On the decision trees with symmetries

Artur Riazanov

SPbAU, PDMI

CSR, 2018

Definitions

Definition

φ is a *propositional formula in conjunctive normal form (CNF)* if

$$\varphi = \bigwedge_{i=1}^{m} \left(\bigvee_{j=1}^{\ell_i} z_{ij} \right)$$

where z_{ij} are literals.

Definitions

Definition

φ is a *propositional formula in conjunctive normal form (CNF)* if

$$\varphi = \bigwedge_{i=1}^{m} \left(\bigvee_{j=1}^{\ell_i} z_{ij} \right)$$

where z_{ij} are literals.

A formula φ with variables $x_1, \ldots, x_n$ is satisfiable if there exist $\alpha_1, \ldots, \alpha_n \in \{0, 1\}$, such that the assignment $x_i := \alpha_i$ makes φ true.

Definitions

Definition

φ is a *propositional formula in conjunctive normal form (CNF)* if

$$\varphi = \bigwedge_{i=1}^{m} \left(\bigvee_{j=1}^{\ell_i} z_{ij} \right)$$

where z_{ij} are literals.

A formula φ with variables $x_1, \ldots, x_n$ is satisfiable if there exist $\alpha_1, \ldots, \alpha_n \in \{0, 1\}$, such that the assignment $x_i := \alpha_i$ makes φ true. The proof of satisfiability is a proper assignment.
It is much harder to prove that a formula is unsatisfiable.

How to prove unsatisfiability?

The resolution rule:

$$\frac{x \vee A \qquad \neg x \vee B}{A \vee B}$$

How to prove unsatisfiability?

The resolution rule:

$$\frac{x \vee A \qquad \neg x \vee B}{A \vee B}$$

A resolution refutation of a CNF formula φ is a derivation of the empty clause $\Box$ from the clauses of φ by the resolution rule.

How to prove unsatisfiability?

The resolution rule:

$$\frac{x \lor A \qquad \neg x \lor B}{A \lor B}$$

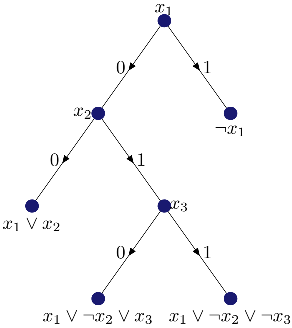
A resolution refutation of a CNF formula φ is a derivation of the empty clause $\square$ from the clauses of φ by the resolution rule.

A decision tree for a CNF formula φ is a protocol of backtracking search of a falsified clause.

Fact

The resolution polynomially simulates decision trees.

$$\begin{aligned} &\neg x_1 \\ &\land (x_1 \lor x_2) \\ &\land (x_1 \lor \neg x_2 \lor x_3) \\ &\land (x_1 \lor \neg x_2 \lor \neg x_3) \end{aligned}$$



Symmetries in formal proof systems

- In informal proofs we use symmetrical reasoning every time we say “without losing the generality” or simply “analougeously”.

Symmetries in formal proof systems

- In informal proofs we use symmetrical reasoning every time we say "without losing the generality" or simply "analougeously".
- Krishnamurthy suggested using of this construction for Resolution. The Resolution with the symmetry rule is $\mathrm{SR\text{-}I}$.

Symmetries in formal proof systems

- In informal proofs we use symmetrical reasoning every time we say “without losing the generality” or simply “analougeously”.
- Krishnamurthy suggested using of this construction for Resolution. The Resolution with the symmetry rule is SR-I.
- There is a short SR-I refutation for the pigeonhole principle (Urquhart, 1999) and for the clique-coloring tautology (Arai, 2000).

Symmetries in formal proof systems

- In informal proofs we use symmetrical reasoning every time we say "without losing the generality" or simply "analougeously".
- Krishnamurthy suggested using of this construction for Resolution. The Resolution with the symmetry rule is SR-I.
- There is a short SR-I refutation for the pigeonhole principle (Urquhart, 1999) and for the clique-coloring tautology (Arai, 2000).
- We are going to consider decision trees equipped with symmetry-based pruning.

Symmetries

Definition

Let φ be a CNF formula with variables from X. A bijection $\pi : X \to X$ is a symmetry of φ if $\pi(\varphi) = \varphi$ i.e. the renaiming π permutes clauses of the formula φ.

For example renaiming $\pi(x) = y$; $\pi(y) = x$ is a symmetry of a formula $x \wedge y$.

