st, 2012

Pancake Flipping Problem

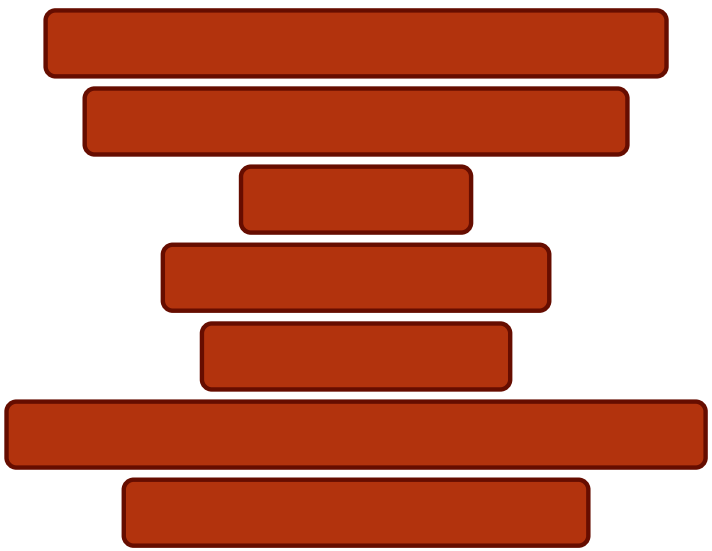
Pancakes

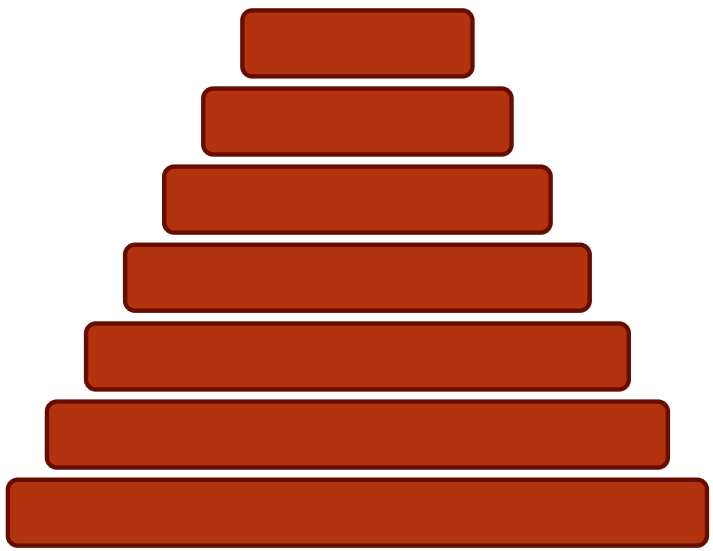
- we are given a stack of pancakes, all of different sizes



Pancakes

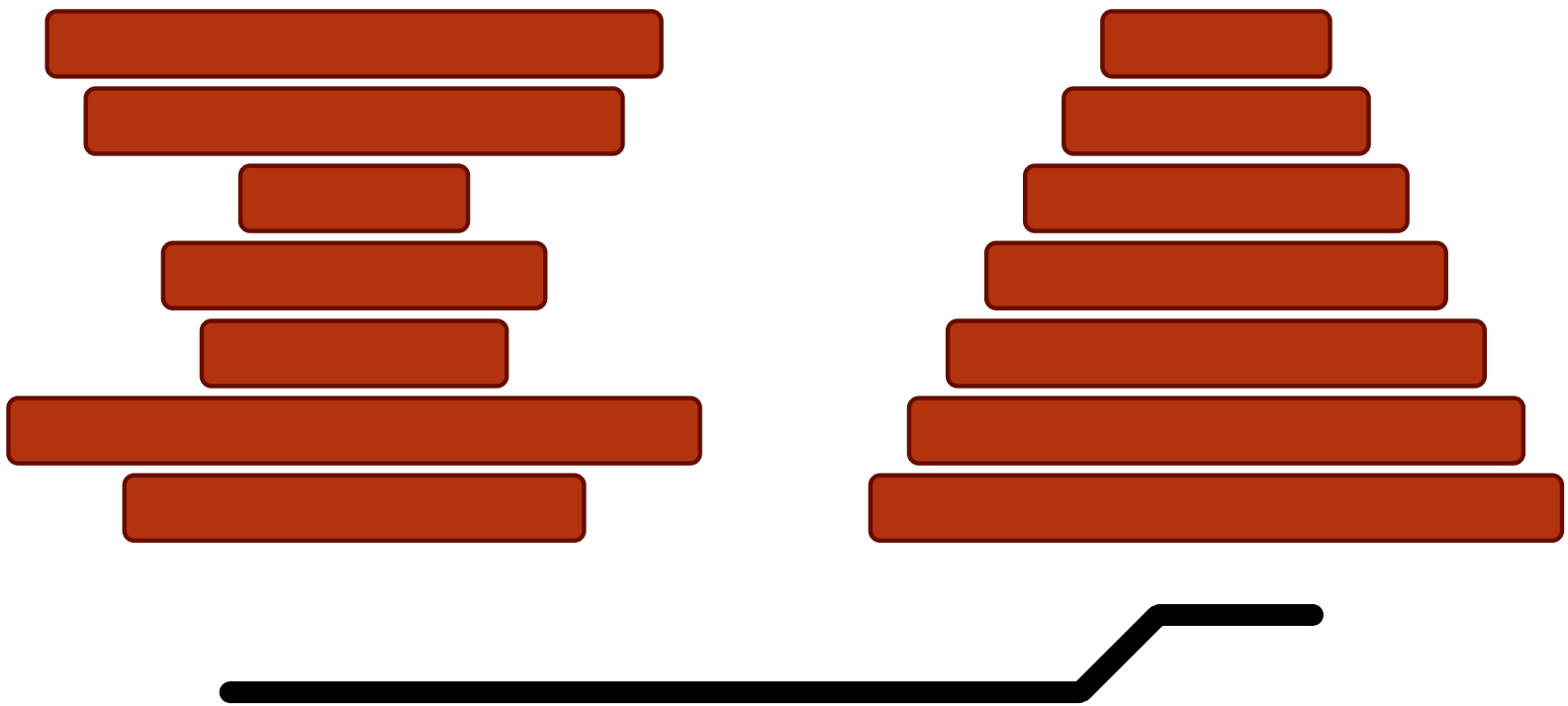
- we are given a stack of pancakes, all of different sizes
- we want to rearrange it into a beautiful pyramidal stack





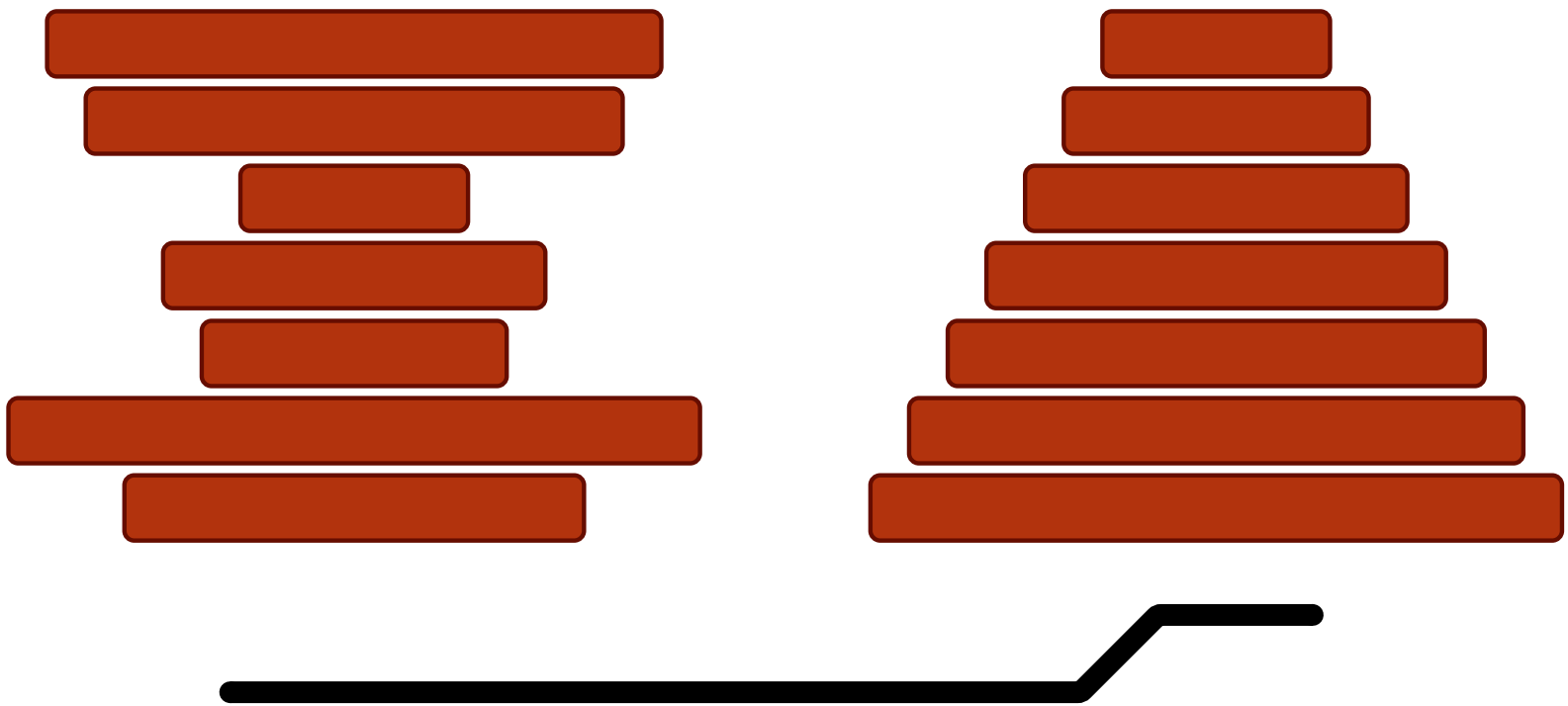
Pancakes

- we are given a stack of pancakes, all of different sizes
- we want to rearrange it into a beautiful pyramidal stack
- we have a spatula, to flip the top of the stack



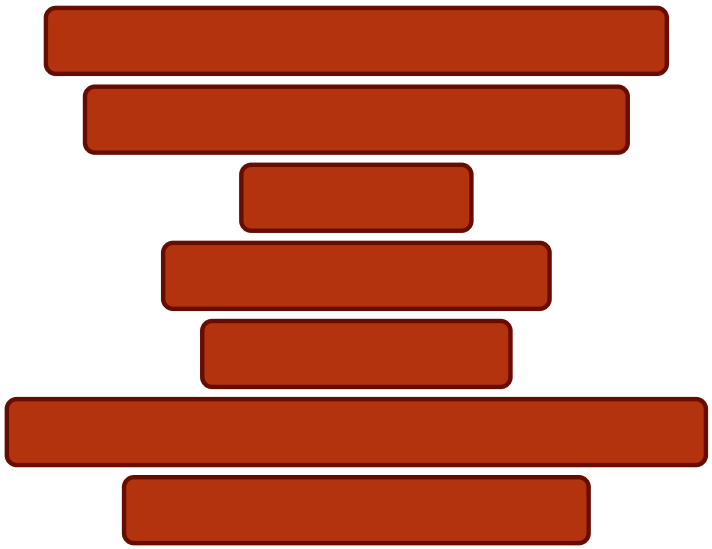
Pancakes

- we are given a stack of pancakes, all of different sizes
- we want to rearrange it into a beautiful pyramidal stack
- we have a spatula, to flip the top of the stack
- we are lazy



