Masaryk University
Faculty of Informatics

Algorithmic Analysis of Code Breaking Games

Master's Thesis

Miroslav Klimoš

Brno, 2014

Declaration

Advisor: prof. RNDr. Antonín Kučera, Ph.D.

Keywords

code braking games, mastermind, counterfeit coin, sat solving, greedy strategy

Contents

1 Introduction

2 Code Breaking Games

2.1 The Counterfeit Coin

There is a lot of variants of a logic puzzle with coins and a pair of scales balance. Here we present the most interesting ones and their generalization, which we study in the sequel.
In all the problems, you can use the scales only to weight coins. You can put as many coins at the sides as you like as long as the number is the same. All information you get is that both sides weight equally or which side is heavier (i.e., there are 3 possible results).
The weight of a *fake* coin is always different than the weight of a authentic one but it is not know whether it is heavier or ligther.

Problem 1 (The twelve coin problem). *You are given 12 coins, exactly one of which is fake. Determine the unique coin and its weight relavite to others. You can use the balance at most 3 times.*

Problem 2 (The thirteen coin problem). *You are given 13 coins, exactly one of which is fake. You have one more coin at your disposal which is guaranteed to be authentic. Determine the unique fake coin and its weight relavite to others.*

Problem 3 (General fake coin problem). *You are given n coins, f of them are fake (some of them may be lighter, some of them heavier). You have another m authentic coins at your disposal. In as less weightings as possible, determine which coins are fake.*

2.2 Mastermind

Mastermind is a classic 2-player board game invented by Mordecai Meirowitz in 1970[wiki]. The principle of the game is the same as of *Bulls and Cows*, it just uses colors instead of letters.

2.3 Black Box

2.4 Bags of Gold

2.5 Code 777

3 General model

3.1 Notation

Let Form_X be the set of all prepositional formulas over the set of variables X; V_X be the set of all valuations of variables X. Formulas $\varphi_0, \varphi_1 \in \mathrm{Form}_X$ are (semantically) equivalent, written $\varphi_0 \equiv \varphi_1$, if $v(\varphi_0) = v(\varphi_1)$ for all $v \in V_X$. For a formula $\varphi \in \mathrm{Form}_X$, let $\tau_X(\varphi) = |\{v \in V_X \mid v(\varphi) = 1\}|$ be the number of valuations by which φ is satisfied. We often omit the index X if it is clear from the context.
For any unary predicate P, $\#i \in A.P(i) = |\{i \in A \mid P(i)\}|$. We often omit the "$\in A$" part and write only $\#i.P(i)$ if the range of i is clear from the context.
Let Perm_X be the set of all permutations of X.

3.2 Formal definition

Motivace: Ve hře chci najít přiřazení proměnných. Hru tedy reprezentuju jako mnozinu promennych, a jakousi mnozinu experimentu. Po provedeni experimentu dostanu formuli v danych promennych, o ktere vim, ze je splnena v hledane valuaci. Tato formule reprezentuje castecnou informaci, kterou jsem experimentem ziskal. Kazdy experiment muze dopadnout vice zpusoby, pro kazdy mam tedy mnozinu moznych formuli, ktere mi da.
Pro kompaktni zapis reprezentuju experiment jako dvojici (typ experimentu, parametrizace), kde parametrizace je retezec nad danou abecedou. Pro kazdy typ experimentu pak mam mnozinu parametrizovanych formuli. This whole idea is formalized below.

Definition 4. A *code-breaking game* is a septuple $\mathcal{G} = (X, \varphi_0, T, \Sigma, E, F, \Phi)$, where

- X is a finite set of propositional variables,
- $\varphi_0 \in \mathrm{Form}_X$ is a satisfiable prepositional formula,
- T is a finite set of types of experiments,
- Σ is a finite alphabet,
- $E \subseteq T \times \Sigma^\star$ is an *experiment* relation, and
- F is a finite collection of functions of type $\Sigma \to X$,
- $\Phi : T \to 2^{\mathrm{PForm}_{X,F,\Sigma}}$ is an *outcome function* such that $\Phi(t)$ is finite for any $t \in T$. Definition of PForm follows (Definition 5).