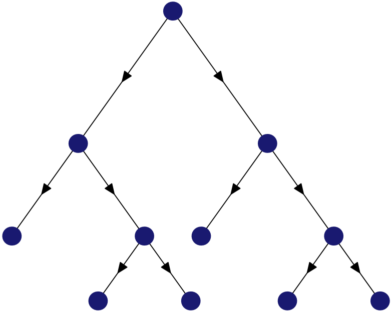
The proof system $\mathrm{SR\text{-}I}$ is defined as the resolution system with additional rule

$$\frac{A}{\pi(A)}$$

where π is a symmetry of the formula φ.

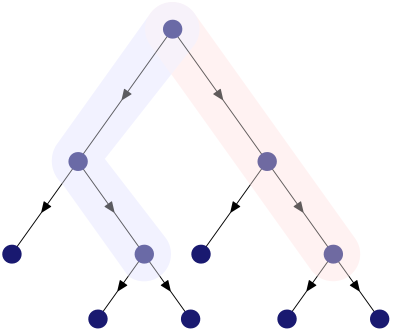
Decision trees with symmetries

Consider a decision tree for an unsatisfiable formula φ.



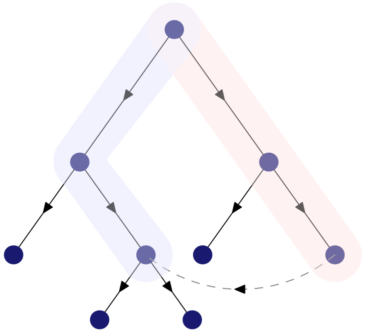
Symmetries in decision trees

Two paths are isomorphic to eachother, i.e. there exists a symmetry of φ that transforms the assignment associated with the first one into the assignment associated with the another one.



Symmetries in decision trees

Then we can prune the subtree of one of the vertices. SDT is a decision tree with removed symmetrical branches.



Proposition

$\mathrm{SR\text{-}I}$ polynomially simulates SDT.

PHP: Pigeonhole Principle

For $m > n$ we define a formula stating the pigeonhole principle: i.e. that there exists a way for m pigeons to fly into n holes such that no two pigeons fly into the same hole.

$$
\begin{array}{c|c|c|c|c|c|}
 & \multicolumn{5}{c}{n} \\
\cline{2-6}
 & P_{1,1} & P_{1,2} & P_{1,3} & P_{1,4} & P_{1,5} \\
\cline{2-6}
 & P_{2,1} & P_{2,2} & P_{2,3} & P_{2,4} & P_{2,5} \\
\cline{2-6}
 & P_{3,1} & P_{3,2} & P_{3,3} & P_{3,4} & P_{3,5} \\
\cline{2-6}
m & P_{4,1} & P_{4,2} & P_{4,3} & P_{4,4} & P_{4,5} \\
\cline{2-6}
 & P_{5,1} & P_{5,2} & P_{5,3} & P_{5,4} & P_{5,5} \\
\cline{2-6}
 & P_{6,1} & P_{6,2} & P_{6,3} & P_{6,4} & P_{6,5} \\
\cline{2-6}
\end{array}
$$

PHP: Pigeonhole Principle

A variable P_{ij} states wherther the i'th pigeon flies into the j'th hole or not.

$$\begin{array}{c|c|c|c|c|c|}
 & \multicolumn{5}{c}{n} \\
\hline
 & P_{1,1} & P_{1,2} & P_{1,3} & P_{1,4} & P_{1,5} \\
\hline
 & P_{2,1} & P_{2,2} & P_{2,3} & P_{2,4} & P_{2,5} \\
\hline
 & P_{3,1} & P_{3,2} & P_{3,3} & P_{3,4} & P_{3,5} \\
\hline
m & P_{4,1} & P_{4,2} & P_{4,3} & P_{4,4} & P_{4,5} \\
\hline
 & P_{5,1} & P_{5,2} & P_{5,3} & P_{5,4} & P_{5,5} \\
\hline
 & P_{6,1} & P_{6,2} & P_{6,3} & P_{6,4} & P_{6,5} \\
\hline
\end{array}$$