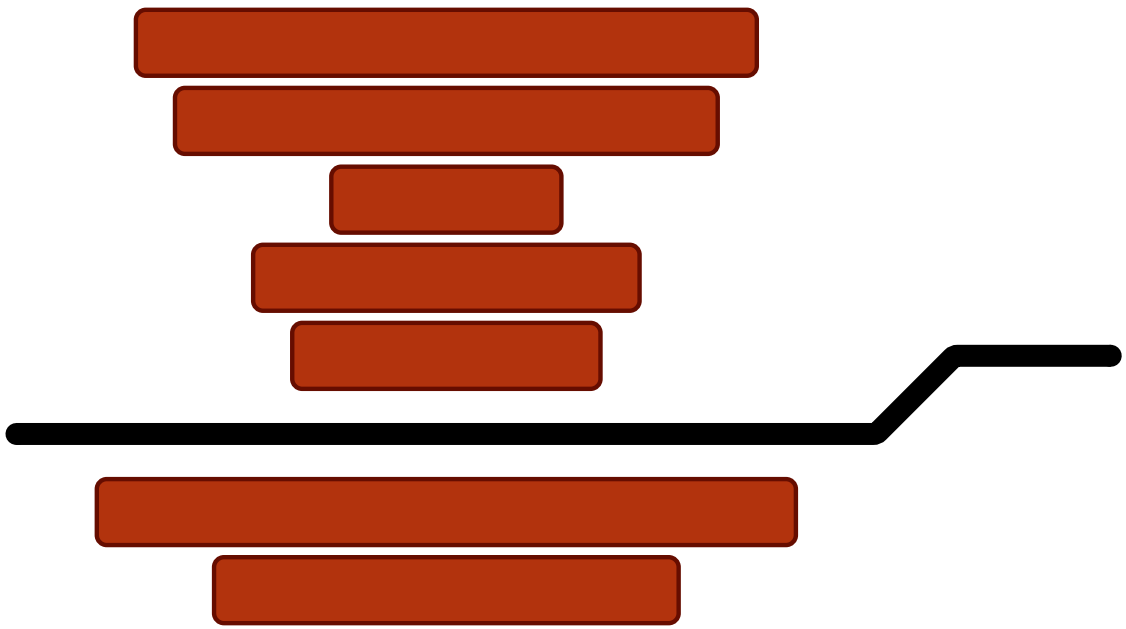
Pancakes

Example



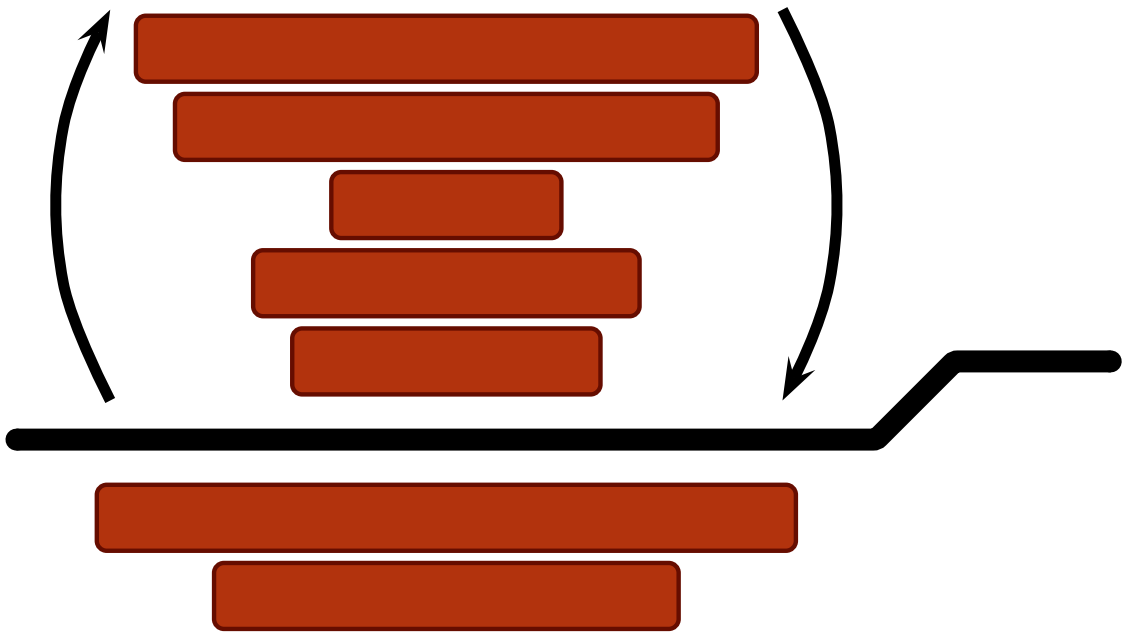
Pancakes

Example



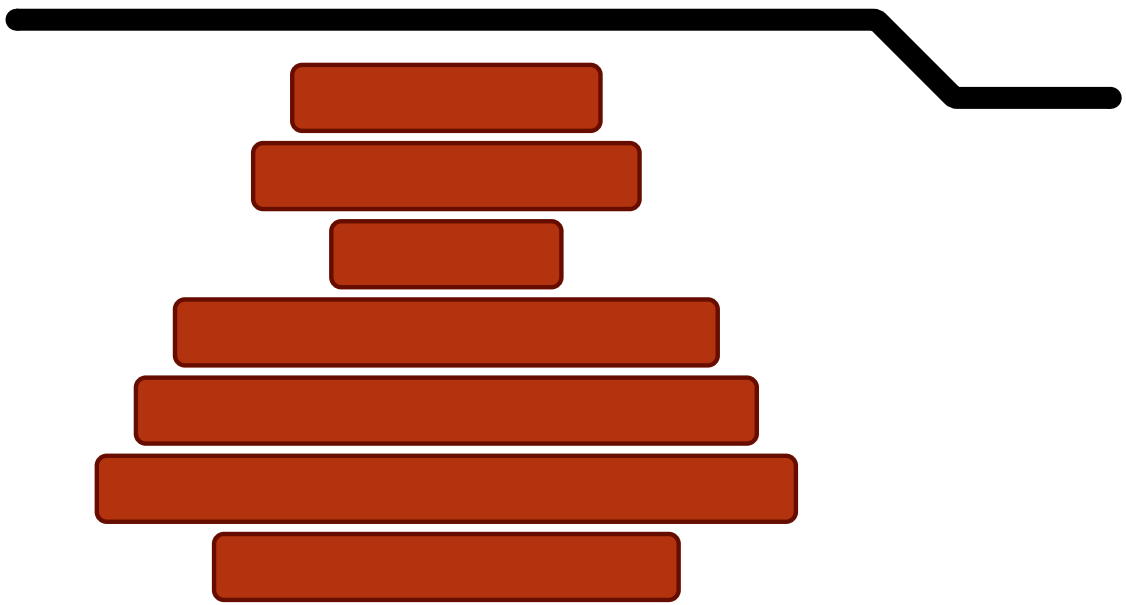
Pancakes

Example



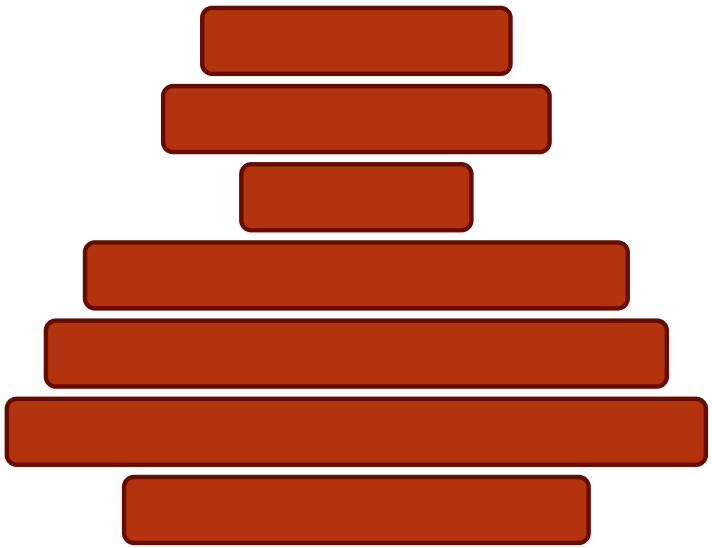
Pancakes

Example



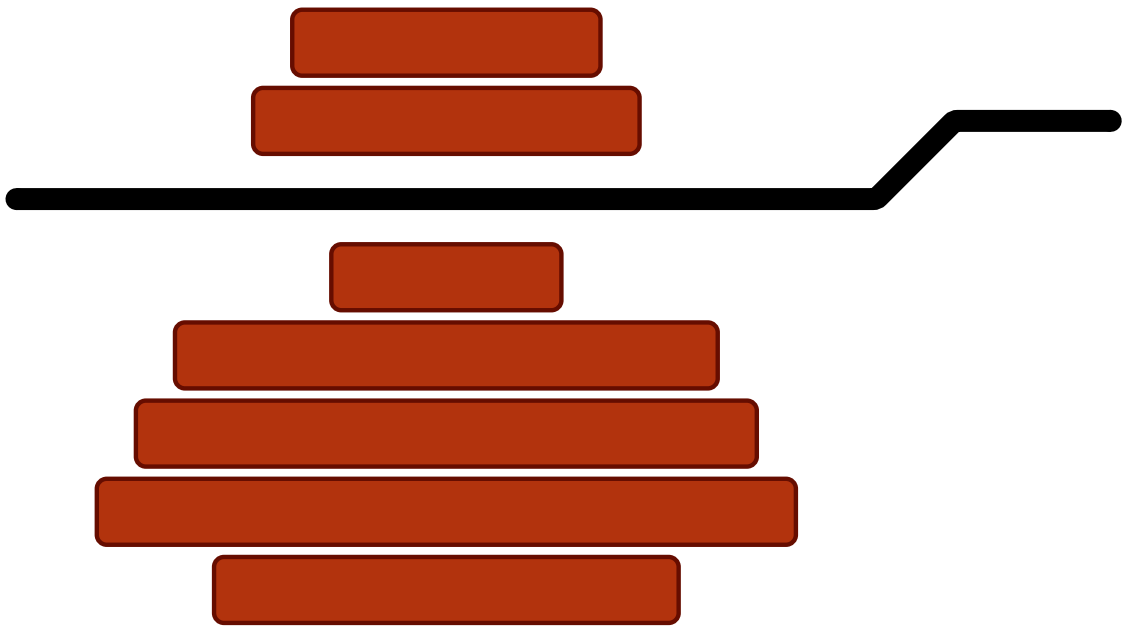
Pancakes

Example



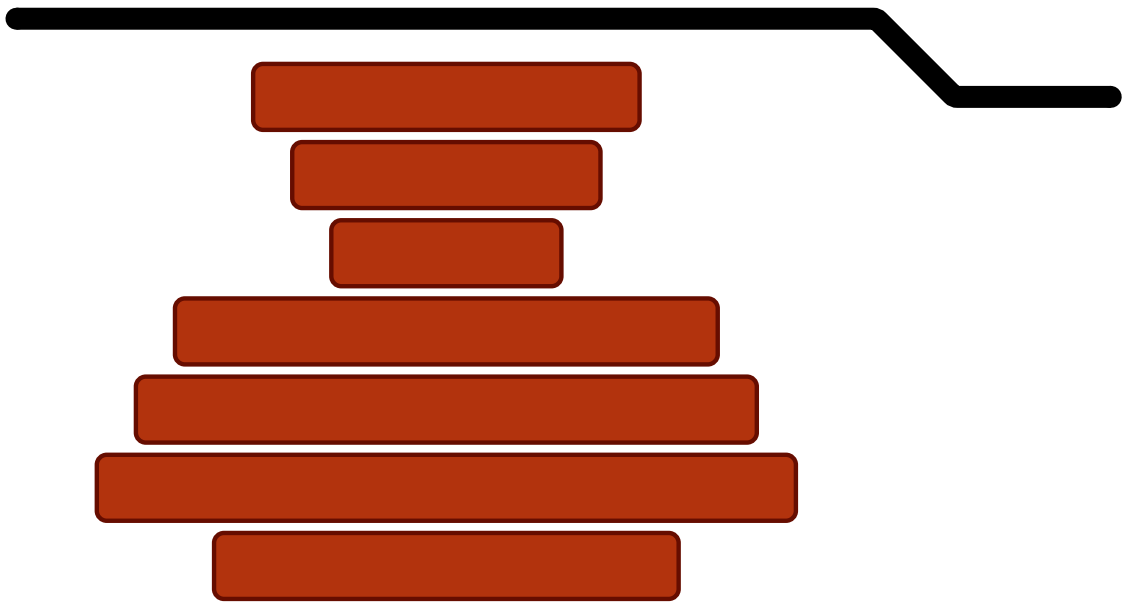
Pancakes

Example



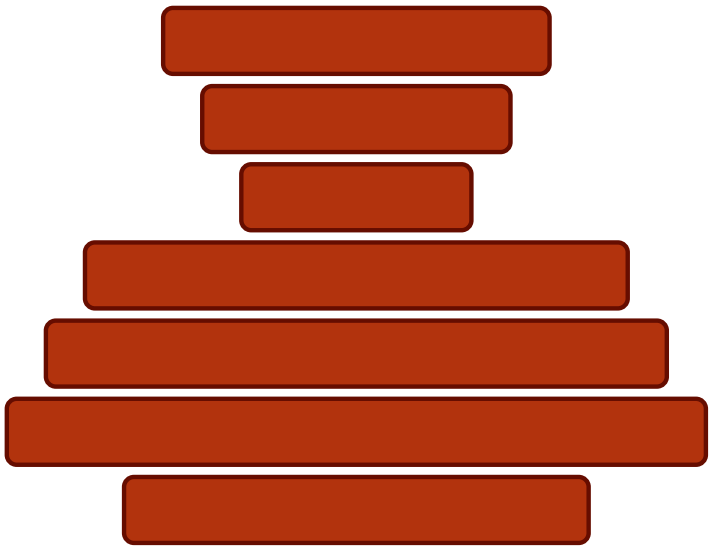
Pancakes

Example



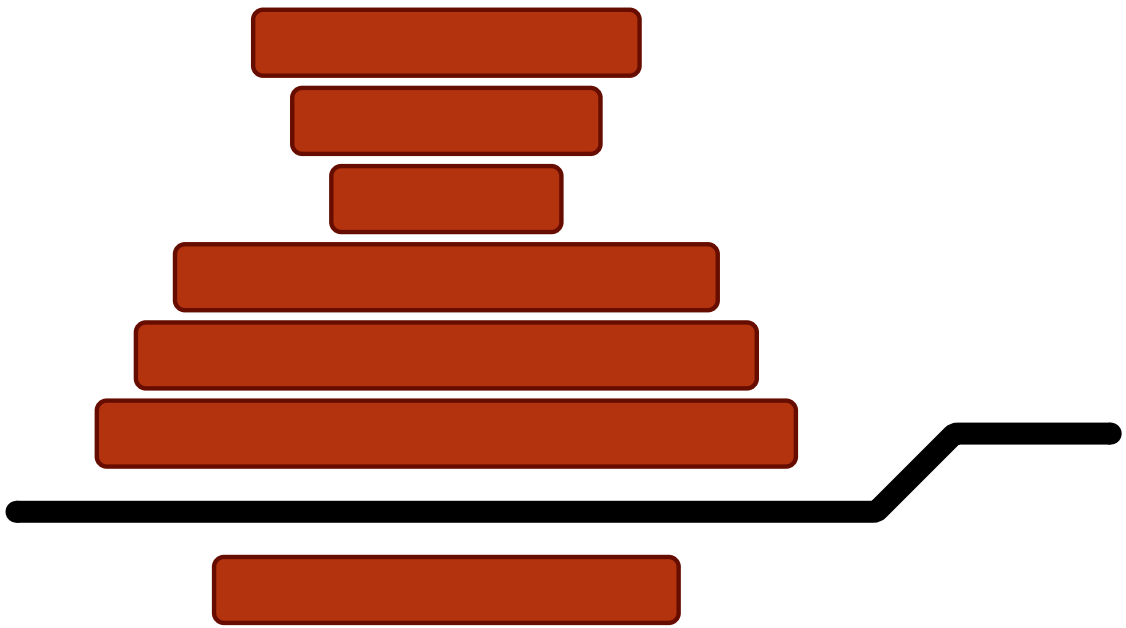
Pancakes

Example



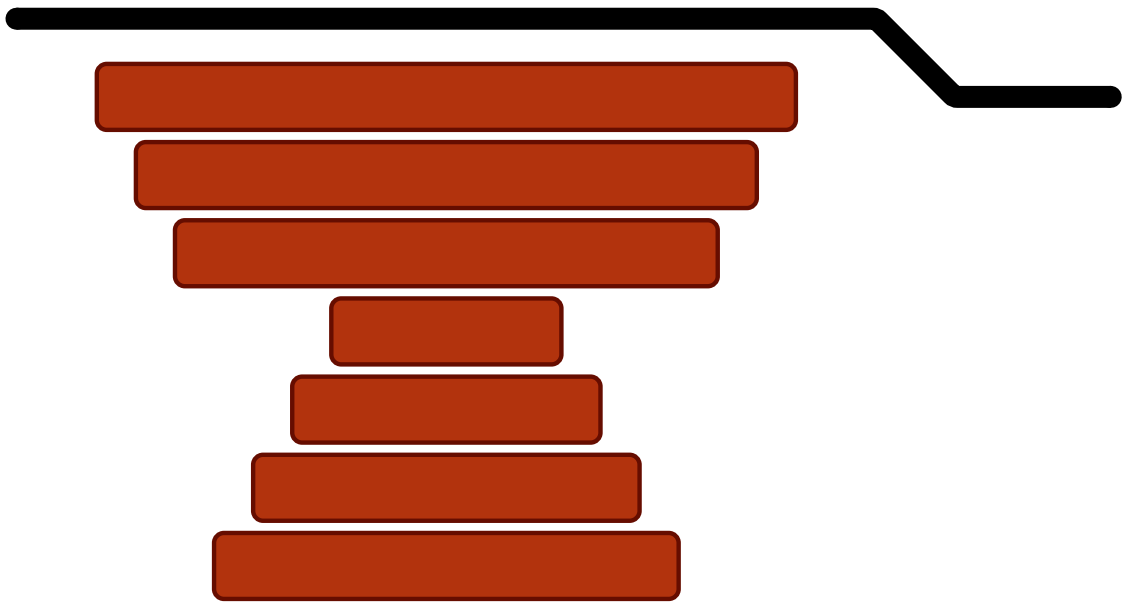
Pancakes

Example



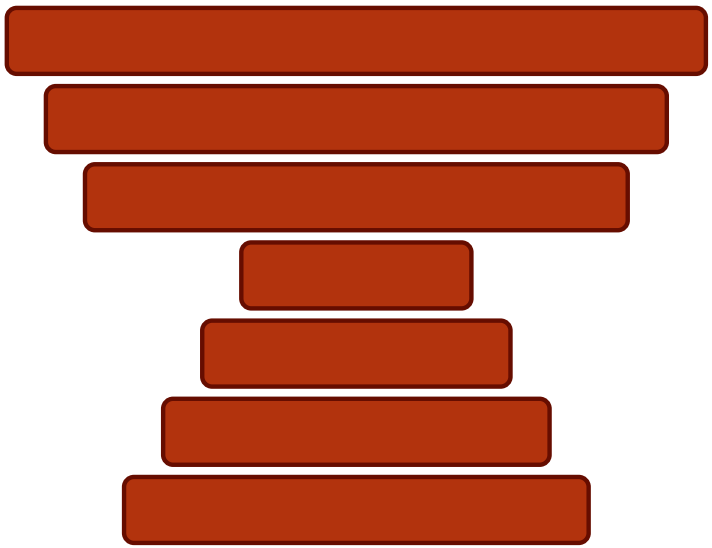
Pancakes

Example



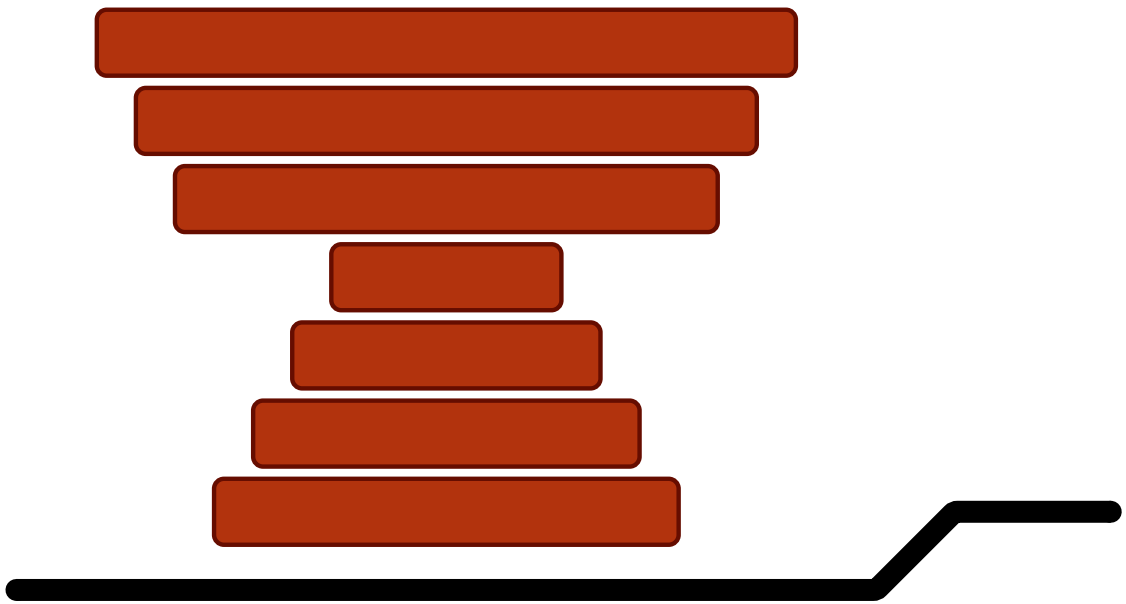
Pancakes

Example



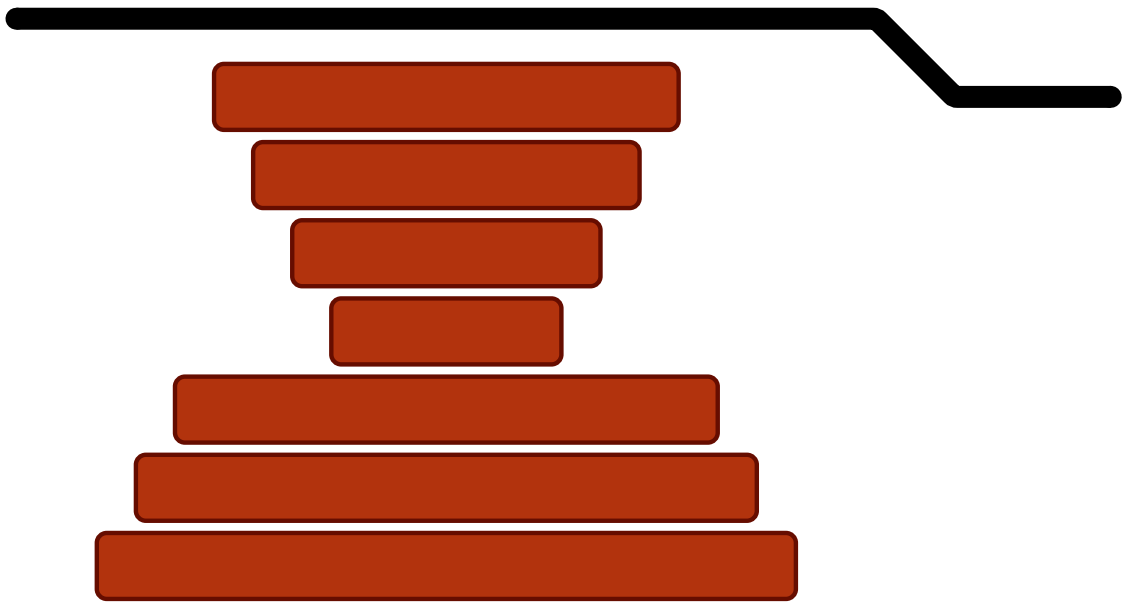
Pancakes

Example



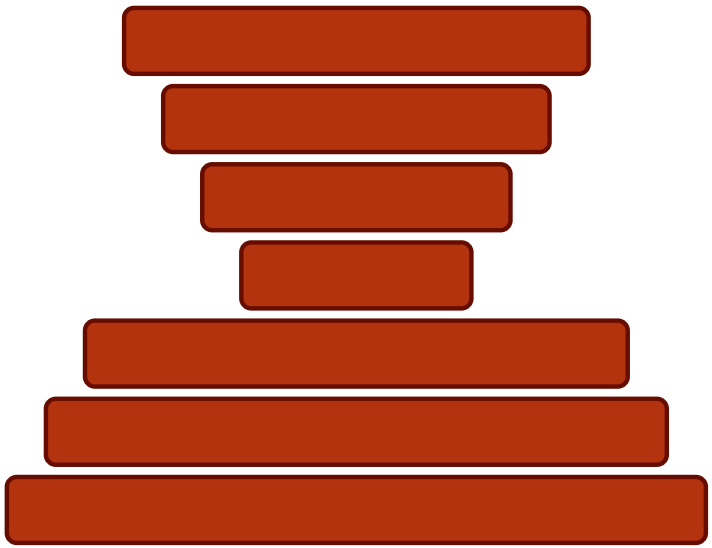
Pancakes

Example



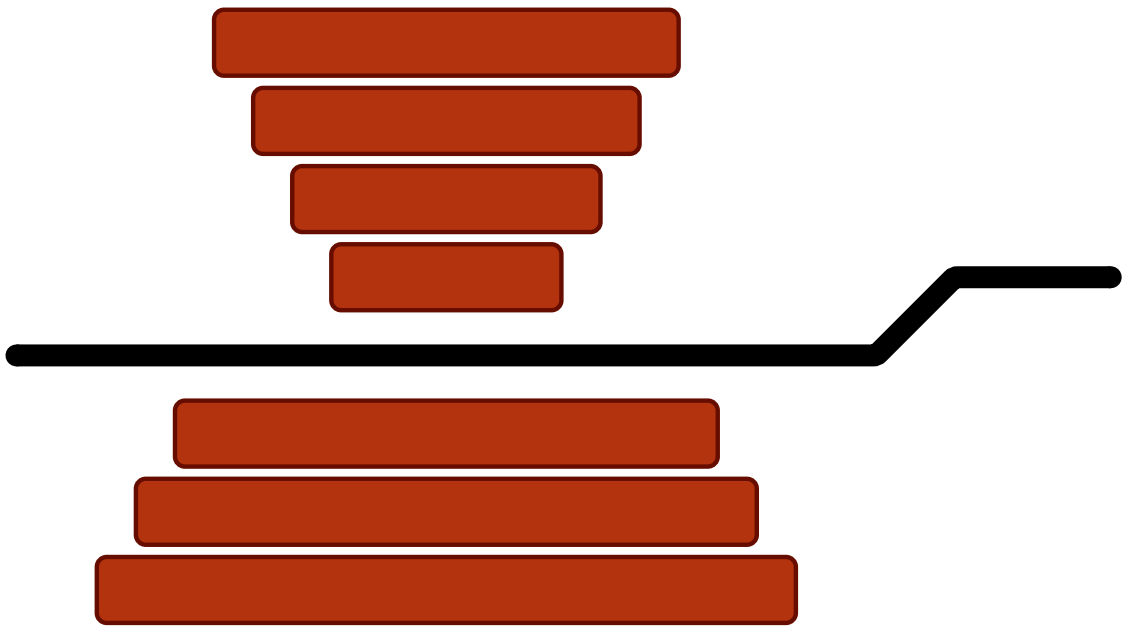
Pancakes

Example



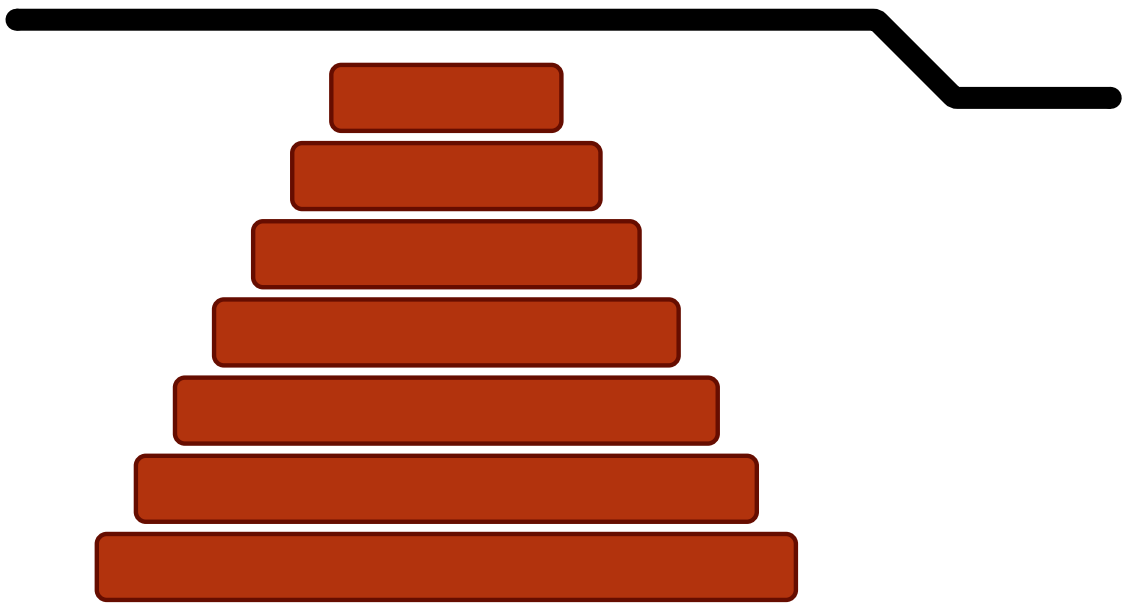
Pancakes

Example



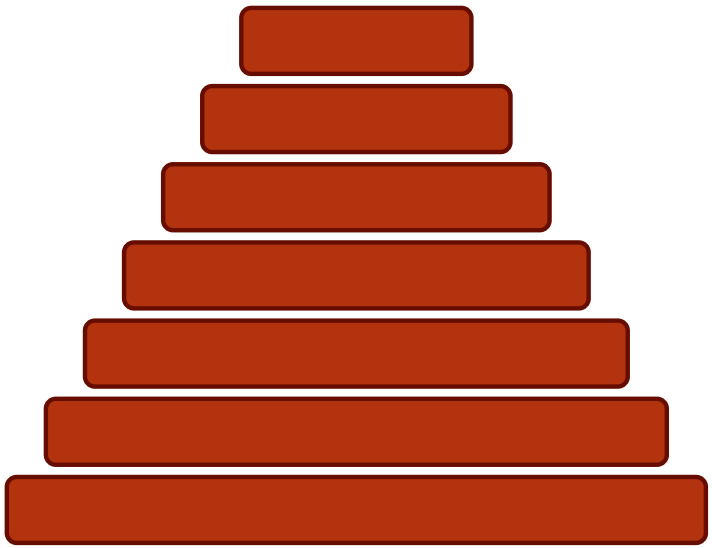
Pancakes

Example



Pancakes

Example



The problem to be solved

Problem

Given a stack of n pancakes, how can it be arranged with as little effort as possible?

Other points of view

"Pancake view"

Given a stack of n pancakes, how can it be arranged with as little effort as possible?

Formal problem : MIN-SBPR

Given a permutation π of $\{1, \ldots, n\}$, compute the **prefix reversal distance** between π and the Identity, written $prd(\pi)$.

"Biology view"

Given two genomes using the same n genes, how many steps have been used in evolution between one and the other?

Other points of view

Pancakes	Formal	Biology
Pancake	Integer	Gene
Stack	Permutation	Genome

Other points of view

Pancakes	Formal	Biology
Pancake	Integer	Gene
Stack	Permutation	Genome
Nice stack	Identity	Reference genome
Flip	Prefix reversal	Evolution step

Other points of view

Pancakes	Formal	Biology
Pancake	Integer	Gene
Stack	Permutation	Genome
Nice stack	Identity	Reference genome
Flip	Prefix reversal	Evolution step
We are lazy	Minimization formulation (distance)	Parsimony principle

Pancake Problem

Complexity: NP-complete

Related results:

- Reversal distance, not necessarily prefix:
  - NP-complete (APX-hard) for unsigned permutations,
  - polynomial for signed permutations.
- Burnt pancakes variant, or Prefix Reversal Distance for signed permutations:
  - complexity unknown.
- Algorithms:
  - polynomial-time algorithm for a subclass of signed permutations (simple permutations [Labarre, Cibulka, 2011])
  - 2-approximation algorithm

Known bounds

Upper bound

$prd(\pi) \leq 2(n-1)$

- Repeat at most $n-1$ times
  - Find the largest unsorted pancake, flip it to the top
  - Flip it back to its destination
- At most $2(n-1)$ flips to sort a stack.

Known bounds

Upper bound

$prd(\pi) \leq 2(n-1)$
Anything better ?

Known bounds

Upper bound

~~$prd(\pi) \leq 2(n-1)$~~

Anything better ? – Yes :

$prd(\pi) \leq (5n+5)/3$ [Gates & Papadimitriou, 1979]

Known bounds

Upper bound

~~$prd(\pi) \leq 2(n-1)$~~
Anything better ? – Yes :
~~$prd(\pi) \leq (5n+5)/3$~~ [Gates & Papadimitriou, 1979]
$prd(\pi) \leq (18n)/11 + O(1)$ [Chitturi et al., 2009]

Known bounds

Upper bound

$prd(\pi) \leq (18n)/11 + O(1)$ [Chitturi et al., 2009]

Lower bound

$prd(\pi) \geq d_b(\pi)$

- Breakpoint at position i if:
  - $i < n$ and $\pi(i+1) \neq \pi(i) \pm 1$
  - $i = n$ and $\pi(n) \neq n$
- $d_b(\pi)$: number of breakpoints

Known bounds

Upper bound

$prd(\pi) \leq (18n)/11 + O(1)$ [Chitturi et al., 2009]

Lower bound

$prd(\pi) \geq d_b(\pi)$

- Breakpoint at position i if:
  - $i < n$ and $\pi(i+1) \neq \pi(i) \pm 1$
  - $i = n$ and $\pi(n) \neq n$
- $d_b(\pi)$: number of breakpoints
- **At most one breakpoint is removed with each flip**

Our result

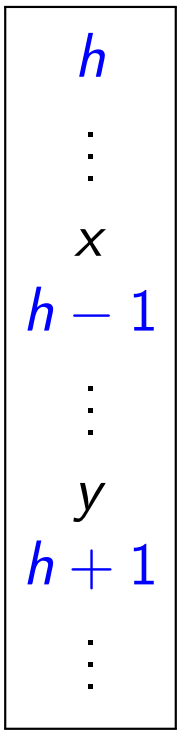
- Reduction from 3-SAT: from a formula ϕ, create a permutation π_ϕ such that $prd(\pi_\phi) = d_b(\pi_\phi)$ iff ϕ is satisfiable.
- Given a permutation π, deciding wether π can be sorted with no more than $d_b(\pi)$ flips is NP-hard.

- **MIN-SBPR is NP-hard** (hence NP-complete)

Reduction

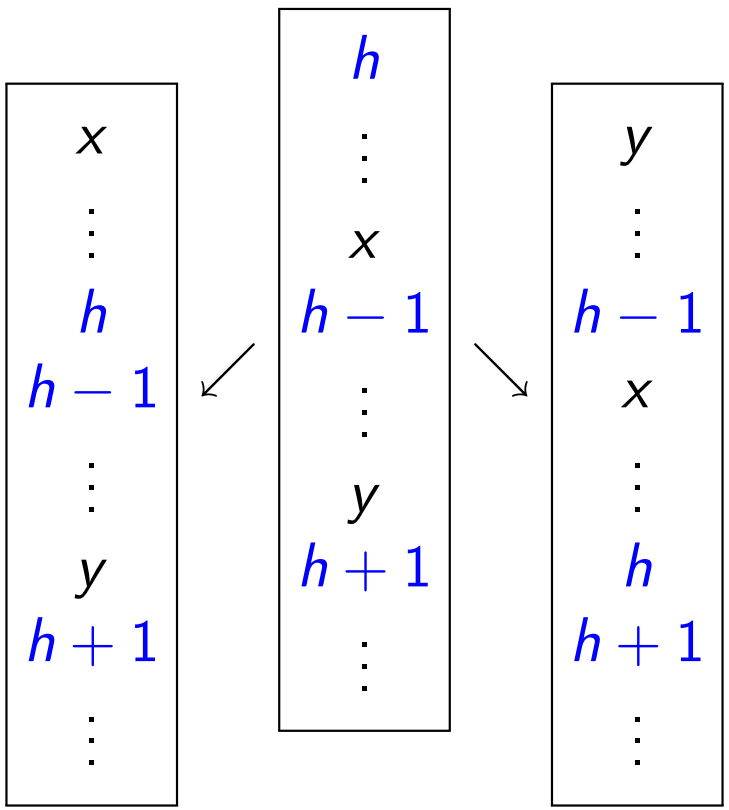
Efficient flips

- A flip is **efficient** if it removes one breakpoint: $\pi \longrightarrow \pi'$
- $prd(\pi_\phi) = d_b(\pi_\phi)$ iff there exists a "path" of efficient flips from π_ϕ to the Identity. $\pi_\phi \longrightarrow^* \mathcal{I}_n^1$
- At most two efficient flips are possible from every permutation



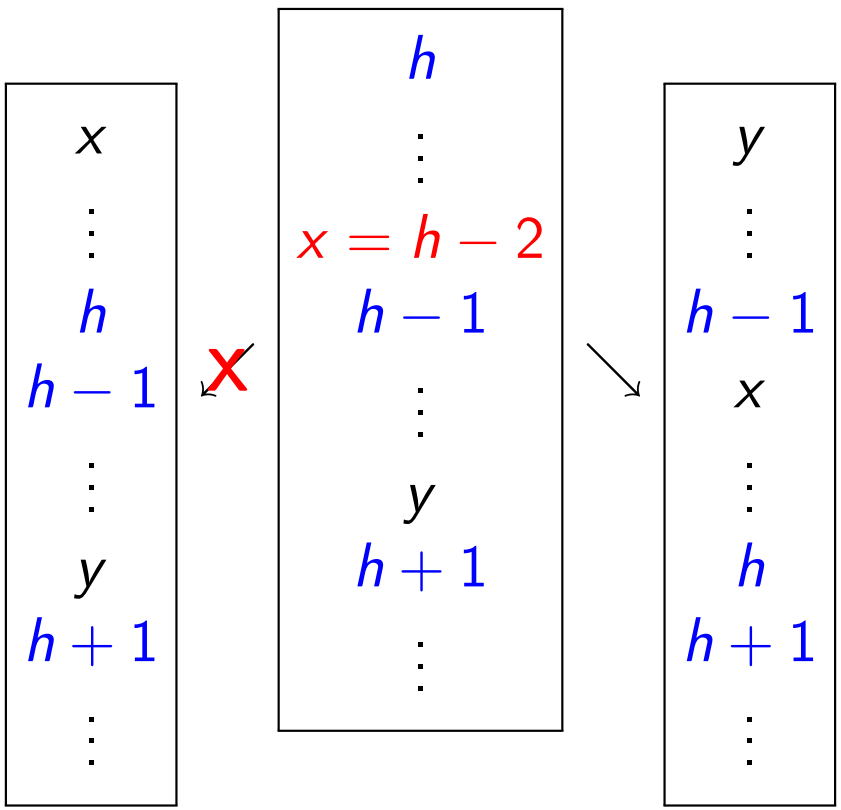
Efficient flips

- A flip is **efficient** if it removes one breakpoint: $\pi \longrightarrow \pi'$
- $prd(\pi_\phi) = d_b(\pi_\phi)$ iff there exists a "path" of efficient flips from π_ϕ to the Identity. $\pi_\phi \longrightarrow^* \mathcal{I}_n^1$
- At most two efficient flips are possible from every permutation