Definition 5. A set of *parametrized formulas* $\mathrm{PForm}_{X,F,\Sigma}$ is a set of all strings ψ generated by the following grammar:

$$\psi ::= x \mid f(\$n) \mid \psi \circ \psi \mid \neg\psi,$$

where $x \in X$, $f \in F$, $n \in \mathbb{N}$, and $\circ \in \{\wedge, \vee, =>\}$. By $\psi(p)$ we denote application of a parametrization $p \in \Sigma^\star$ on a formula ψ, which is defined recursively on the stucture of ψ in the following way:

$$\begin{aligned}(x)(p) &= x,\\ (f(\$n))(p) &= f(p[n]),\\ (\psi_1 \circ \psi_2)(p) &= \psi_1(p) \circ \psi_2(p),\\ (\neg\psi)(p) &= \neg(\psi(p)).\end{aligned}$$

Znak \$ v $f(\$n)$ je od toho, aby to nematlo, ze n je argumentem funkce. Poznamka o tom, ze tohle me neomezuje oproti popsane intuici. V nejhorsim muzu pro kazdy experiment udelat zvlastni typ.
For the sake of simplicity, let us write $\Phi(e) = \{\psi(p) \mid \psi \in \Phi(t)\}$ for any experiment $e = (t,p) \in E$.
A code-breaking game is *well-formed* if for all $(t,p) \in E$,

$$\forall v \in V_X : v(\varphi_0) = 1 \Rightarrow \exists \text{ exactly one } \psi \in \Phi(t)\ .\ v(\psi(p)) = 1$$

This guarantees that the result of every experiment is uniquely defined for any valuation. Note that this property is not easy to check. In sequel, we suppose a game to be well-fordmed, if not stated otherwise.

Example 6 (Fake-coin problem). Fake-coin problem with n coins, one of which is fake, can be formalized as a code breaking game $\mathcal{F}_n = (X, \varphi_0, T, \Sigma, E, F, \Phi)$.

- $X = \{x_1, x_2, \dots, x_n, y\}$,
  $\varphi_0 = \text{Exactly-1}\ \{x_1, \dots, x_n\}$.
  Intuitively, variable x_i tells weather the coin i is fake. Variable y tells weather it is lighter or heavier. Formule φ_0 says that exactly one coin is fake.
- $T = \{w_2, w_4, \dots, w_n\}$,
  $\Sigma = \{1, 2, \dots, n\}$,
  $E = \bigcup_{1 \le m \le n/2} \{(w_{2m}, p) \mid p \in \{1, \dots, n\}^{2m}, \forall x \in X. \#_x(p) \le 1\}$.
  There are $n/2$ types of experiment – according to the number of coins we put on the weights. The alphabet contains natural numbers up to n and possible parametrizations for w_{2m} are strings of length $2m$ with no repetitions.

- $F = \{f_x\}$, where $f_x(i) = x_i$ for $1 \leq i \leq n$,
  $\Phi(w_m) =$
  $$\big\{((f_x(\$1) \vee \ldots \vee f_x(\$m)) \wedge \neg y) \vee ((f_x(\$m+1) \vee \ldots \vee f_x(\$2m)) \wedge y),$$
  $$((f_x(\$1) \vee \ldots \vee f_x(\$m)) \wedge y) \vee ((f_x(\$m+1) \vee \ldots \vee f_x(\$2m)) \wedge \neg y),$$
  $$\neg(f_x(\$1) \vee \ldots \vee f_x(\$2m))\big\}.$$

  There are 3 possible outcomes of every experiment. First, the right side is heavier. This happens if the fake coin is lighter and it appears in the first half of the parametrization, or if it is heavier and it appears in the second half. Second, analogously, the left side is heavier. Third, the weights are balanced if the fake coin do not participate in the experiment.

Example 7 (Fake-coin problem, alternative). For demonstration purposes, here is another formalization of the same problem. $\mathcal{F}'_n = (X, \varphi_0, T, \Sigma, E, F, \Phi)$.