PHP: Pigeonhole Principle

$$\mathrm{PHP}_n^m = \bigwedge_{i=1}^{m} \left(\bigvee_{j=1}^{n} P_{ij} \right) \wedge \bigwedge_{\substack{k \in [m] \\ i,j \in [n] \\ i \neq j}} (\neg P_{ki} \vee \neg P_{kj})$$

n

m

$P_{1,1}$	$P_{1,2}$	$P_{1,3}$	$P_{1,4}$	$P_{1,5}$
$P_{2,1}$	$P_{2,2}$	$P_{2,3}$	$P_{2,4}$	$P_{2,5}$
$P_{3,1}$	$P_{3,2}$	$P_{3,3}$	$P_{3,4}$	$P_{3,5}$
$P_{4,1}$	$P_{4,2}$	$P_{4,3}$	$P_{4,4}$	$P_{4,5}$
$P_{5,1}$	$P_{5,2}$	$P_{5,3}$	$P_{5,4}$	$P_{5,5}$
$P_{6,1}$	$P_{6,2}$	$P_{6,3}$	$P_{6,4}$	$P_{6,5}$

PHP: Pigeonhole Principle

$$\mathrm{PHP}_n^m = \bigwedge_{i=1}^{m} \left(\bigvee_{j=1}^{n} P_{ij} \right) \wedge \bigwedge_{k \in [n]} \bigwedge_{\substack{i,j \in [m] \\ i \neq j}} (\neg P_{ik} \vee \neg P_{jk})$$

$$\begin{array}{cc} & n \\ m & \begin{array}{|c|c|c|c|c|} \hline P_{1,1} & P_{1,2} & P_{1,3} & P_{1,4} & P_{1,5} \\ \hline P_{2,1} & P_{2,2} & P_{2,3} & P_{2,4} & P_{2,5} \\ \hline P_{3,1} & P_{3,2} & P_{3,3} & P_{3,4} & P_{3,5} \\ \hline P_{4,1} & P_{4,2} & P_{4,3} & P_{4,4} & P_{4,5} \\ \hline P_{5,1} & P_{5,2} & P_{5,3} & P_{5,4} & P_{5,5} \\ \hline P_{6,1} & P_{6,2} & P_{6,3} & P_{6,4} & P_{6,5} \\ \hline \end{array} \end{array}$$

FPHP: Functional Pigeonhole Principle

$$\mathrm{FPHP}_n^m = \mathrm{PHP}_n^m \wedge \bigwedge_{k \in [m]} \bigwedge_{\substack{i,j \in [n] \\ i \neq j}} (\neg P_{ki} \vee \neg P_{kj})$$

$$\begin{array}{c|ccccc|}
 & & & n & & \\
\hline
 & \boxed{P_{1,1}} & \boxed{P_{1,2}} & \boxed{P_{1,3}} & \boxed{P_{1,4}} & \boxed{P_{1,5}} \\
 & P_{2,1} & P_{2,2} & P_{2,3} & P_{2,4} & P_{2,5} \\
 & P_{3,1} & P_{3,2} & P_{3,3} & P_{3,4} & P_{3,5} \\
m & P_{4,1} & P_{4,2} & P_{4,3} & P_{4,4} & P_{4,5} \\
 & P_{5,1} & P_{5,2} & P_{5,3} & P_{5,4} & P_{5,5} \\
 & P_{6,1} & P_{6,2} & P_{6,3} & P_{6,4} & P_{6,5} \\
\hline
\end{array}$$

CLIQUE-COLORING

The formula $\mathrm{CLIQUE\text{-}COLORING}_{n,k}(x, y, z)$ states that a graph defined by the adjascency matrix z, contains a clique of size k, defined by x and has a $(k-1)$-coloring defined by y.
There are exponential lower bounds for the sizes of refutations of all encodings of CLIQUE-COLORING in the Resolution and the Cutting Planes proof systems (Pudlak, 1997).

Theorem (Urquhart, 1999)

There is a SR-I *refutation of the standard encoding of* $\mathrm{CLIQUE\text{-}COLORING}_{n,k}$ *of size* $\mathrm{poly}(n, k)$.

CLIQUE-COLORING

The formula $\mathrm{CLIQUE\text{-}COLORING}_{n,k}(x, y, z)$ states that a graph defined by the adjascency matrix z, contains a clique of size k, defined by x and has a $(k-1)$-coloring defined by y.
There are exponential lower bounds for the sizes of refutations of all encodings of $\mathrm{CLIQUE\text{-}COLORING}$ in the Resolution and the Cutting Planes proof systems (Pudlak, 1997).

Theorem (Urquhart, 1999)

There is a $\mathrm{SR\text{-}I}$ *refutation of the standard encoding of* $\mathrm{CLIQUE\text{-}COLORING}_{n,k}$ *of size* $\mathrm{poly}(n, k)$.