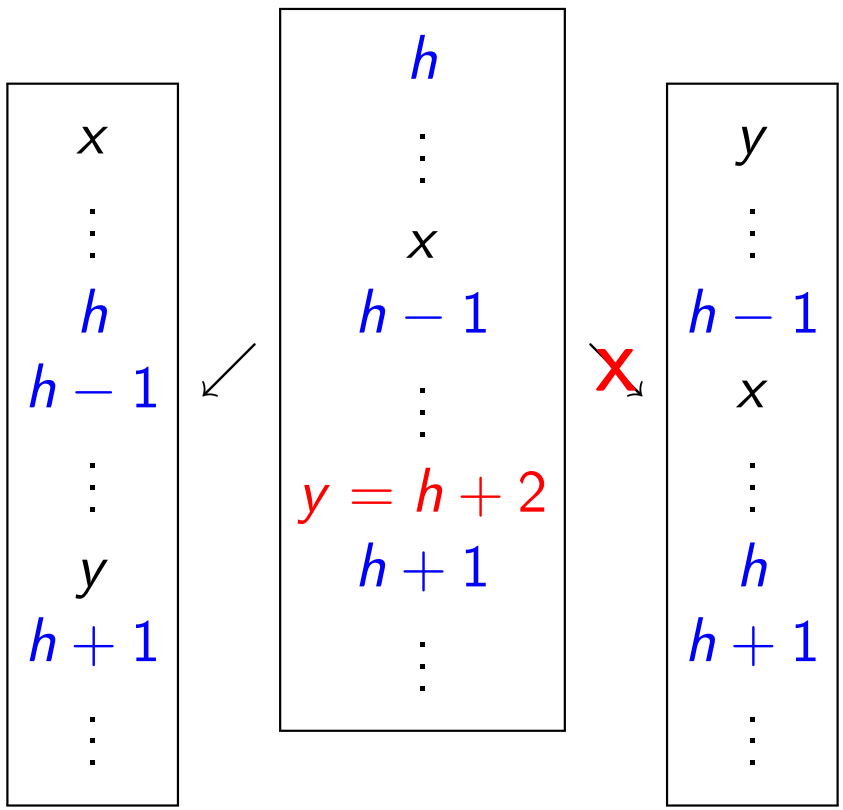
Efficient flips

- A flip is **efficient** if it removes one breakpoint: $\pi \longrightarrow \pi'$
- $prd(\pi_\phi) = d_b(\pi_\phi)$ iff there exists a "path" of efficient flips from π_ϕ to the Identity. $\pi_\phi \longrightarrow^* \mathcal{I}_n^1$
- At most two efficient flips are possible from every permutation



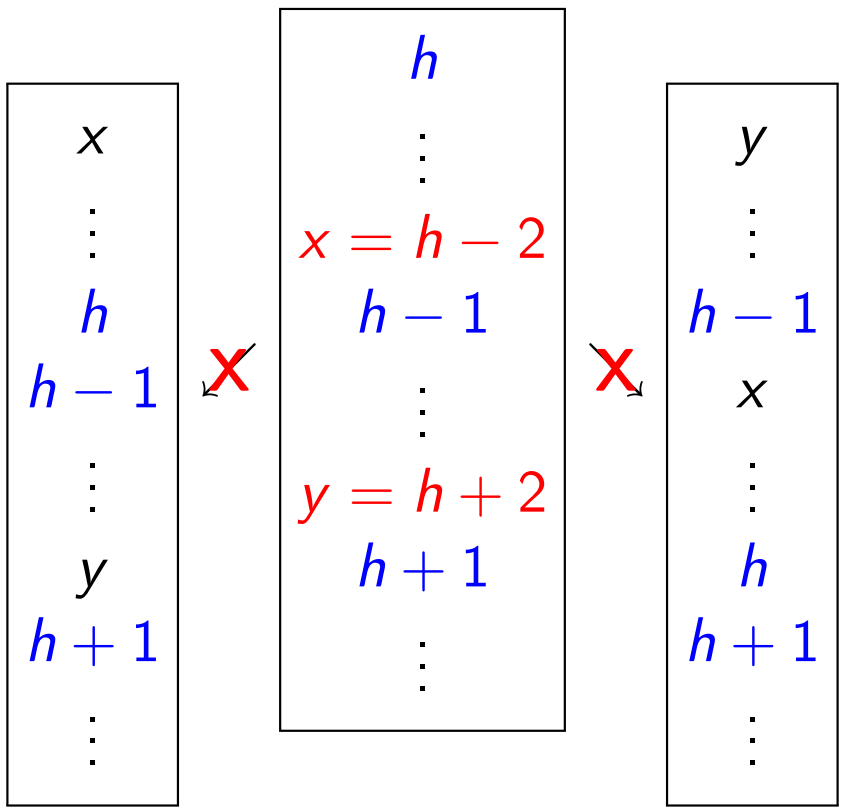
Efficient flips

- A flip is **efficient** if it removes one breakpoint: $\pi \longrightarrow \pi'$
- $prd(\pi_\phi) = d_b(\pi_\phi)$ iff there exists a "path" of efficient flips from π_ϕ to the Identity. $\pi_\phi \longrightarrow^* \mathcal{I}_n^1$
- At most two efficient flips are possible from every permutation



Efficient flips

- A flip is **efficient** if it removes one breakpoint: $\pi \longrightarrow \pi'$
- $prd(\pi_\phi) = d_b(\pi_\phi)$ iff there exists a "path" of efficient flips from π_ϕ to the Identity. $\pi_\phi \longrightarrow^* \mathcal{I}_n^1$
- At most two efficient flips are possible from every permutation



Reduction ideas

Create π_ϕ in order to know precisely which efficient flips are possible (from π_ϕ or subsequent permutations)

- One possible flip: usual case, there is one path to follow
- Two possible flips: a choice has to be made

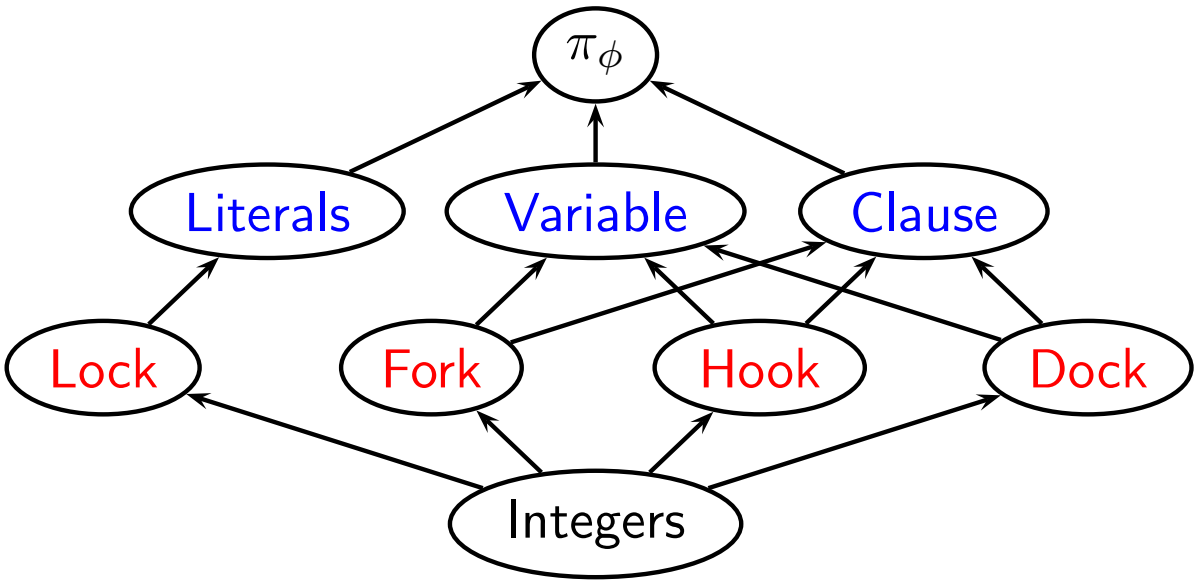
  e.g., assigning "true" or "false" to a variable
- No possible flip: bad choices have been made

  e.g., a clause is unsatisfied

Necessity to end with the Identity permutation.