- $X = \{x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n\}$,
  $\varphi_0 =$ Exactly-1 $\{x_1, \ldots, x_n, y_1, \ldots, y_n\}$.
  Variable x_i tells that the coin i is lighter, variable y_i tells that the coin i is heavier. Formule φ_0 says that exactly one coin is different.
- T, Σ, E is defined as in Example 6.
- $F = \{f_x, f_y\}$, where $f_x(i) = x_i$ and $f_y(i) = y_i$ for $1 \le i \le n$,

$$\begin{aligned}\Phi(w_m) = \big\{&(f_x(\$1) \lor \ldots \lor f_x(\$m)) \lor (f_y(\$m+1) \lor \ldots \lor f_y(\$2m)),\\ &(f_y(\$1) \lor \ldots \lor f_y(\$m)) \lor (f_x(\$m+1) \lor \ldots \lor f_x(\$2m)),\\ &\neg(f_x(\$1) \lor \ldots \lor f_x(\$2m) \lor f_y(\$1) \lor \ldots \lor f_y(\$2m))\big\}.\end{aligned}$$

Example 8 (Mastermind). Mastermind puzzle with n pegs and m colors can be formalized as a code breaking game $\mathcal{M}_{n,m} = (X, \varphi_0, T, \Sigma, E, F, \Phi)$.

- $X = \{x_{i,j} \mid 1 \le i \le n, 1 \le j \le m\}$,
  $\varphi_0 = \bigwedge\{\text{Exactly-1}\,\{x_{i,j} \mid 1 \le j \le m\} \mid 1 \le i \le n\}$.
  Variable $x_{i,j}$ tells whether there is the color j at position i. Formula φ_0 says that there is exactly one color at each position.
- $T = \{g_{k_1,\ldots,k_m} \mid k_i \in \{1, \ldots, n\}, \sum_i k_i = n\}$,
  $\Sigma = C$,
  $E = \{(g_{k_1,\ldots,k_m}, p) \mid p \in \Sigma^n, \#i.(p[i] = j) = k_j\}$.
  The type $g_{k_1,\ldots,k_m}$ covers all the guesses in which the number of j-colored pegs is k_j. Therefore, two guesses for which we use the same pegs (pegs are just shuffled) are of the same type, but if we change a peg for one with different color, it is other type of experiment.
- $F = \{f_1, \ldots, f_n\}$, where $f_i(c) = x_{i,c}$ for $1 \le i \le n$,

$$\begin{aligned}\Phi(g_{k_1,\ldots,k_n}) = \Big\{&\\ &\text{Exactly-b}\,\{f_i(\$i) \mid 1 \le i \le n\} \land\\ &\text{Exactly-t}\,\bigcup\big\{\{\text{AtLeast-l}\,(x_{1,j}, \ldots, x_{n,j}) \mid 1 \le l \le k_j\} \mid 1 \le j \le m\big\}\\ &\qquad\qquad \mid 0 \le b \le t, 0 \le t \le n\Big\}.\end{aligned}$$

Zdůvodnit proč to funguje.

Example 9 (Mastermind (alternative)). $\mathcal{M}'_{n,m} = (X, \varphi_0, T, \Sigma, E, F, \Phi)$.

- X and φ_0 is defined as in Example 8.

- $T = \{g\}$,
  $\Sigma = C$,
  $E = \{(g, p) \mid p \in \Sigma^n\}$.

- $F = \{f_1, \dots, f_n\}$, where $f_i(c) = x_{i,c}$ for $1 \leq i \leq n$, $\Phi(g) = \{\text{Outcome}(b, w) \mid 0 \leq b \leq n, 0 \leq w \leq n, b + w \leq n\}$, where Outcome function is computed by the algorithm described below.