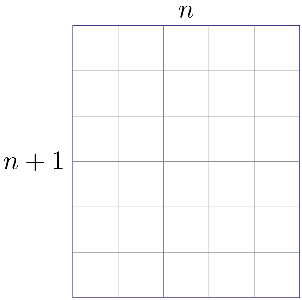
Theorem

There is an SDT for one of the encodings of $\mathrm{CLIQUE\text{-}COLORING}_{n,k}$ *of size* $\mathrm{poly}(n, k)$.

FPHP

Theorem

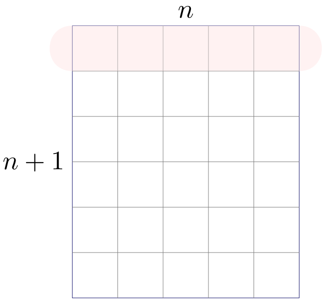
There exists an SDT for FPHP_n^{n+1} *of size* $O(n^3)$.



FPHP

Theorem

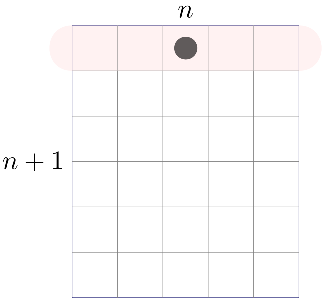
There exists an SDT for FPHP_n^{n+1} *of size* $O(n^3)$.



FPHP

Theorem

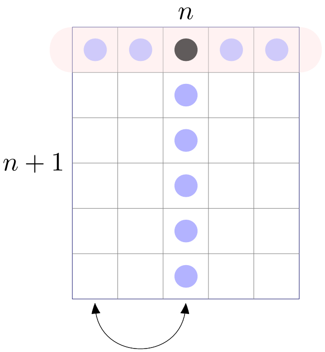
There exists an SDT for FPHP_n^{n+1} *of size* $O(n^3)$.



FPHP

Theorem

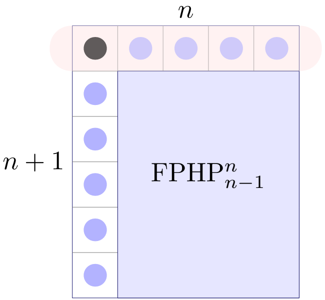
There exists an SDT for FPHP_n^{n+1} of size $O(n^3)$.



FPHP

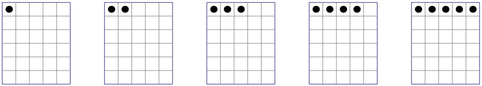
Theorem

There exists an SDT for FPHP_n^{n+1} *of size* $O(n^3)$*.*



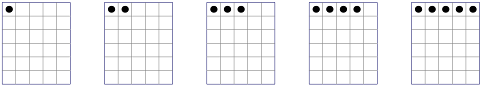
FPHP and PHP

For PHP_n^{n+1} there are multiple non-isomorphic assignments to the variables of the first row.



FPHP and PHP

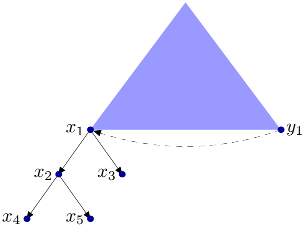
For PHP_n^{n+1} there are multiple non-isomorphic assignments to the variables of the first row.



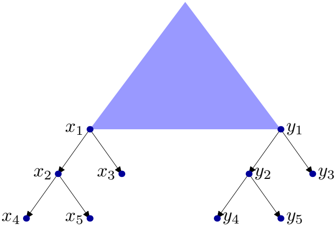
The approach that worked for FPHP_n^{n+1} yields an SDT of size $2^{O(\sqrt{n})}$.

Lower bound for the size of an SDT for PHP_n^{n+1}

Suppose there is only one symmetry pruning in an SDT.



SDT

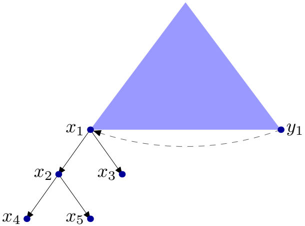


Decision tree

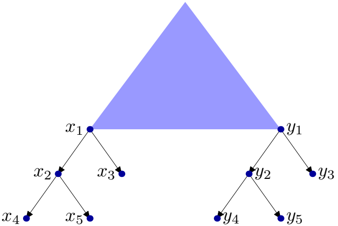
It is easy to see that $x_i \sim y_i$ (i.e. the assignments corresponding to x_i and to y_i are isomorphic).