Gadgets

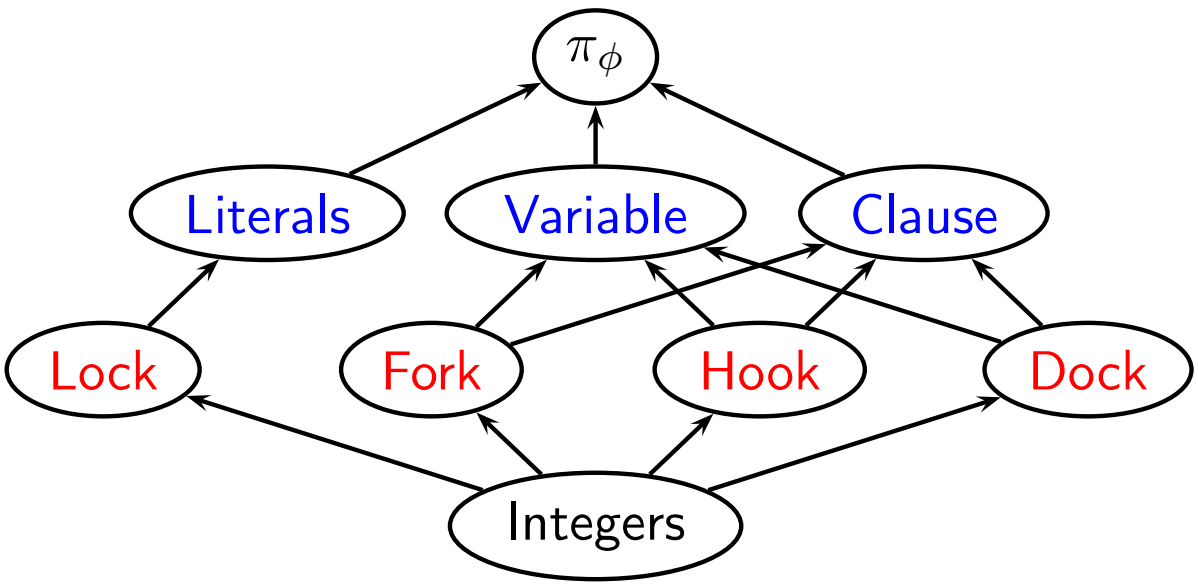
Lock 3 states: closed, open, tested



Gadgets

Lock 3 states: closed, open, tested

Fork chooses between two options

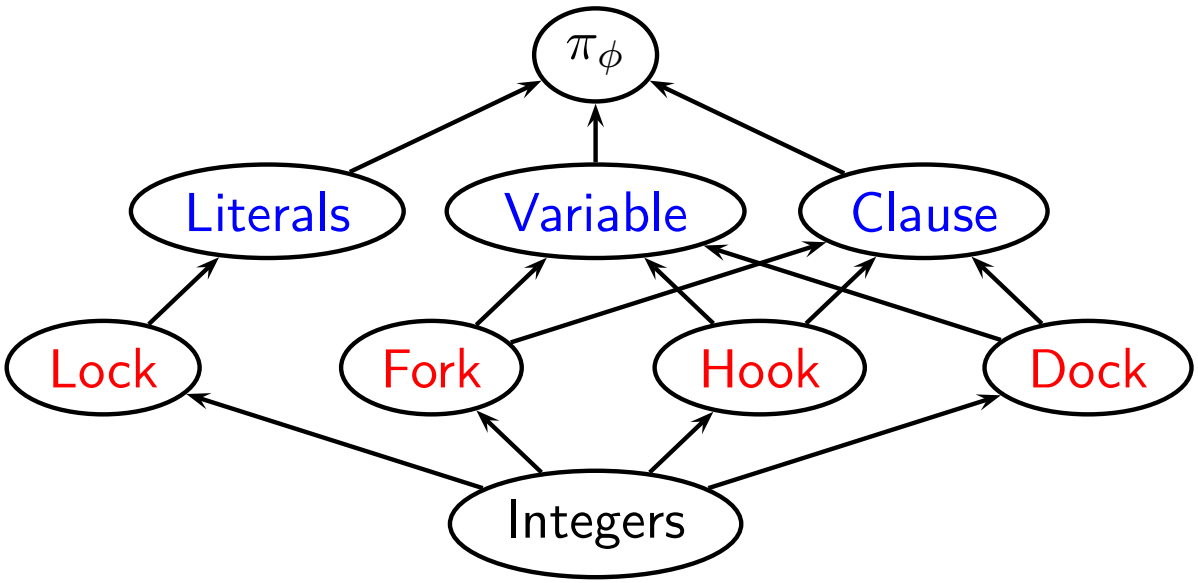


Gadgets

Lock 3 states: closed, open, tested

Fork chooses between two options

Hook moves a subsequence in/out of the head



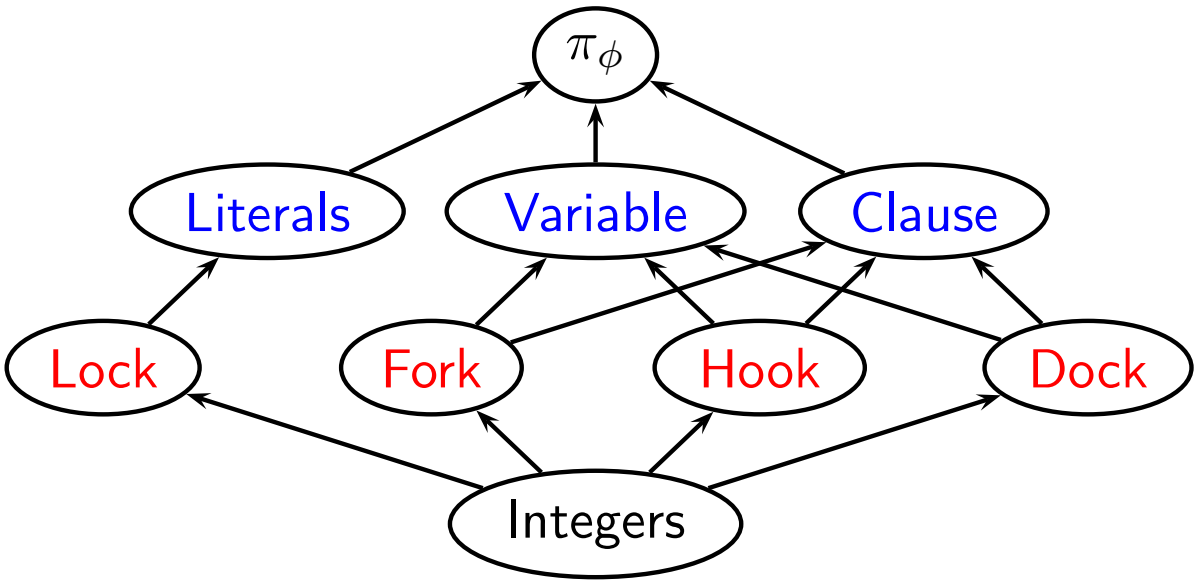
Gadgets

Lock 3 states: closed, open, tested

Fork chooses between two options

Hook moves a subsequence in/out of the head

Dock stores subsequences when they are sorted



Gadgets

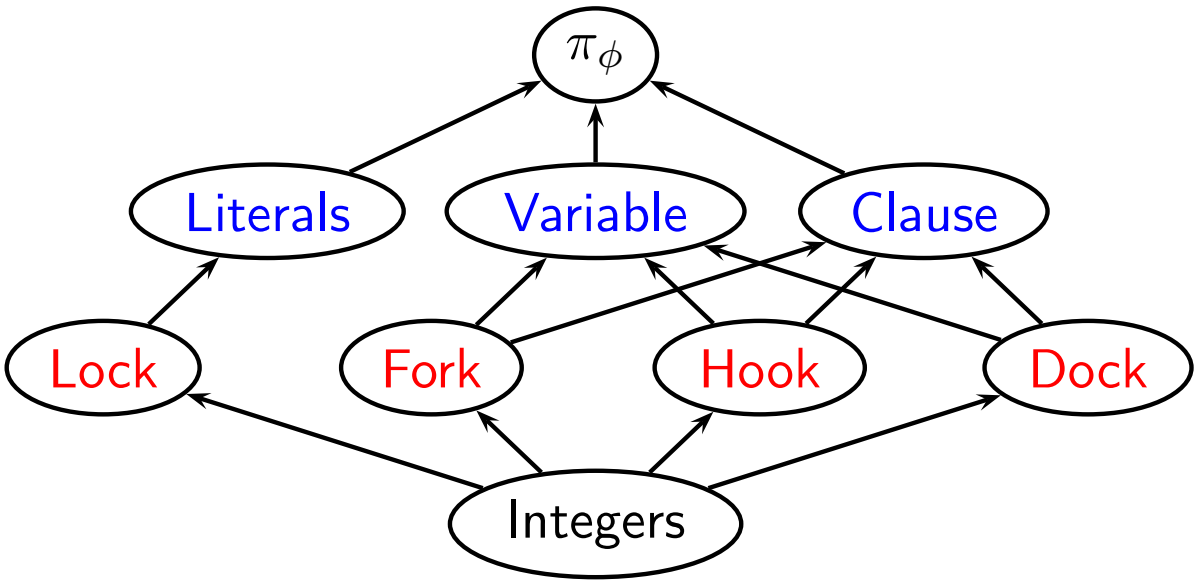
Lock 3 states: closed, open, tested

Fork chooses between two options

Hook moves a subsequence in/out of the head

Dock stores subsequences when they are sorted

Literals holds a lock for each literal in the formula



Gadgets

Lock 3 states: closed, open, tested

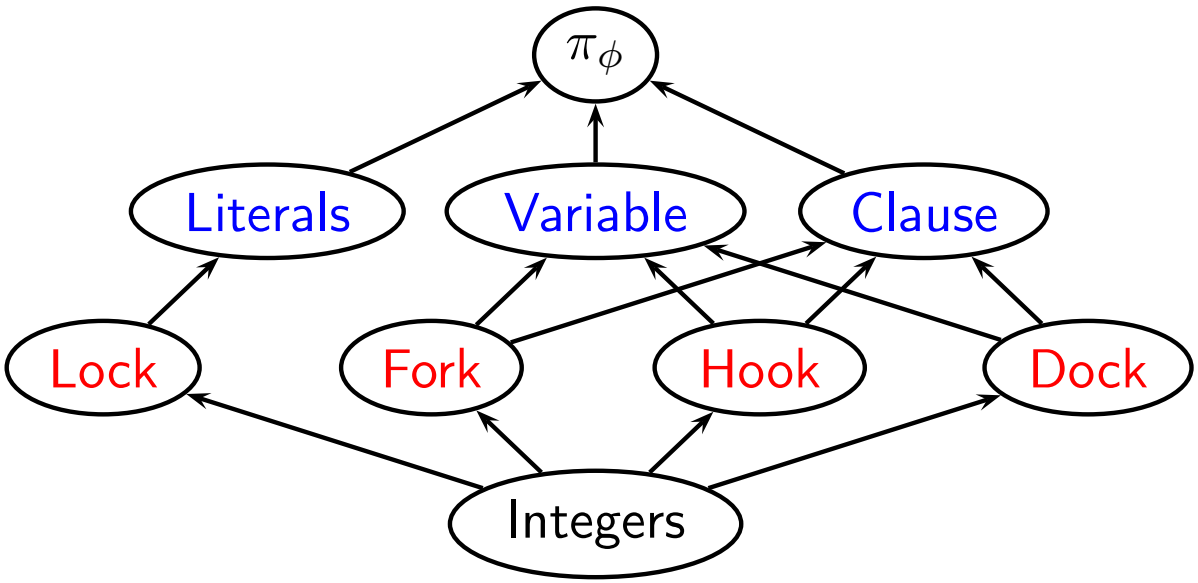
Fork chooses between two options

Hook moves a subsequence in/out of the head

Dock stores subsequences when they are sorted

Literals holds a lock for each literal in the formula

Variable opens locks corresponding to either x or $\neg x$



Gadgets

Lock 3 states: closed, open, tested

Fork chooses between two options

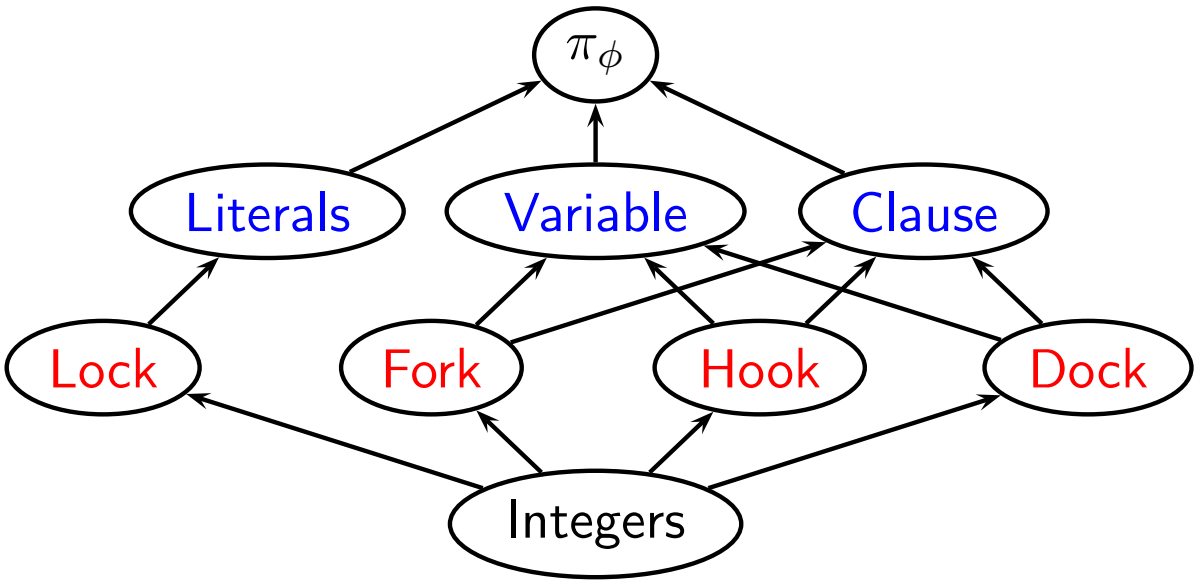
Hook moves a subsequence in/out of the head

Dock stores subsequences when they are sorted

Literals holds a lock for each literal in the formula

Variable opens locks corresponding to either x or $\neg x$

Clause tests one lock out of three



Lock gadget

Lock gadget (closed)

$$L = \begin{bmatrix} 1 \\ 2 \\ 9 \\ 8 \\ 5 \\ 6 \\ 4 \\ 3 \\ 11 \\ 12 \end{bmatrix}$$

$key = 10$

$test = 7$

Lock gadget

Lock gadget (closed)

$$L = \begin{bmatrix} 1 \\ 2 \\ 9 \\ 8 \\ 5 \\ 6 \\ 4 \\ 3 \\ 11 \\ 12 \end{bmatrix}$$

Open

$$L^o = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 6 \\ 5 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \end{bmatrix}$$

$key = 10$

$test = 7$

Lock gadget

Lock gadget (closed)

$$L = \begin{bmatrix} 1 \\ 2 \\ 9 \\ 8 \\ 5 \\ 6 \\ 4 \\ 3 \\ 11 \\ 12 \end{bmatrix}$$

$key = 10$

$test = 7$

Open

$$L^o = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 6 \\ 5 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \end{bmatrix}$$

Tested

$$\mathcal{I}_{12}^1 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \end{bmatrix}$$

Lock gadget

Lock gadget (closed)

$$L = \begin{bmatrix} p+1 \\ p+2 \\ p+9 \\ p+8 \\ p+5 \\ p+6 \\ p+4 \\ p+3 \\ p+11 \\ p+12 \end{bmatrix}$$

$key = p + 10$

$test = p + 7$

Open

$$L^o = \begin{bmatrix} p+1 \\ p+2 \\ p+3 \\ p+4 \\ p+6 \\ p+5 \\ p+8 \\ p+9 \\ p+10 \\ p+11 \\ p+12 \end{bmatrix}$$

Tested

$$\mathcal{I}_{p+12}^{p+1} = \begin{bmatrix} p+1 \\ p+2 \\ p+3 \\ p+4 \\ p+5 \\ p+6 \\ p+7 \\ p+8 \\ p+9 \\ p+10 \\ p+11 \\ p+12 \end{bmatrix}$$

Lock gadget

Lock gadget (closed)

$$L = \begin{bmatrix} 1 \\ 2 \\ 9 \\ 8 \\ 5 \\ 6 \\ 4 \\ 3 \\ 11 \\ 12 \end{bmatrix}$$

$key = 10$

$test = 7$

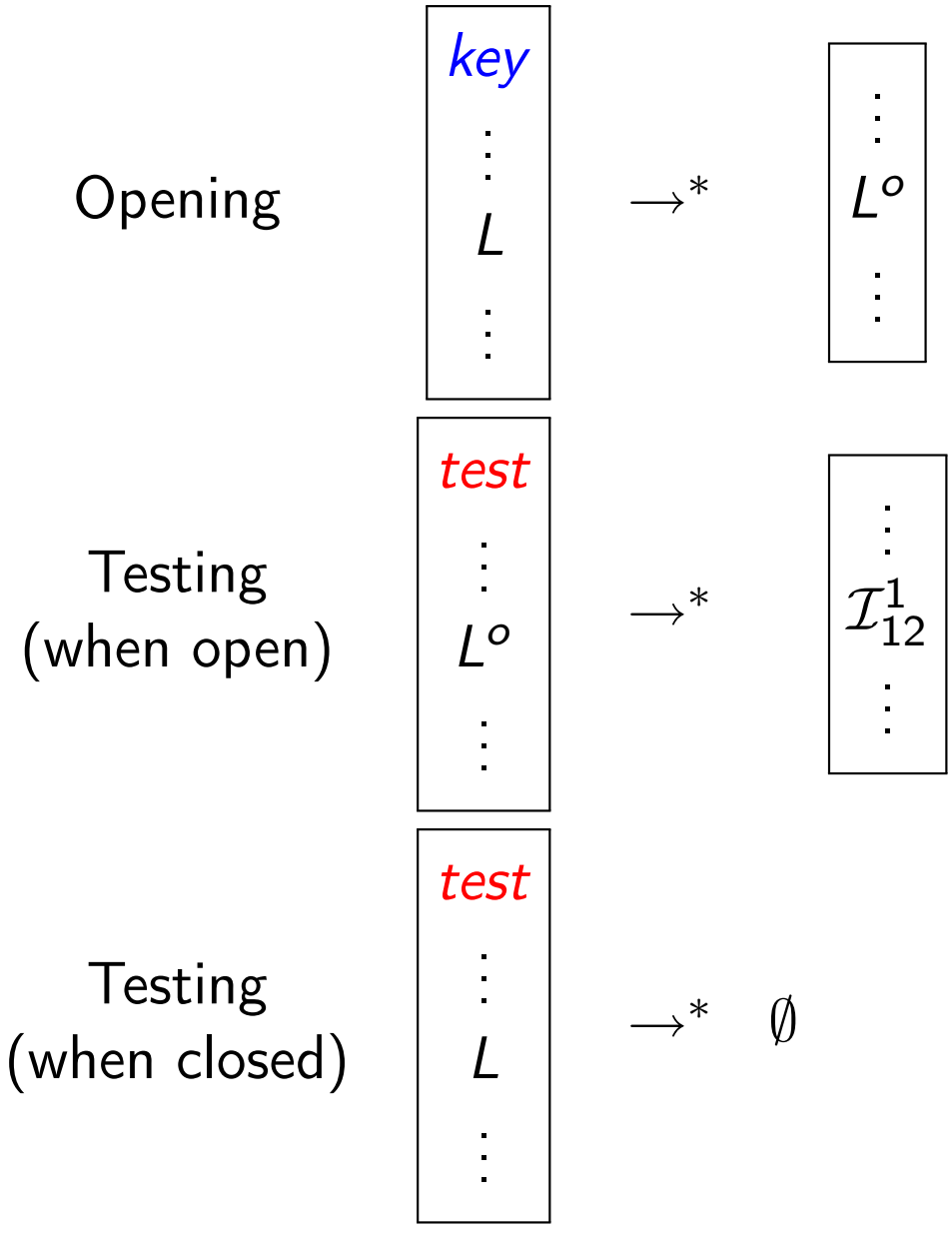
Open

$$L^o = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 6 \\ 5 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \end{bmatrix}$$

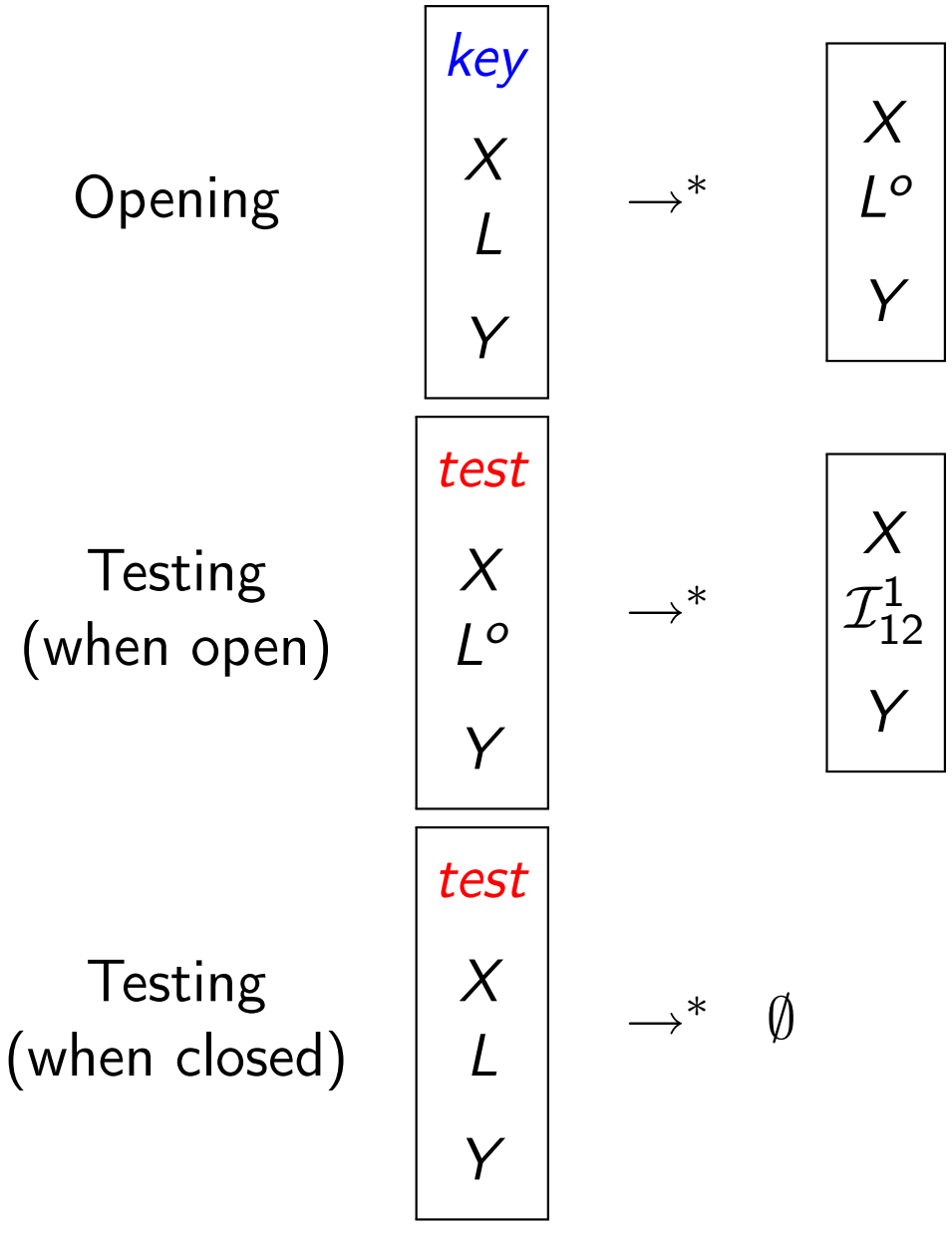
Tested

$$\mathcal{I}_{12}^1 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \end{bmatrix}$$

Lock gadget


Opening
key
L
$\rightarrow^*$
L^o
Testing
(when open)
test
L^o
$\rightarrow^*$
$\mathcal{I}_{12}^1$
Testing
(when closed)
test
L
$\rightarrow^*$
$\emptyset$

Lock gadget


Opening
key
X
L
Y
→*
X
L^o
Y
Testing
(when open)
test
X
L^o
Y
→*
X
I^1_12
Y
Testing
(when closed)
test
X
L
Y
→*
∅