Consider a fixed color combination (code) and a guess. We assign a symbol $\bullet$, $\times$ or a number to each position in the following way. If the color at a position is same in the code and the guess, we assign $\bullet$ to this position and the player gets a black peg for it. If the color in the guess at position i differs but it appears appears at position j in the code, we assign j to position i and the player gets a white peg. Also, position j must not be assigned $\bullet$ and the number j must not be assigned to any other position. We assign $\times$ to all other positions. For example, if the code is 1234 and the guess is 5251, the assignment is $\underline{\times \bullet \times 1}$.
We start the computation of $\text{Outcome}(b, w)$ with generation of all combinations of $\bullet$, $\times$ and different numbers from 1 to n, so that there is b-times $\bullet$, w-times a number and no number refers to a position with $\bullet$.
Next, for each combination, we generate a conjunction in the following way:

- For $\bullet$ at position i, we add $f_i(\$i)$.
- For a number j at position i, we add $\neg f_i(\$i) \land f_j(\$i)$.
- For $\times$ at position i, we add $\neg f_j(\$i)$ for any other position j with $\times$.

The result is a disjunction of all these clauses, which effectively enumerates all the cases.
To get a better idea about the results, this is $\text{Outcome}(1, 1)$ for n = 4:

$(\neg f_0(\$0) \land \neg f_1(\$1) \land \neg f_1(\$2) \land \neg f_2(\$1) \land \neg f_2(\$2) \land f_3(\$3)) \lor (\neg f_0(\$0) \land \neg f_0(\$1) \land \neg f_0(\$2) \land \neg f_1(\$0) \land \neg f_1(\$1) \land$
$\neg f_2(\$1) \land \neg f_2(\$2) \land f_3(\$3)) \lor (\neg f_0(\$0) \land \neg f_0(\$1) \land \neg f_0(\$2) \land \neg f_1(\$1) \land \neg f_1(\$2) \land \neg f_2(\$0) \land \neg f_2(\$2) \land f_3(\$3)) \lor$
$(\neg f_0(\$0) \land \neg f_0(\$2) \land \neg f_1(\$0) \land \neg f_1(\$1) \land \neg f_2(\$0) \land \neg f_2(\$1) \land \neg f_2(\$2) \land f_3(\$3)) \lor (\neg f_0(\$0) \land \neg f_0(\$1) \land \neg f_1(\$1) \land$
$\neg f_1(\$2) \land \neg f_2(\$0) \land \neg f_2(\$1) \land \neg f_2(\$2) \land f_3(\$3)) \lor (\neg f_0(\$0) \land \neg f_0(\$1) \land \neg f_1(\$0) \land \neg f_1(\$1) \land \neg f_2(\$2) \land f_3(\$3)) \lor$
$(\neg f_0(\$0) \land \neg f_0(\$1) \land \neg f_1(\$0) \land \neg f_1(\$1) \land \neg f_1(\$2) \land \neg f_2(\$0) \land \neg f_2(\$2) \land f_3(\$3)) \lor (\neg f_0(\$0) \land \neg f_0(\$2) \land \neg f_1(\$1) \land$
$\neg f_2(\$0) \land \neg f_2(\$2) \land f_3(\$3)) \lor (\neg f_0(\$0) \land \neg f_0(\$2) \land \neg f_1(\$0) \land \neg f_1(\$1) \land \neg f_1(\$2) \land \neg f_2(\$1) \land \neg f_2(\$2) \land f_3(\$3)) \lor$
$(\neg f_1(\$1) \land \neg f_1(\$3) \land \neg f_2(\$1) \land \neg f_2(\$2) \land \neg f_3(\$1) \land \neg f_3(\$2) \land \neg f_3(\$3) \land f_0(\$0)) \lor (\neg f_1(\$1) \land \neg f_1(\$2) \land \neg f_2(\$2) \land$
$\neg f_2(\$3) \land \neg f_3(\$1) \land \neg f_3(\$2) \land \neg f_3(\$3) \land f_0(\$0)) \lor (\neg f_1(\$1) \land \neg f_1(\$2) \land \neg f_2(\$1) \land \neg f_2(\$2) \land \neg f_3(\$3) \land f_0(\$0)) \lor$
$(\neg f_1(\$1) \land \neg f_1(\$2) \land \neg f_2(\$1) \land \neg f_2(\$2) \land \neg f_2(\$3) \land \neg f_3(\$1) \land \neg f_3(\$3) \land f_0(\$0)) \lor (\neg f_1(\$1) \land \neg f_1(\$3) \land \neg f_2(\$2) \land$
$\neg f_3(\$1) \land \neg f_3(\$3) \land f_0(\$0)) \lor (\neg f_1(\$1) \land \neg f_1(\$3) \land \neg f_2(\$1) \land \neg f_2(\$2) \land \neg f_2(\$3) \land \neg f_3(\$2) \land \neg f_3(\$3) \land f_0(\$0)) \lor$
$(\neg f_1(\$1) \land \neg f_2(\$2) \land \neg f_2(\$3) \land \neg f_3(\$2) \land \neg f_3(\$3) \land f_0(\$0)) \lor (\neg f_1(\$1) \land \neg f_1(\$2) \land \neg f_1(\$3) \land \neg f_2(\$1) \land \neg f_2(\$2) \land$
$\neg f_3(\$2) \land \neg f_3(\$3) \land f_0(\$0)) \lor (\neg f_1(\$1) \land \neg f_1(\$2) \land \neg f_1(\$3) \land \neg f_2(\$2) \land \neg f_2(\$3) \land \neg f_3(\$1) \land \neg f_3(\$3) \land f_0(\$0)) \lor$
$(\neg f_0(\$0) \land \neg f_0(\$3) \land \neg f_1(\$0) \land \neg f_1(\$1) \land \neg f_3(\$0) \land \neg f_3(\$1) \land \neg f_3(\$3) \land f_2(\$2)) \lor (\neg f_0(\$0) \land \neg f_0(\$1) \land \neg f_1(\$1) \land$
$\neg f_1(\$3) \land \neg f_3(\$0) \land \neg f_3(\$1) \land \neg f_3(\$3) \land f_2(\$2)) \lor (\neg f_0(\$0) \land \neg f_0(\$1) \land \neg f_1(\$0) \land \neg f_1(\$1) \land \neg f_3(\$3) \land f_2(\$2)) \lor$
$(\neg f_0(\$0) \land \neg f_1(\$1) \land \neg f_1(\$3) \land \neg f_3(\$1) \land \neg f_3(\$3) \land f_2(\$2)) \lor (\neg f_0(\$0) \land \neg f_0(\$1) \land \neg f_0(\$3) \land \neg f_1(\$0) \land \neg f_1(\$1) \land$