Lower bound for the size of an SDT for PHP_n^{n+1}

Suppose there is only one symmetry pruning in an SDT.



SDT



Decision tree

It is easy to see that $x_i \sim y_i$ (i.e. the assignments corresponding to x_i and to y_i are isomorphic).
The number of vertices in an SDT for φ is at least the number of equivalence classes on the set of vertices of a plain decision tree with respect to $\sim$.

Game interpretation

Let S be a set of non-falsifying partial assignments to PHP_n^{n+1}. Alice and Bob maintain an assignment α to variables of PHP_n^{n+1}. Initially it is empty. At each turn Alice chooses a variable x then Bob chooses a value $b \in \{0, 1\}$ and they assign $\alpha(x) := b$.

- Alice wins if α falsifies a clause of PHP_n^{n+1};
- Bob wins if $\alpha \in S$ at some moment of the game.

Game interpretation

Let S be a set of non-falsifying partial assignments to PHP_n^{n+1}. Alice and Bob maintain an assignment α to variables of PHP_n^{n+1}. Initially it is empty. At each turn Alice chooses a variable x then Bob chooses a value $b \in \{0,1\}$ and they assign $\alpha(x) := b$.

- Alice wins if α falsifies a clause of PHP_n^{n+1};
- Bob wins if $\alpha \in S$ at some moment of the game.

Lemma

Bob has a winning strategy in the game with set S iff every decision tree has a vertex with the assignment from S.

Game interpretation

Let S be a set of non-falsifying partial assignments to PHP_n^{n+1}. Alice and Bob maintain an assignment α to variables of PHP_n^{n+1}. Initially it is empty. At each turn Alice chooses a variable x then Bob chooses a value $b \in \{0,1\}$ and they assign $\alpha(x) := b$.

- Alice wins if α falsifies a clause of PHP_n^{n+1};
- Bob wins if $\alpha \in S$ at some moment of the game.

Lemma

Bob has a winning strategy in the game with set S iff every decision tree has a vertex with the assignment from S.

Proposition

Suppose there exists a family of sets of assignments to variables of PHP_n^{n+1} that do not falsify the formula, $S_1, \ldots, S_k$ such that

- two assignments from different sets are not isomorphic;
- Bob has a winning strategy for each of the sets $S_1, \ldots, S_k$.

Then every SDT for PHP_n^{n+1} contains at least k vertices.

Invariant

We denote the set of assignments to the variables of φ by $\mathcal{A}_\varphi$. A function $\mu : \mathcal{A}_\varphi \to \{a_1, \ldots, a_k\}$ is an invariant wrt symmetries of φ if for two assignments α and β, $\mu(\alpha) \neq \mu(\beta) \implies \alpha \not\sim \beta$.

Invariant

We denote the set of assignments to the variables of φ by $\mathcal{A}_\varphi$. A function $\mu : \mathcal{A}_\varphi \to \{a_1, \ldots, a_k\}$ is an invariant wrt symmetries of φ if for two assignments α and β, $\mu(\alpha) \neq \mu(\beta) \implies \alpha \not\sim \beta$. Let $S_1 := \mu^{-1}(a_1), \ldots, S_k := \mu^{-1}(a_k)$.

Invariant

We denote the set of assignments to the variables of φ by $\mathcal{A}_\varphi$. A function $\mu : \mathcal{A}_\varphi \to \{a_1, \ldots, a_k\}$ is an invariant wrt symmetries of φ if for two assignments α and β, $\mu(\alpha) \neq \mu(\beta) \implies \alpha \not\sim \beta$.
Let $S_1 := \mu^{-1}(a_1), \ldots, S_k := \mu^{-1}(a_k)$.
Let $\mu_0(\alpha) = \{(\#\{j\colon \alpha(P_{ij}) = 0\}, \#\{j\colon \alpha(P_{ij}) = 1)\colon i \in [n+1]\}$.