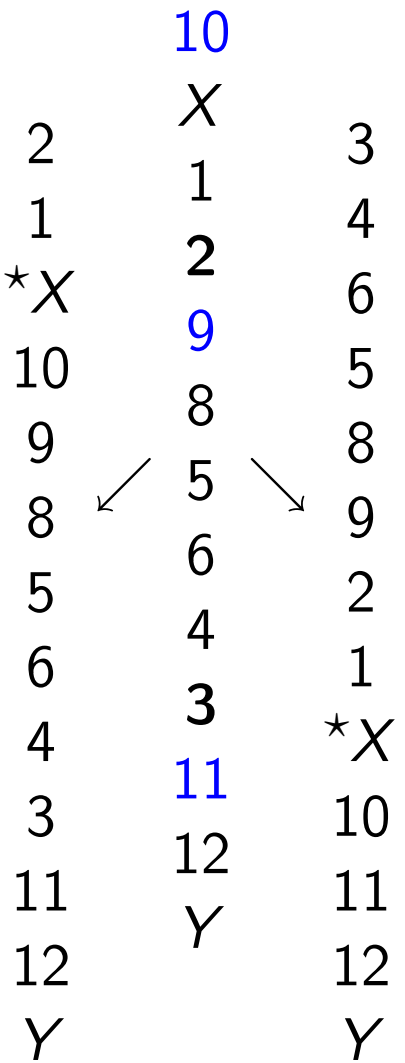
Lock gadget

key
X
L
Y

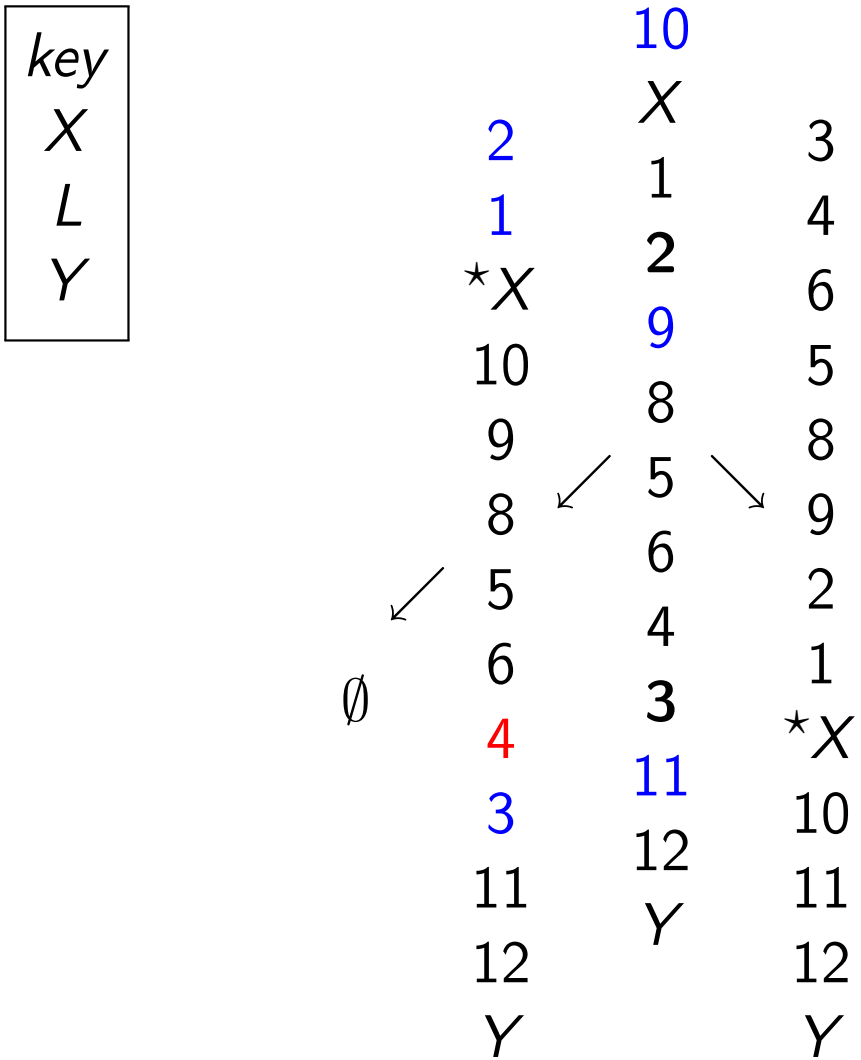
10
X
1
2
9
8
5
6
4
3
11
12
Y

Lock gadget

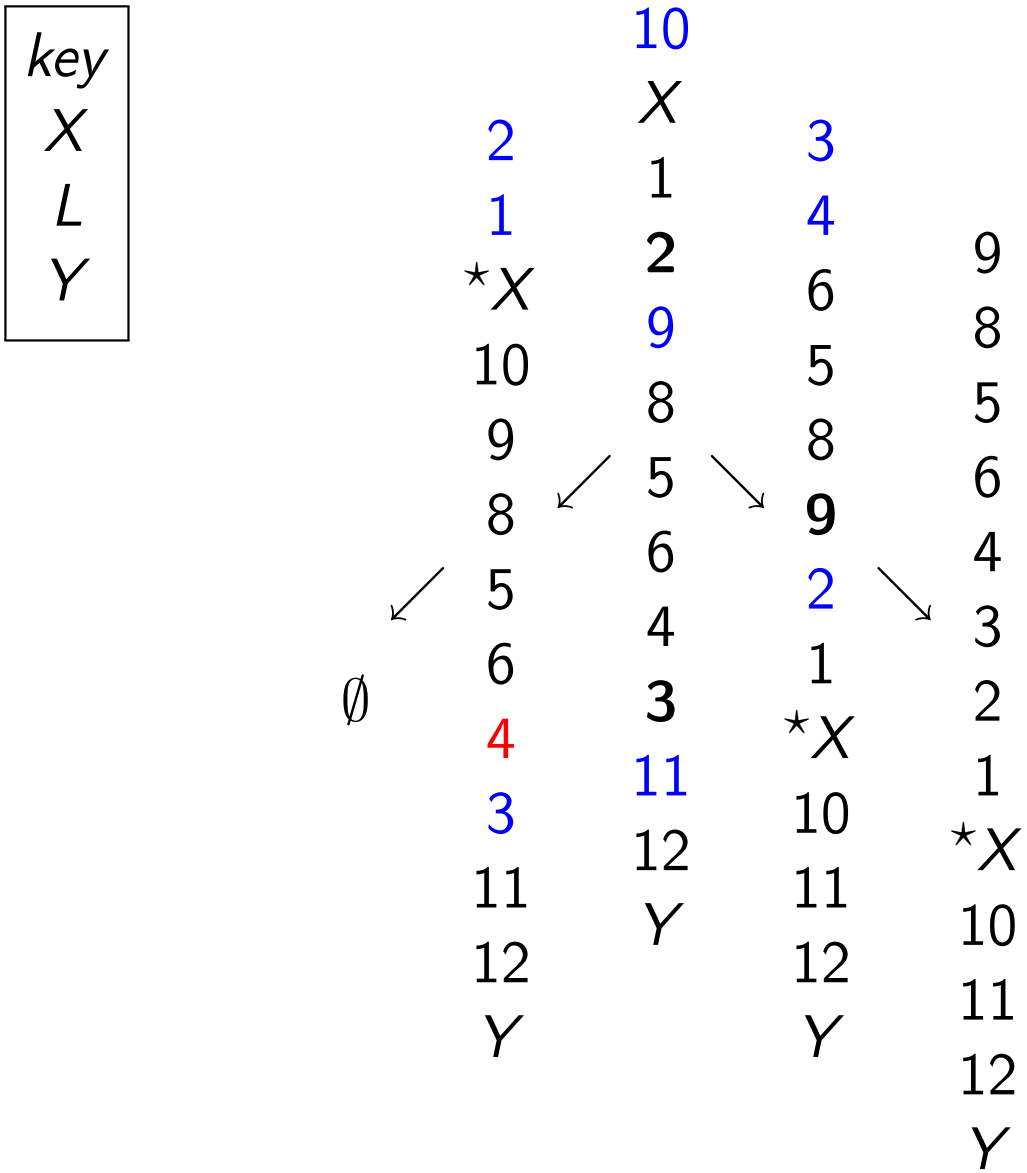
key
X
L
Y



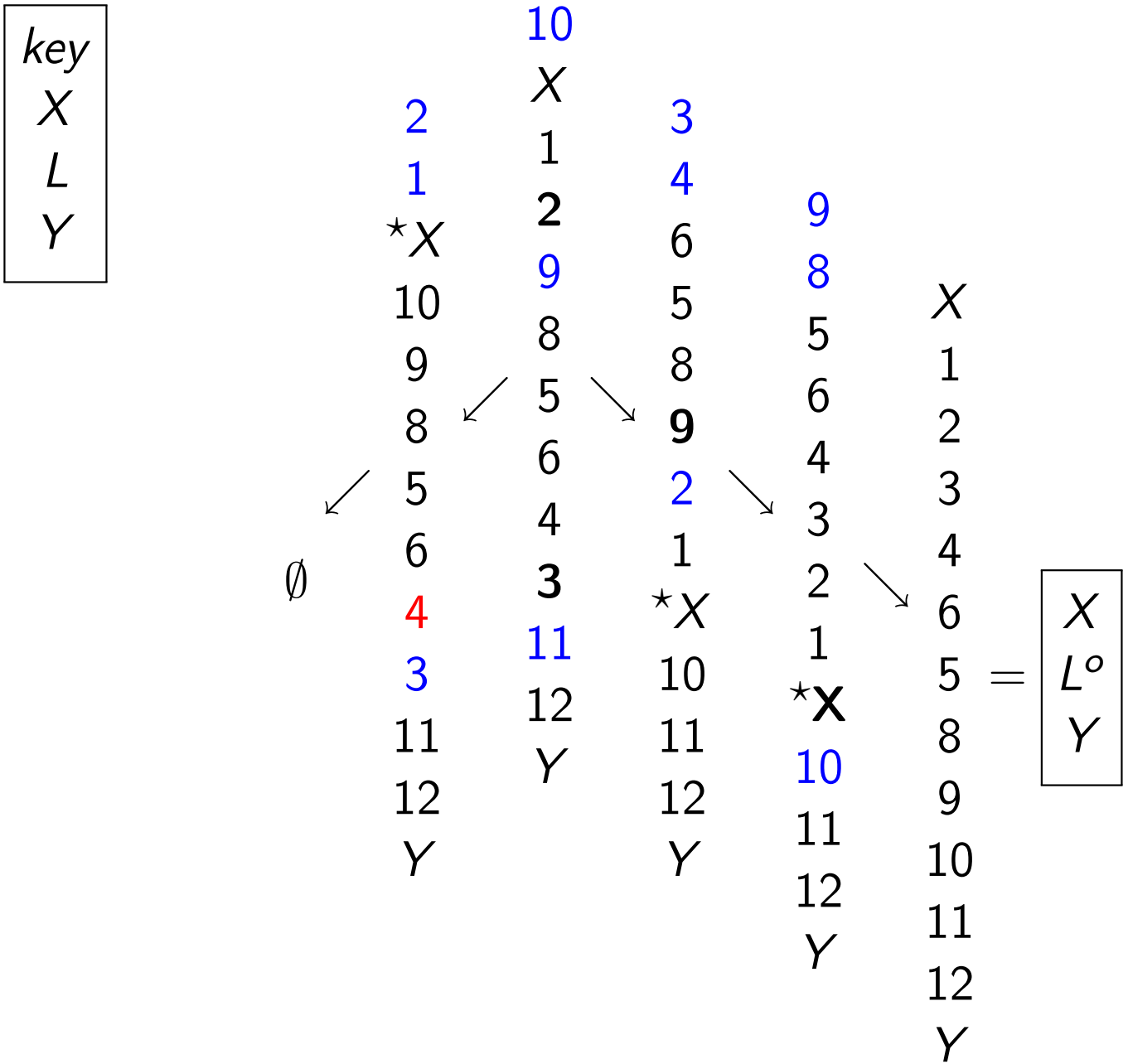
Lock gadget


key
X
L
Y
10
X
1
2
9
8
5
6
4
3
11
12
Y
2
1
⋆X
10
9
8
5
6
4
3
11
12
Y
3
4
6
5
8
9
2
1
⋆X
10
11
12
Y
∅

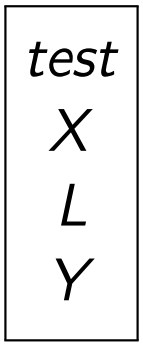
Lock gadget


key
X
L
Y
10
X
1
2
9
8
5
6
4
3
11
12
Y
2
1
⋆X
10
9
8
5
6
4
3
11
12
Y
∅
3
4
6
5
8
9
2
1
⋆X
10
11
12
Y
9
8
5
6
4
3
2
1
⋆X
10
11
12
Y

Lock gadget

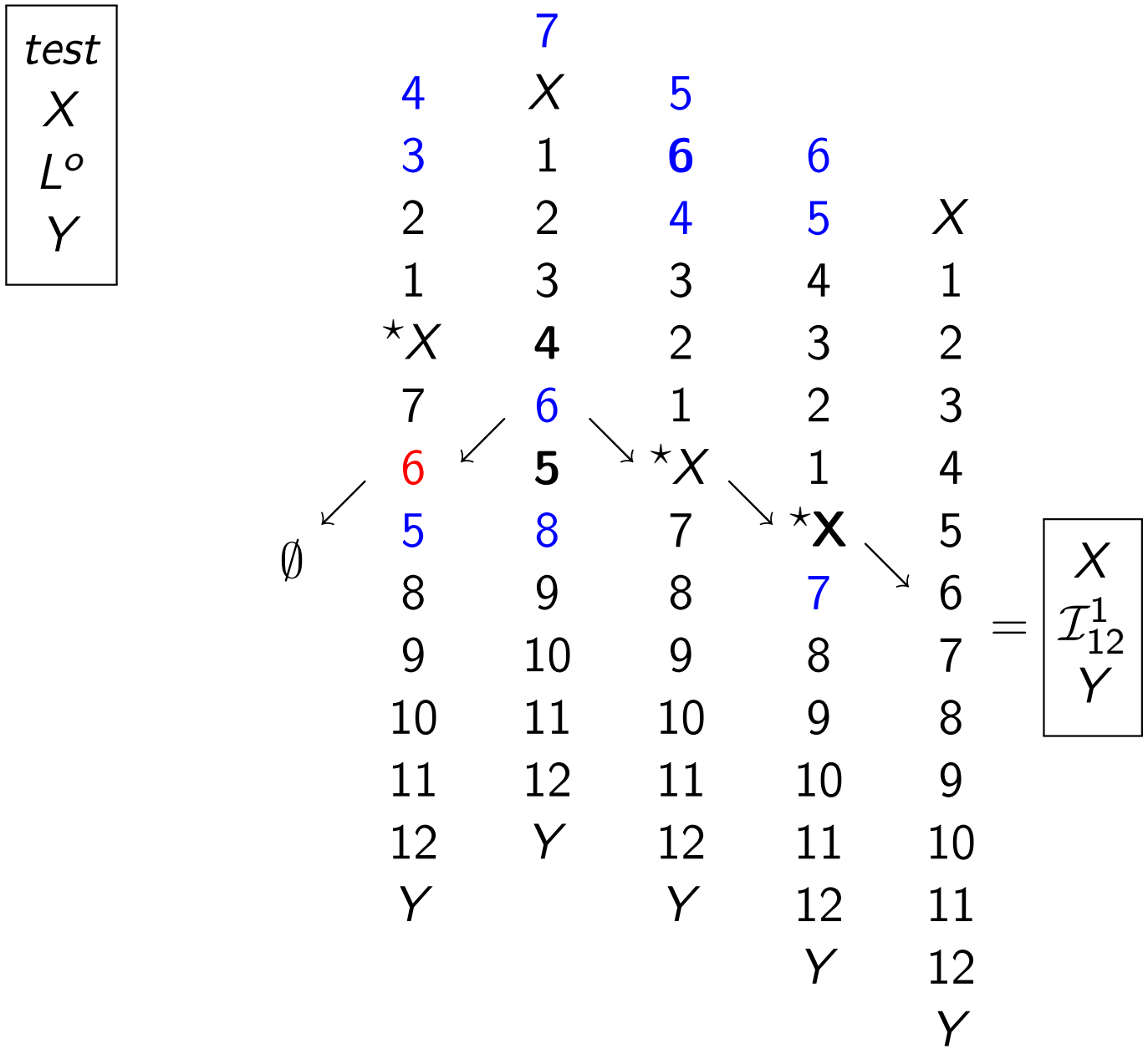

key
X
L
Y
2 1 ⋆X 10 9 8 5 6 4 3 11 12 Y
10 X 1 2 9 8 5 6 4 3 11 12 Y
3 4 6 5 8 9 2 1 ⋆X 10 11 12 Y
9 8 5 6 4 3 2 1 ⋆X 10 11 12 Y
X 1 2 3 4 6 5 8 9 10 11 12 Y
∅
=
X
L^o
Y

Lock gadget



7
X
1
2
9
8
5 $\longrightarrow \emptyset$
6
4
3
11
12
Y

Lock gadget


test
X
L°
Y
7
4 X 5
3 1 6 6
2 2 4 5 X
1 3 3 4 1
*X 4 2 3 2
7 6 1 2 3
6 5 *X 1 4
∅ 5 8 7 *X 5
8 9 8 7 6
9 10 9 8 7
10 11 10 9 8
11 12 11 10 9
12 Y 12 11 10
Y Y 12 11
Y 12
Y
=
X
$\mathcal{I}_{12}^{1}$
Y