$\neg f_3(\$1) \wedge \neg f_3(\$3) \wedge f_2(\$2)) \vee (\neg f_0(\$0) \wedge \neg f_0(\$1) \wedge \neg f_0(\$3) \wedge \neg f_1(\$1) \wedge \neg f_1(\$3) \wedge \neg f_3(\$0) \wedge \neg f_3(\$3) \wedge f_2(\$2)) \vee$
$(\neg f_0(\$0) \wedge \neg f_0(\$1) \wedge \neg f_1(\$0) \wedge \neg f_1(\$1) \wedge \neg f_1(\$3) \wedge \neg f_3(\$0) \wedge \neg f_3(\$3) \wedge f_2(\$2)) \vee (\neg f_0(\$0) \wedge \neg f_0(\$3) \wedge \neg f_1(\$1) \wedge$
$\neg f_3(\$0) \wedge \neg f_3(\$3) \wedge f_2(\$2)) \vee (\neg f_0(\$0) \wedge \neg f_0(\$3) \wedge \neg f_1(\$0) \wedge \neg f_1(\$1) \wedge \neg f_1(\$3) \wedge \neg f_3(\$1) \wedge \neg f_3(\$3) \wedge f_2(\$2)) \vee$
$(\neg f_0(\$0) \wedge \neg f_0(\$3) \wedge \neg f_2(\$0) \wedge \neg f_2(\$2) \wedge \neg f_3(\$0) \wedge \neg f_3(\$2) \wedge \neg f_3(\$3) \wedge f_1(\$1)) \vee (\neg f_0(\$0) \wedge \neg f_0(\$2) \wedge \neg f_2(\$2) \wedge$
$\neg f_2(\$3) \wedge \neg f_3(\$0) \wedge \neg f_3(\$2) \wedge \neg f_3(\$3) \wedge f_1(\$1)) \vee (\neg f_0(\$0) \wedge \neg f_0(\$2) \wedge \neg f_2(\$0) \wedge \neg f_2(\$2) \wedge \neg f_3(\$3) \wedge f_1(\$1)) \vee$
$(\neg f_0(\$0) \wedge \neg f_2(\$2) \wedge \neg f_2(\$3) \wedge \neg f_3(\$2) \wedge \neg f_3(\$3) \wedge f_1(\$1)) \vee (\neg f_0(\$0) \wedge \neg f_0(\$2) \wedge \neg f_0(\$3) \wedge \neg f_2(\$0) \wedge \neg f_2(\$2) \wedge$
$\neg f_3(\$2) \wedge \neg f_3(\$3) \wedge f_1(\$1)) \vee (\neg f_0(\$0) \wedge \neg f_0(\$2) \wedge \neg f_0(\$3) \wedge \neg f_2(\$2) \wedge \neg f_2(\$3) \wedge \neg f_3(\$0) \wedge \neg f_3(\$3) \wedge f_1(\$1)) \vee$
$(\neg f_0(\$0) \wedge \neg f_0(\$2) \wedge \neg f_2(\$0) \wedge \neg f_2(\$2) \wedge \neg f_2(\$3) \wedge \neg f_3(\$0) \wedge \neg f_3(\$3) \wedge f_1(\$1)) \vee (\neg f_0(\$0) \wedge \neg f_0(\$3) \wedge \neg f_2(\$2) \wedge$
$\neg f_3(\$0) \wedge \neg f_3(\$3) \wedge f_1(\$1)) \vee (\neg f_0(\$0) \wedge \neg f_0(\$3) \wedge \neg f_2(\$0) \wedge \neg f_2(\$2) \wedge \neg f_2(\$3) \wedge \neg f_3(\$2) \wedge \neg f_3(\$3) \wedge f_1(\$1)).$