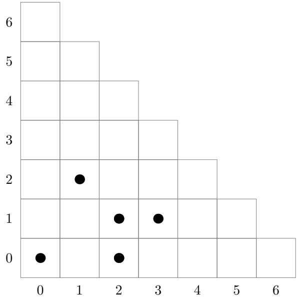
1		0	0			$(2,1)$
0	1			1		$(1,2)$
0	0	1		0		$(3,1)$
						$(0,0)$
0	0					$(2,0)$
0				0		$(2,0)$
0				0	1	$(2,1)$

$\{(0,0), (1,2), (2,0), (2,1), (3,1)\}$

Invariant

We denote the set of assignments to the variables of φ by $\mathcal{A}_\varphi$. A function $\mu : \mathcal{A}_\varphi \to \{a_1, \dots, a_k\}$ is an invariant wrt symmetries of φ if for two assignments α and β, $\mu(\alpha) \neq \mu(\beta) \implies \alpha \not\sim \beta$.
Let $S_1 := \mu^{-1}(a_1), \dots, S_k := \mu^{-1}(a_k)$.
Let $\mu_0(\alpha) = \{(\#\{j : \alpha(P_{ij}) = 0\}, \#\{j : \alpha(P_{ij}) = 1) : i \in [n+1]\}$.

1		0	0			$(2,1)$
0	1			1		$(1,2)$
0	0	1		0		$(3,1)$
						$(0,0)$
0	0					$(2,0)$
0				0		$(2,0)$
0				0	1	$(2,1)$

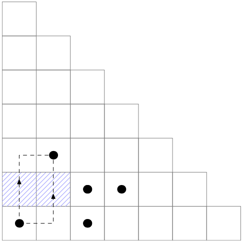


$\{(0,0), (1,2), (2,0), (2,1), (3,1)\}$

Invariant issues

Fact

Alice has a winning strategy for most of the pre-images of μ_0.

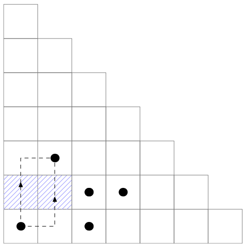


Alice can put a pebble into one of the hetched cells and then falsify the formula using the remaining pebbles.

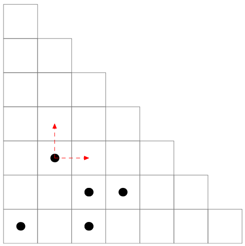
Invariant issues

Fact

Alice has a winning strategy for most of the pre-images of μ_0.



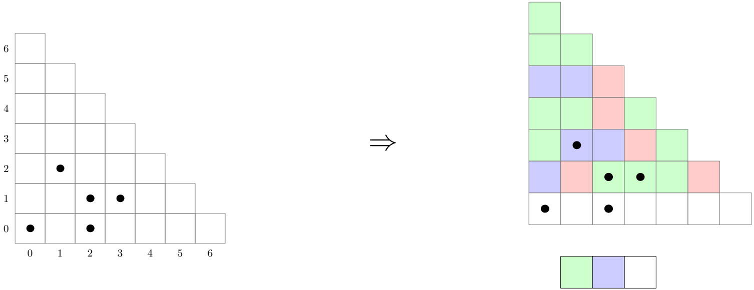
Alice can put a pebble into one of the hetched cells and then falsify the formula using the remaining pebbles.



Alice can make Bob move the top pebble and then Bob will be unable to obtain the needed picture.

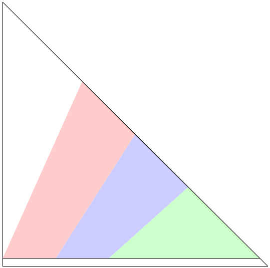
Robust invariant

Instead of the set of pebbles itself we use the set of colors under the pebbles as the invariant.



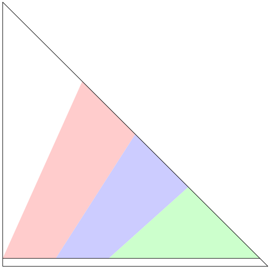
Plan

- We color the board in a clever way:



Plan

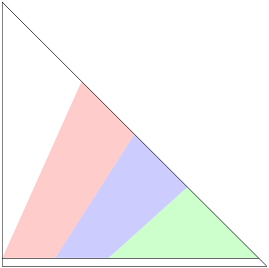
- We color the board in a clever way:

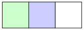


- Images of the invariant are subsets of colors containing white.