Overall flow

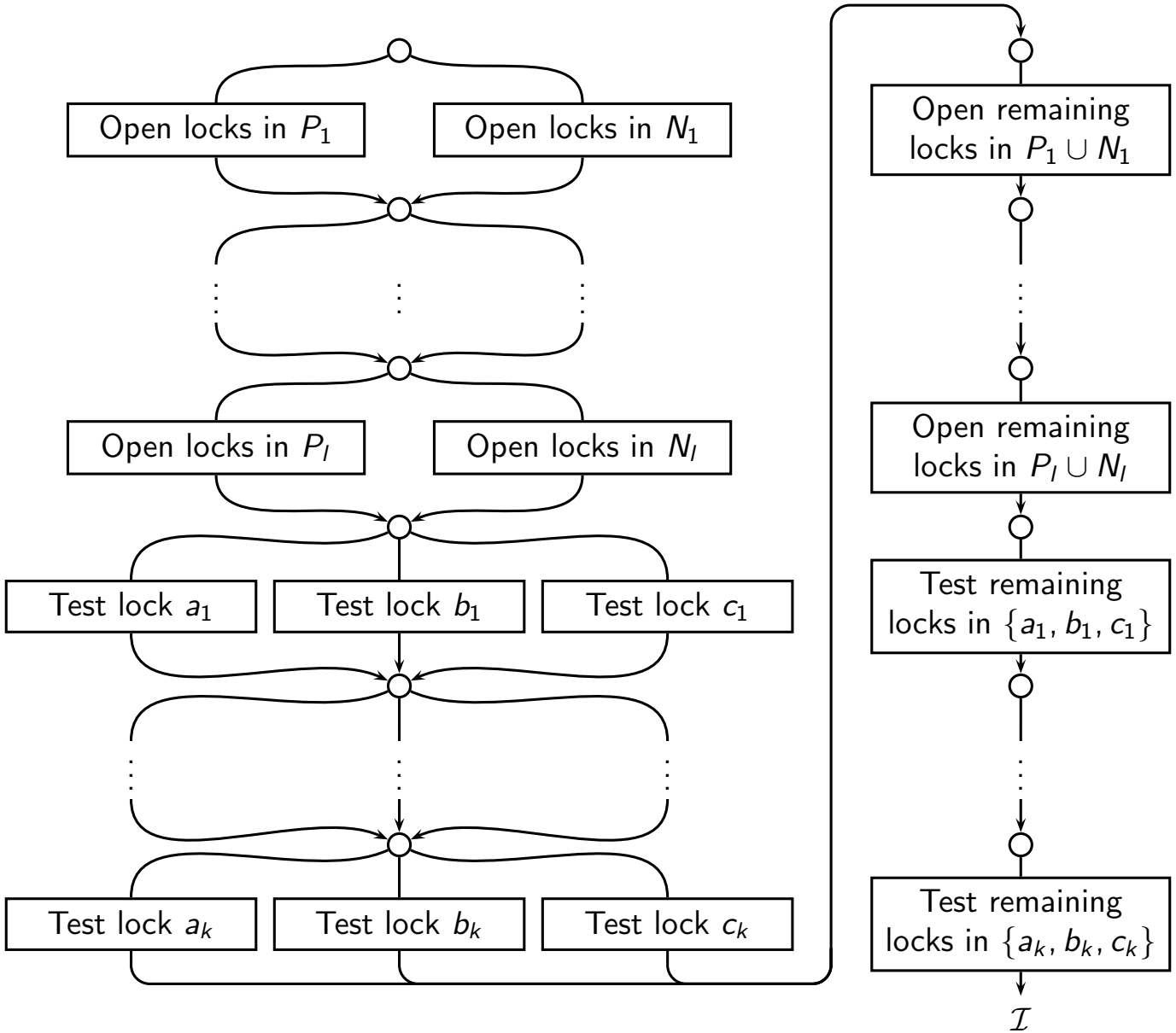
Literals

x_i: set P_i

$\neg x_i$: set N_i

Clause C_j

$a_j \vee b_j \vee c_j$



Fork gadget

$$E = \boxed{\begin{array}{c} 11 \\ 8 \\ 7 \\ 3 \end{array}}$$

$$F = \boxed{\begin{array}{c} 10 \\ 9 \\ 6 \\ 12 \\ 13 \\ 4 \\ 5 \\ 15 \\ 14 \\ 2 \\ 1 \end{array}}$$

Fork gadget

$$E = \begin{bmatrix} 11 \\ 8 \\ 7 \\ 3 \end{bmatrix}$$

$$F = \begin{bmatrix} 10 \\ 9 \\ 6 \\ 12 \\ 13 \\ 4 \\ 5 \\ 15 \\ 14 \\ 2 \\ 1 \end{bmatrix}$$

$$F^1 = \begin{bmatrix} 10 \\ 9 \\ 6 \\ 7 \\ 8 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \\ 5 \\ 4 \\ 3 \\ 2 \\ 1 \end{bmatrix}$$

$$F^2 = \begin{bmatrix} 3 \\ 7 \\ 8 \\ 11 \\ 10 \\ 9 \\ 6 \\ 12 \\ 13 \\ 4 \\ 5 \\ 15 \\ 14 \\ 2 \\ 1 \end{bmatrix}$$

Fork gadget

$$E = \begin{bmatrix} 11 \\ 8 \\ 7 \\ 3 \end{bmatrix}$$

$$F = \begin{bmatrix} 10 \\ 9 \\ 6 \\ 12 \\ 13 \\ 4 \\ 5 \\ 15 \\ 14 \\ 2 \\ 1 \end{bmatrix}$$

$$F^1 = \begin{pmatrix} \vdots \end{pmatrix} \quad F^2 = \begin{pmatrix} \vdots \end{pmatrix}$$

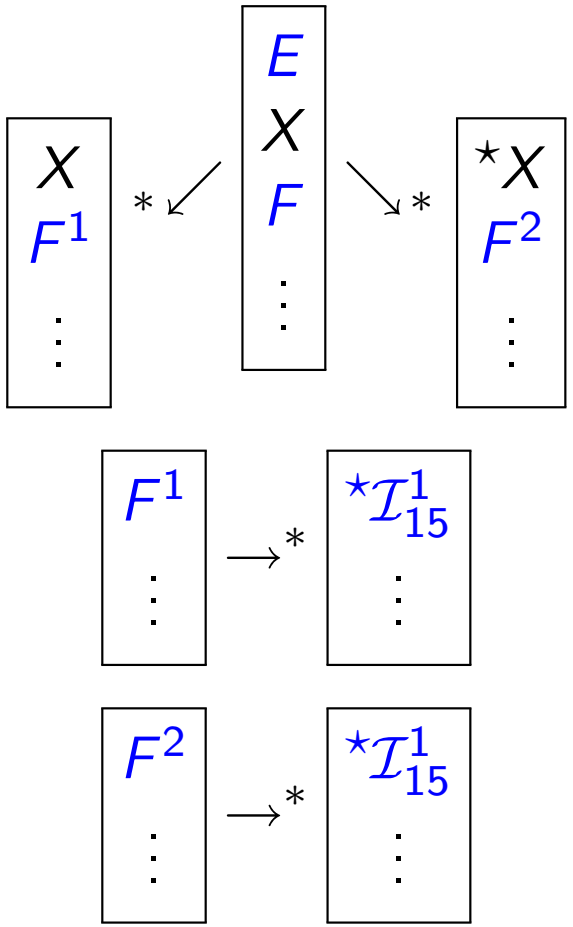
Fork gadget

$$E = \begin{bmatrix} 11 \\ 8 \\ 7 \\ 3 \end{bmatrix}$$

$$F = \begin{bmatrix} 10 \\ 9 \\ 6 \\ 12 \\ 13 \\ 4 \\ 5 \\ 15 \\ 14 \\ 2 \\ 1 \end{bmatrix}$$

$$F^1 = \begin{pmatrix} \vdots \end{pmatrix} \quad F^2 = \begin{pmatrix} \vdots \end{pmatrix}$$

Two efficient paths



Hook gadget

$$G = \boxed{\begin{matrix} 3 \\ 4 \end{matrix}}$$

$$H = \boxed{\begin{matrix} 12 \\ 11 \\ 6 \\ 5 \\ 9 \\ 8 \\ 2 \\ 1 \end{matrix}}$$

$$take = 10$$

$$put = 7$$

Hook gadget

$$G = \boxed{\begin{matrix}3\\4\end{matrix}} \qquad G' = \begin{pmatrix}\vdots\end{pmatrix}$$

$$H' = \begin{pmatrix}\vdots\end{pmatrix}$$

$$H = \boxed{\begin{matrix}12\\11\\6\\5\\9\\8\\2\\1\end{matrix}} \qquad G'' = \begin{pmatrix}\vdots\end{pmatrix}$$

$$H'' = \begin{pmatrix}\vdots\end{pmatrix}$$

$$take = 10$$

$$put = 7$$

Hook gadget

$$G = \begin{bmatrix} 3 \\ 4 \end{bmatrix} \qquad G' = \begin{pmatrix} \vdots \end{pmatrix}$$

$$H' = \begin{pmatrix} \vdots \end{pmatrix}$$

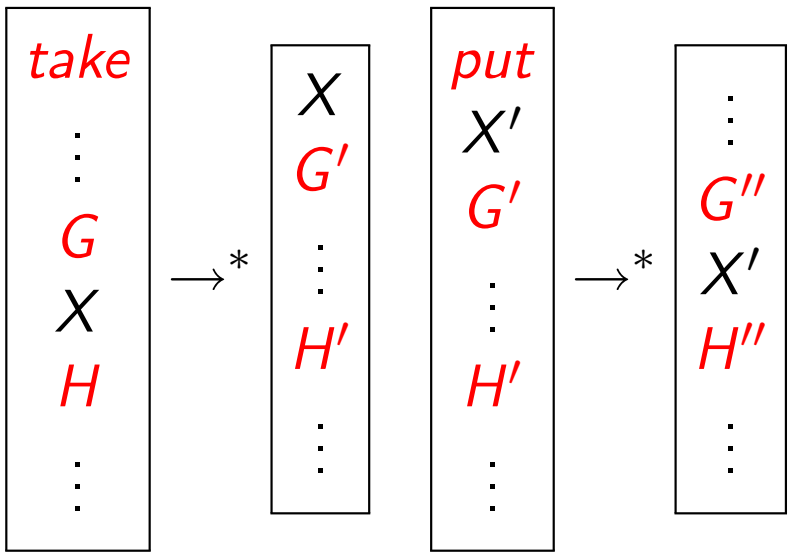
$$H = \begin{bmatrix} 12 \\ 11 \\ 6 \\ 5 \\ 9 \\ 8 \\ 2 \\ 1 \end{bmatrix} \qquad G'' = \begin{pmatrix} \vdots \end{pmatrix}$$

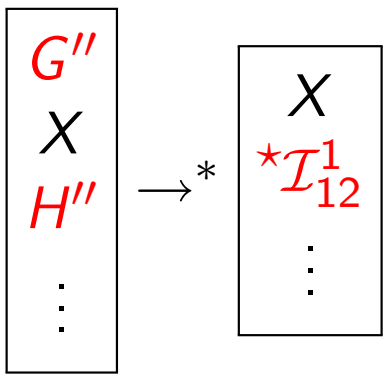
$$H'' = \begin{pmatrix} \vdots \end{pmatrix}$$

$take = 10$

$put = 7$

Moves a substring up and down





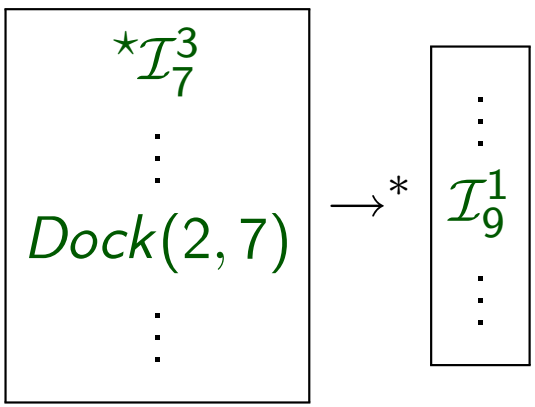
Dock gadget

$$Dock(2,7) = \boxed{\begin{matrix}1\\2\\8\\9\end{matrix}}$$

Dock gadget

$$Dock(2,7) = \boxed{\begin{array}{c}1\\2\\8\\9\end{array}}$$

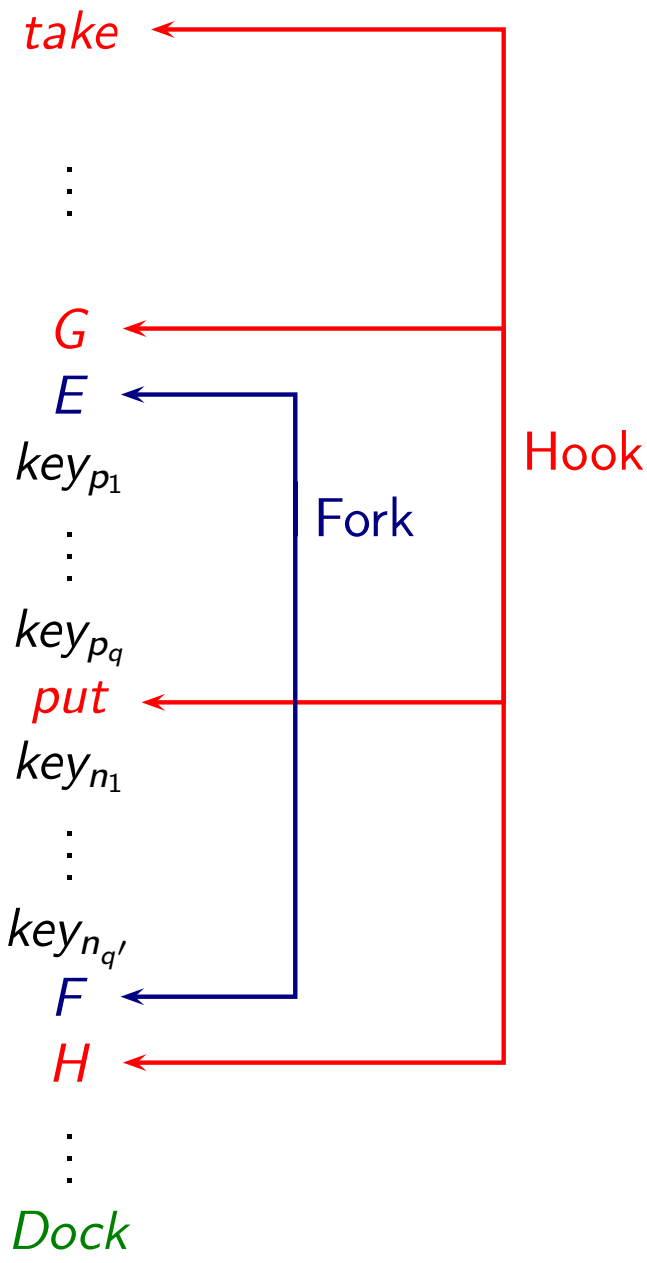
Stores a sorted substring ($^\star\mathcal{I}$) out of the head of the stack.



Variable gadget

Variable gadget

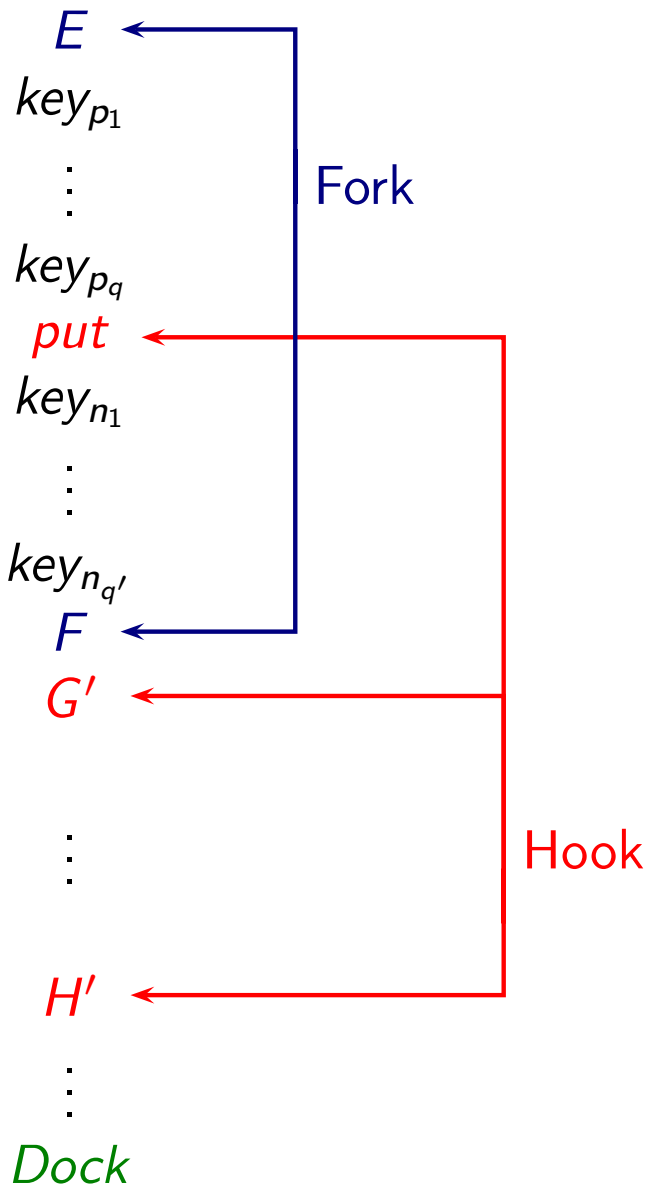
1 First part Move up the main sequence



Variable gadget

Variable gadget

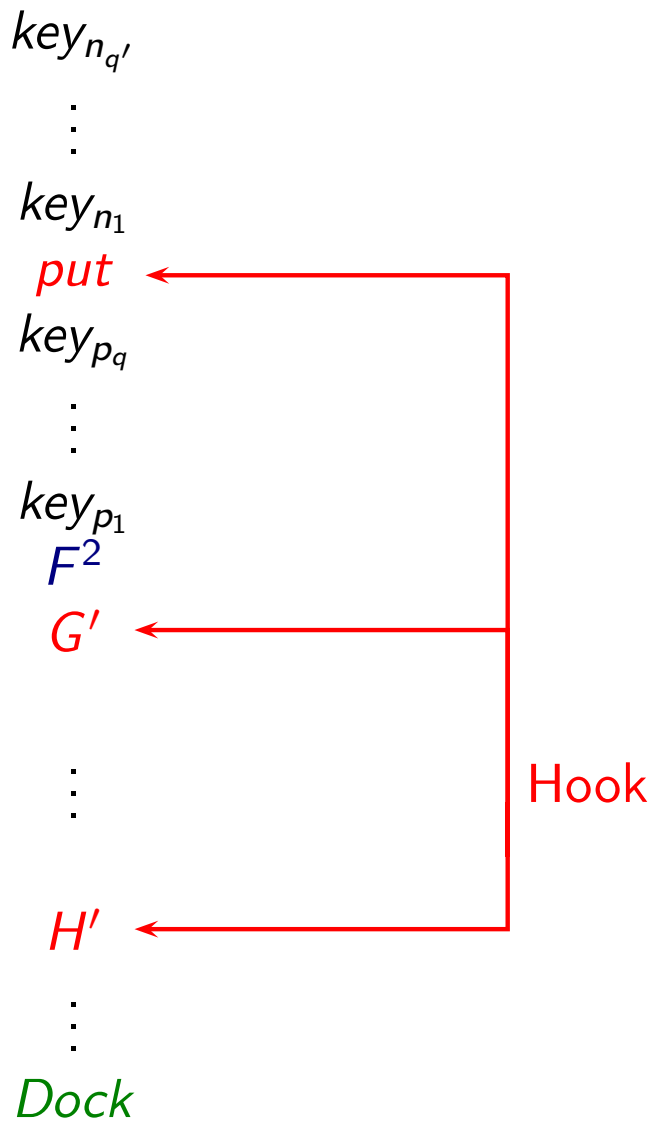
1. First part Move up the main sequence
2. Choose between x_i and $\neg x_i$