However complicated this may look, note that it is not much different from the previous model where the complexity of the formulas was hidden in the Exactly and AtLeast macro operators.

3.3 Strategies

Definition 10. A *strategy* is a function $\sigma : \mathrm{Form}_X \to E$, determining the next experiment for given accumulated knowledge, such that

$$\varphi_0 \equiv \varphi_1 \Rightarrow \sigma(\varphi_0) = \sigma(\varphi_1).$$

A strategy σ together with a valuation v (satisfying φ_0) induce a *solving process*, which is an infinite sequence

$$\pi_{\sigma,v} = \varphi_0 \xrightarrow{e_1} \varphi_1 \xrightarrow{e_2} \varphi_2 \xrightarrow{e_3} \dots,$$

where $e_i = \sigma(\varphi_0 \wedge \varphi_1 \wedge \dots \wedge \varphi_{i-1})$ and $\varphi_i \in \Phi(e_i)$ and $v(\varphi_i) = 1$, for all $i \in \mathbb{N}$. Notice that due to the condition 3.2, there is always exactly one such ψ_i.
For the sake of simplicity, let us write $\varphi_{0..k}$ instead of $\varphi_0 \wedge \varphi_1 \wedge \dots \wedge \varphi_k$.
We define *length* of the solving process, denoted $|\pi_{\sigma,v}|$ (despite the infinite length of the sequence), as the smallest $k \in \mathbb{N}_0$ such that $\tau_X(\varphi_{0..k}) = 1$. This corresponds to the point where we can uniquely determine the valuation used in the process (i.e. we can determine the secret code). Note that it always holds $\tau(\varphi_{0..k}) > 0$ because $v(\varphi_i) = 1$ for all $i \in \mathbb{N}_0$.
The following lemma is a straightforward consequence of the memory-less nature of the games. It says that once a strategy gives us an experiment that yields no new information, we will never get any new information (using the strategy).