Plan

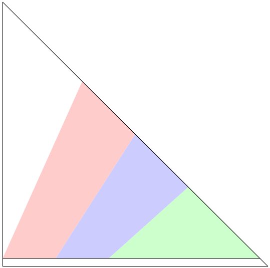
- We color the board in a clever way:

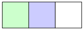


- Images of the invariant are subsets of colors containing white.
- For example 

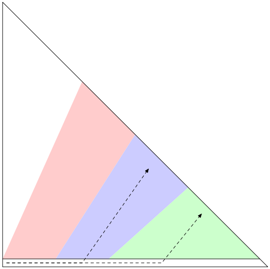
Plan

- We color the board in a clever way:



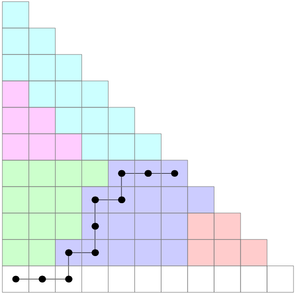
- Images of the invariant are subsets of colors containing white.
- For example 

- By default Bob moves all pebbles to the right on the white strip and, as soon as he can, moves pebbles to the invariant set colors.



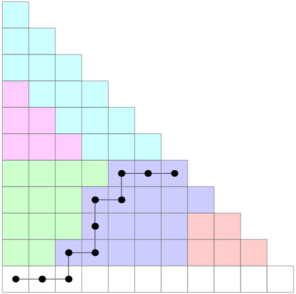
Robust invariant

When Alice chooses a variable P_{ij} Bob moves the i'th pebble one cell up ($b = 1$) or one cell right ($b = 0$).



Robust invariant

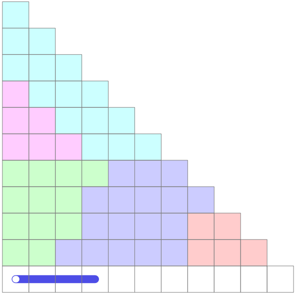
When Alice chooses a variable P_{ij} Bob moves the i'th pebble one cell up ($b = 1$) or one cell right ($b = 0$).



Bob must not change the color under a checker if it is not white.

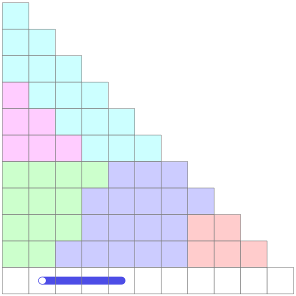
Robust invariant

But sometimes Bob have to forcefuly assign 0 (when the hole is already occupied by another pigeon). We ignore such moves and make the pebbles wide enough in order to be able to move the wide ones such that cells under the wide pebble have the same color throughout the game and move the real ones inside wide ones in case of forceful moves.



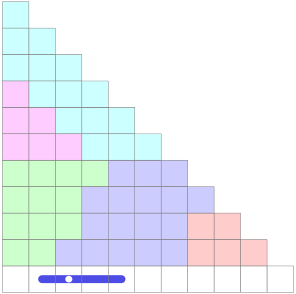
Robust invariant

But sometimes Bob have to forcefuly assign 0 (when the hole is already occupied by another pigeon). We ignore such moves and make the pebbles wide enough in order to be able to move the wide ones such that cells under the wide pebble have the same color throughout the game and move the real ones inside wide ones in case of forceful moves.



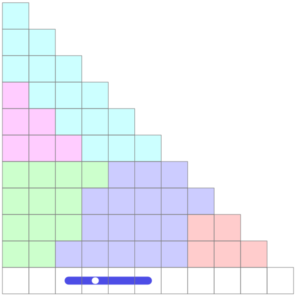
Robust invariant

But sometimes Bob have to forcefuly assign 0 (when the hole is already occupied by another pigeon). We ignore such moves and make the pebbles wide enough in order to be able to move the wide ones such that cells under the wide pebble have the same color throughout the game and move the real ones inside wide ones in case of forceful moves.



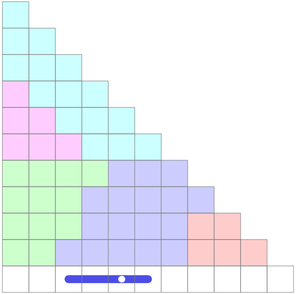
Robust invariant

But sometimes Bob have to forcefuly assign 0 (when the hole is already occupied by another pigeon). We ignore such moves and make the pebbles wide enough in order to be able to move the wide ones such that cells under the wide pebble have the same color throughout the game and move the real ones inside wide ones in case of forceful moves.