Variable gadget

Variable gadget

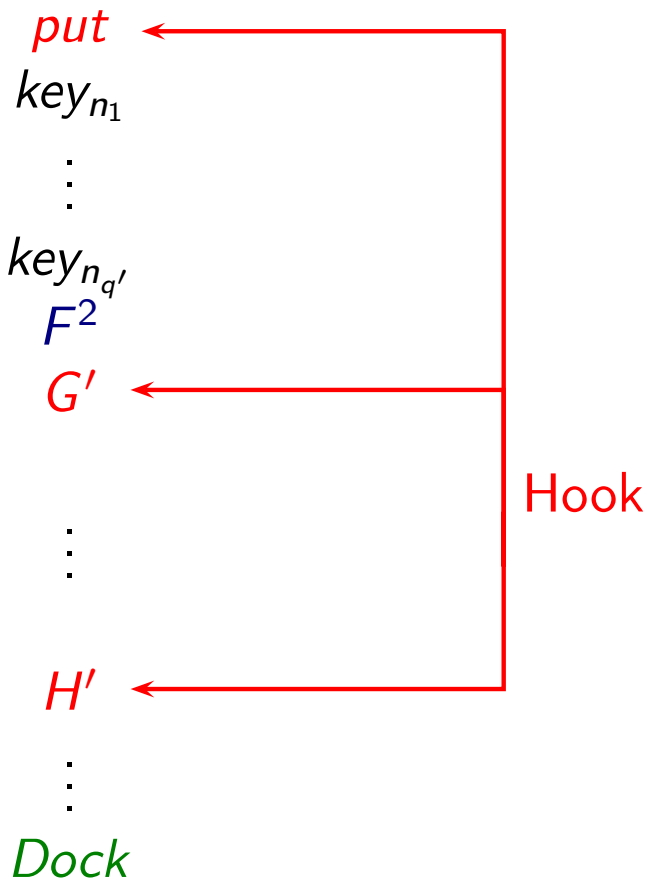
1. First part Move up the main sequence
2. Choose between x_i and $\neg x_i$
3. Open locks in N_i



Variable gadget

Variable gadget

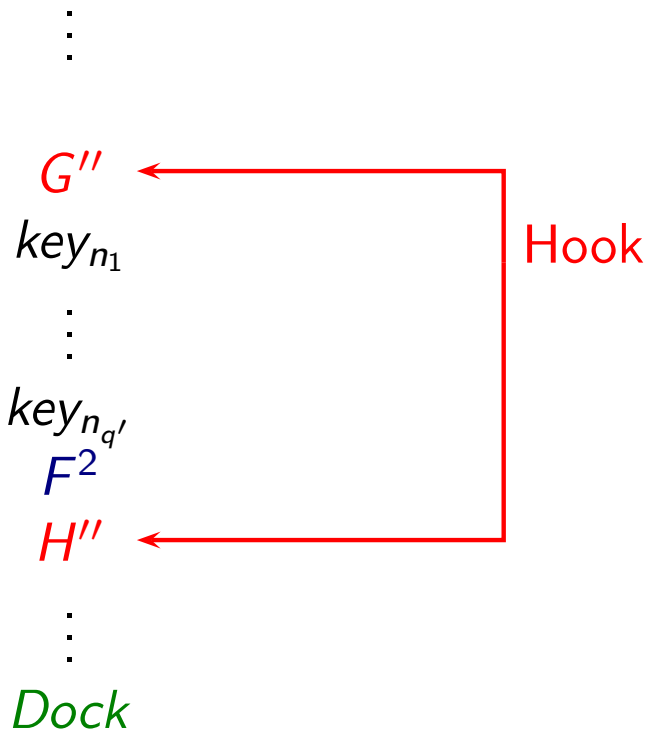
1. First part Move up the main sequence
2. Choose between x_i and $\neg x_i$
3. Open locks in N_i
4. Put back main sequence



Variable gadget

Variable gadget

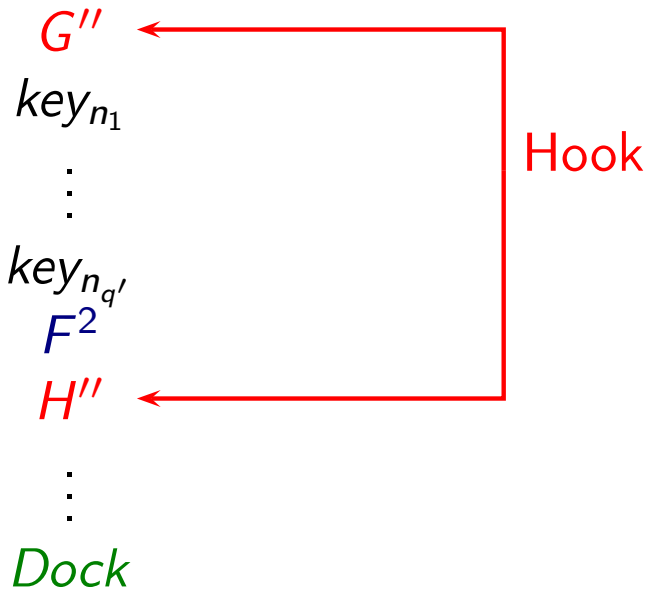
1. First part Move up the main sequence
2. Choose between x_i and $\neg x_i$
3. Open locks in N_i
4. Put back main sequence
5. Other gadgets are activated



Variable gadget

Variable gadget

1. First part Move up the main sequence
2. Choose between x_i and $\neg x_i$
3. Open locks in N_i
4. Put back main sequence
5. Other gadgets are activated
6. Second part Hook collapses



Variable gadget

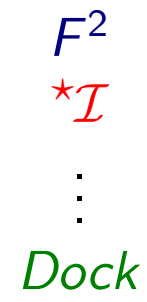
Variable gadget

1. First part Move up the main sequence
2. Choose between x_i and $\neg x_i$
3. Open locks in N_i
4. Put back main sequence
5. Other gadgets are activated
6. Second part Hook collapses
7. Open locks in P_i

Variable gadget

Variable gadget

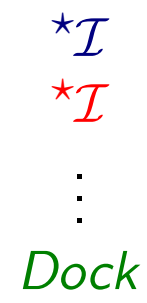
1. First part Move up the main sequence
2. Choose between x_i and $\neg x_i$
3. Open locks in N_i
4. Put back main sequence
5. Other gadgets are activated
6. Second part Hook collapses
7. Open locks in P_i
8. Fork collapses



Variable gadget

Variable gadget

1. First part Move up the main sequence
2. Choose between x_i and $\neg x_i$
3. Open locks in N_i
4. Put back main sequence
5. Other gadgets are activated
6. Second part Hook collapses
7. Open locks in P_i
8. Fork collapses
9. Dock stores sorted sequences



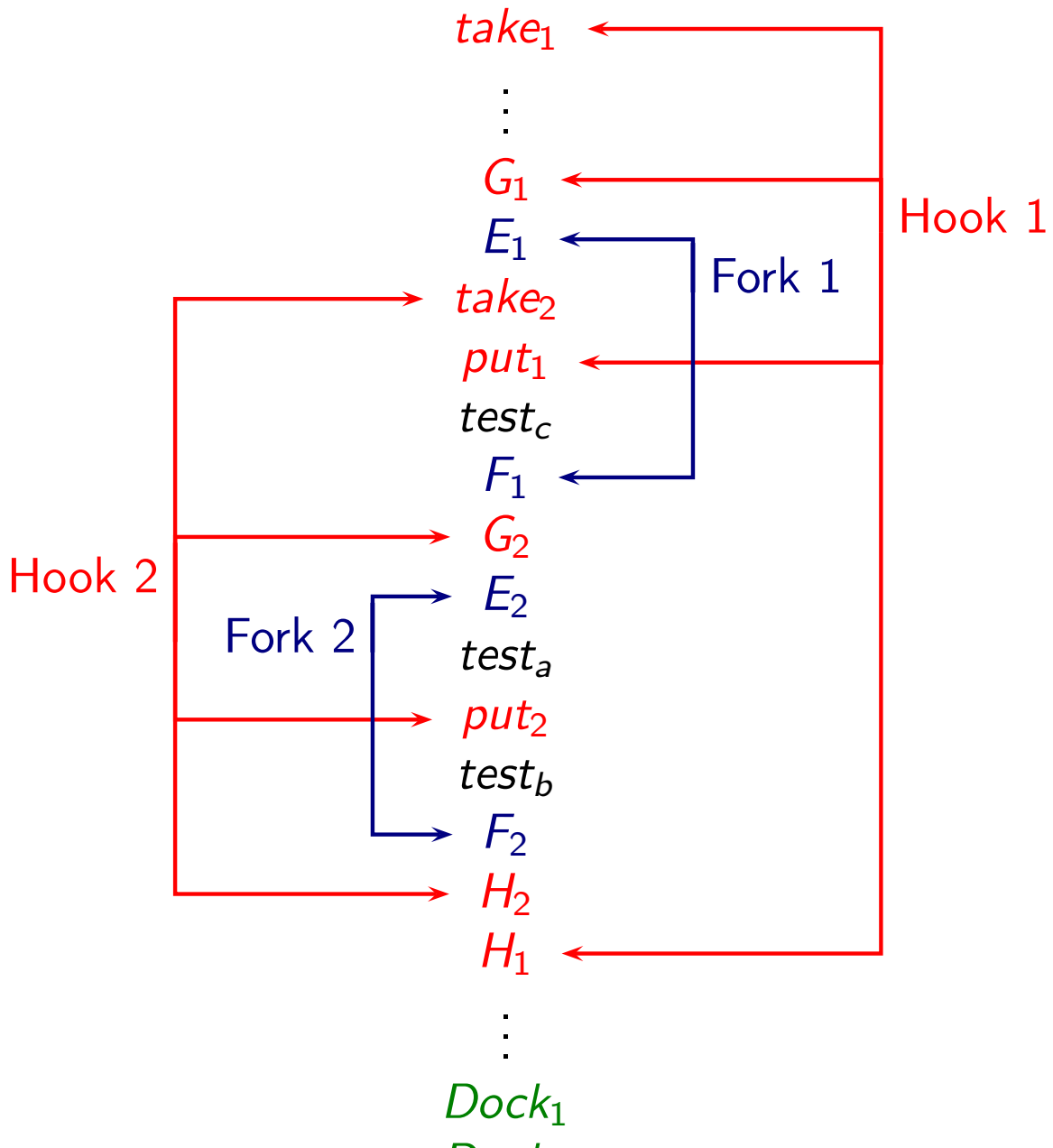
Variable gadget

Variable gadget

1. First part Move up the main sequence
2. Choose between x_i and $\neg x_i$
3. Open locks in N_i
4. Put back main sequence
5. Other gadgets are activated
6. Second part Hook collapses
7. Open locks in P_i
8. Fork collapses
9. Dock stores sorted sequences
10. End Gadget sorted

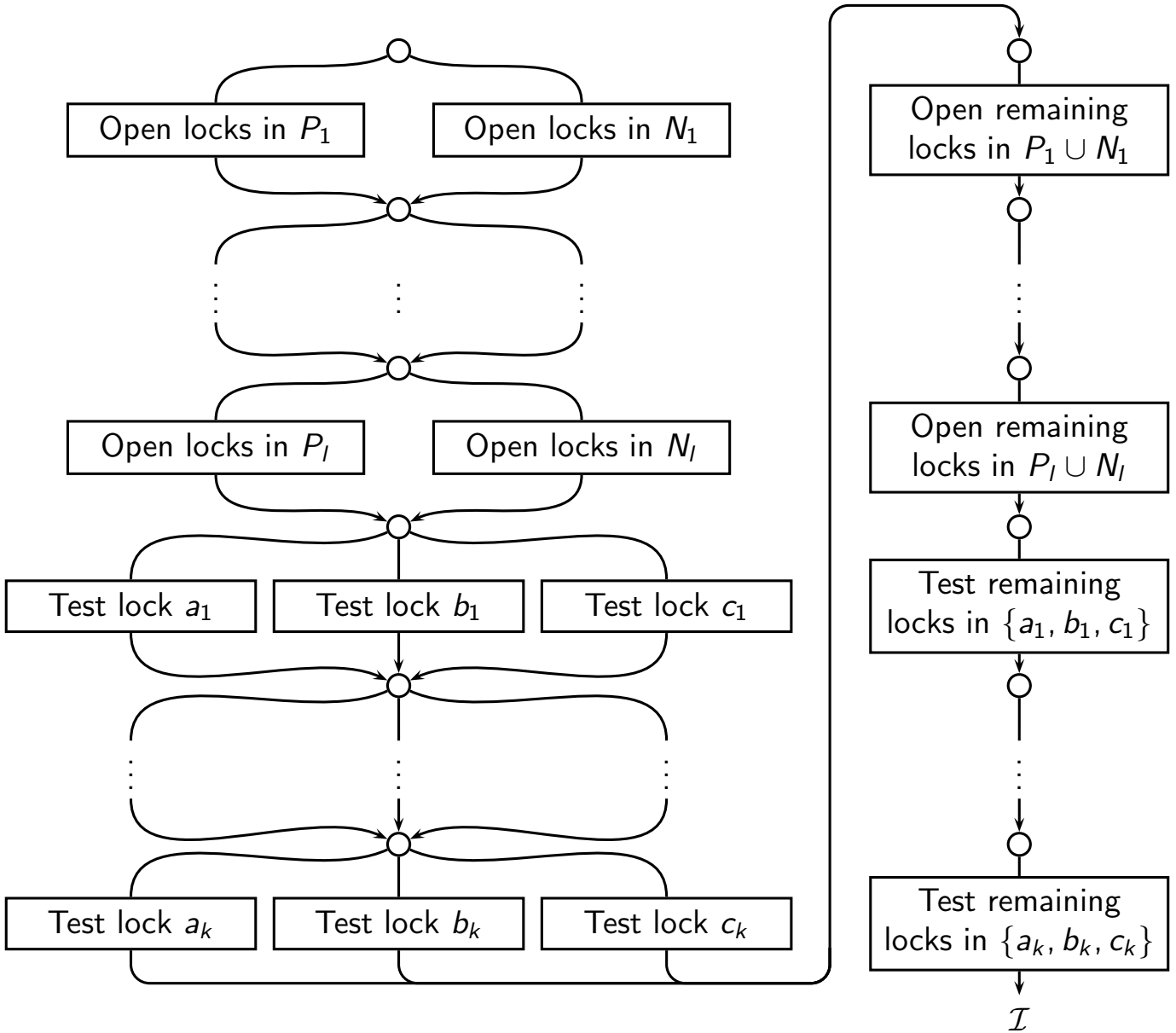
Clause gadget

Same structure,
with two
choices:
[[a or b] or c]



Overall construction

$$\pi_\phi = \begin{array}{c} take_{V_1} \\ \vdots \\ take_{V_l} \\ take_{C_1} \\ \vdots \\ take_{C_k} \\ V_1 \\ \vdots \\ V_l \\ C_1 \\ \vdots \\ C_k \\ (docks) \\ (locks) \end{array}$$



Finally

- There exists an efficient path from π_ϕ to the identity iff ϕ is satisfiable.
- The construction requires a polynomial time.
- **MIN-SBPR is NP-hard.**

Conclusion

- The complexity class of the Pancake Flipping problem is settled
- There remains many intriguing questions:
  - What about the burnt variant?
  - Any approximation algorithm?
  - Any FPT algorithm with a relevant parameter?
  - Any better bound for the diameter than $1.07n \leq f(n) \leq 1.64n$ (unburnt) and $1.5n \leq g(n) \leq 2n$ (burnt)?

Conclusion

- The complexity class of the Pancake Flipping problem is settled
- There remains many intriguing questions:
  - What about the burnt variant?
  - Any approximation algorithm?
  - Any FPT algorithm with a relevant parameter?
  - Any better bound for the diameter than $1.07n \leq f(n) \leq 1.64n$ (unburnt) and $1.5n \leq g(n) \leq 2n$ (burnt)?

Thank you!