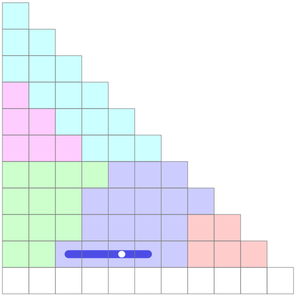
Robust invariant

But sometimes Bob have to forcefuly assign 0 (when the hole is already occupied by another pigeon). We ignore such moves and make the pebbles wide enough in order to be able to move the wide ones such that cells under the wide pebble have the same color throughout the game and move the real ones inside wide ones in case of forceful moves.



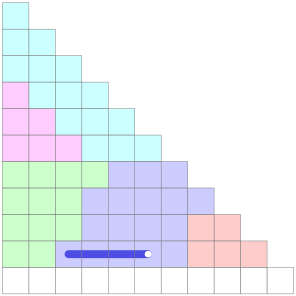
Robust invariant

But sometimes Bob have to forcefuly assign 0 (when the hole is already occupied by another pigeon). We ignore such moves and make the pebbles wide enough in order to be able to move the wide ones such that cells under the wide pebble have the same color throughout the game and move the real ones inside wide ones in case of forceful moves.



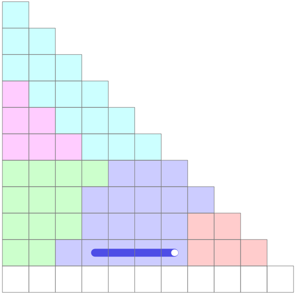
Robust invariant

But sometimes Bob have to forcefuly assign 0 (when the hole is already occupied by another pigeon). We ignore such moves and make the pebbles wide enough in order to be able to move the wide ones such that cells under the wide pebble have the same color throughout the game and move the real ones inside wide ones in case of forceful moves.



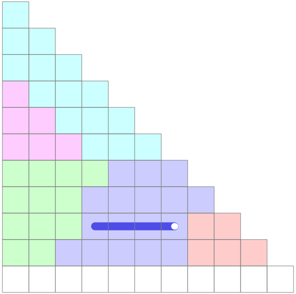
Robust invariant

But sometimes Bob have to forcefuly assign 0 (when the hole is already occupied by another pigeon). We ignore such moves and make the pebbles wide enough in order to be able to move the wide ones such that cells under the wide pebble have the same color throughout the game and move the real ones inside wide ones in case of forceful moves.



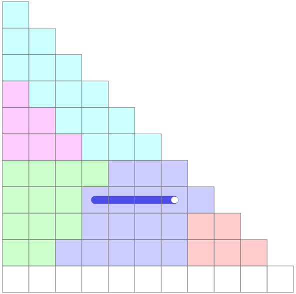
Robust invariant

But sometimes Bob have to forcefuly assign 0 (when the hole is already occupied by another pigeon). We ignore such moves and make the pebbles wide enough in order to be able to move the wide ones such that cells under the wide pebble have the same color throughout the game and move the real ones inside wide ones in case of forceful moves.



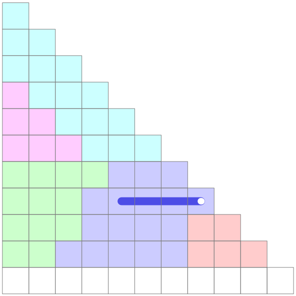
Robust invariant

But sometimes Bob have to forcefuly assign 0 (when the hole is already occupied by another pigeon). We ignore such moves and make the pebbles wide enough in order to be able to move the wide ones such that cells under the wide pebble have the same color throughout the game and move the real ones inside wide ones in case of forceful moves.



Robust invariant

But sometimes Bob have to forcefuly assign 0 (when the hole is already occupied by another pigeon). We ignore such moves and make the pebbles wide enough in order to be able to move the wide ones such that cells under the wide pebble have the same color throughout the game and move the real ones inside wide ones in case of forceful moves.



Comparison with CP, Decision Trees and the Resolution

	PHP	FPHP	CLIQUE-COLORING
RES	$2^{\Theta(n)}$	$2^{\Theta(n)}$	$2^{\Omega(n^{1/4})}$
CP	$\mathrm{poly}(n)$	$\mathrm{poly}(n)$	$2^{n^{\Omega(1)}}$
SR-I	$\mathrm{poly}(n)$	$\mathrm{poly}(n)$	$\mathrm{poly}(n)$
DT	$2^{\Theta(n \log n)}$	$2^{\Theta(n \log n)}$	$2^{\Omega(n)}$
SDT	$2^{\Omega(n^{1/3-o(1)})}$; $2^{\mathcal{O}(n^{1/2})}$	$\mathcal{O}(n^3)$	$\mathcal{O}(nk^2)$