Lemma 11. *If $\tau(\varphi_{0..k}) = \tau(\varphi_{0..k+1})$ for some $k \in \mathbb{N}$, then $\tau(\varphi_{0..k}) = \tau(\varphi_{0..k+l})$ for any $l \in \mathbb{N}$.*

Proof. If $\varphi_{0..k+1} = \varphi_{0..k} \wedge \varphi_{k+1}$ is satisfied by valuation v, so must be $\varphi_{0..k}$. Since $\tau(\varphi_{0..k}) = \tau(\varphi_{0..k+1})$, the sets of valuations satisfying $\varphi_{0..k}$ and $\varphi_{0..k+1}$ must be exactly the same and the formulas are thus equivalent. This implies $\sigma(\varphi_{0..k}) = \sigma(\varphi_{0..k+1})$ and thus also $\varphi_{k+2} = \varphi_{k+1}$. By induction, $\varphi_{k+l} = \varphi_{k+1}$ and $\varphi_{0..k+l} \equiv \varphi_{0..k}$ for any $l \in \mathbb{N}$. ∎

The *worst-case number of experiments* λ^σ of a strategy σ is the maximal length of the solving process $\pi_{\sigma,v}$ over all valuations v by which φ_0 is satisfied, i.e. $\lambda^\sigma = \max_{v \in V_X, v(\varphi_0)=1} |\pi_{\sigma,v}|$. We say that a strategy σ *solves the game* if λ^σ is finite. The game is *solvable* if there exists a strategy that solves the game.
Není to jednoduchý, ale je to zajímavý? Given a code-breaking game $\mathcal{G}$, decide whether $\mathcal{G}$ is solvable.

Definition 12. A strategy σ is *optimal* if $\lambda^\sigma \leq \lambda^{\sigma'}$ for any strategy σ'.

Definition 13. Let $f : \mathrm{Form}_X \to \mathbb{Z}$. A strategy σ is *f-greedy* if for every $\varphi \in \mathrm{Form}_X$ and $e' \in E$,

$$\max_{\psi \in \Phi(\sigma(\varphi)), SAT(\varphi \wedge \psi)} f(\varphi \wedge \psi) \leq \max_{\psi \in \Phi(e), SAT(\varphi \wedge \psi)} f(\varphi \wedge \psi).$$

In words, a greedy strategy minimizes the value of f on the formula in the next step.
We say σ is *greedy* if it is τ_X-greedy.

Problem 14. *Given a code-breaking game $\mathcal{G}$, decide whether all greedy strategies are optimal. Hypothesis: It is the case for Fake-coin problem (?). It is not the case for Mastermind[ref].*

Definition 15. An experiment e_1 is in relation $\sim_\varphi$ with an experiment e_2 if and only if there exists a permutation $\pi \in \mathrm{Perm}_X$ such that $\{\varphi \wedge \psi \mid \psi \in \Phi(e_1)\} \equiv \{(\varphi \wedge \psi)^\pi \mid \psi \in \Phi(e_2)\}$.

3.4 Symmetry Breaking

Motivace: u některých úloh je fakt mrtě experimentů, když uvážíme možné parametrizace. Třeba u standardních 12 koulí u vážení 4 vs 4 $\frac{1}{2} \cdot \binom{12}{4} \cdot \binom{8}{4}$ kombinací. Ne všechny jsou zajímavé. V téhle kapitole zformalizujem, co znamená zajímavé a jak takové vyberem.

Definition 16. An experiment e_1 is in relation $\sim_\varphi$ with an experiment e_2 if and only if there exists a permutation $\pi \in \mathrm{Perm}_X$ such that $\{\varphi \wedge \psi \mid \psi \in \Phi(e_1)\} \equiv \{(\varphi \wedge \psi)^\pi \mid \psi \in \Phi(e_2)\}$.

This is clearly an equivalence relation.

Lemma 17. *Let σ_1, σ_2 be two strageties, such that $\sigma_1(\varphi) \sim_\varphi \sigma_2(\varphi)$ for any $\varphi \in Form_X$. Then for any valuation v $|\pi_{\sigma_1,v}| = |\pi_{\sigma_1,v}|$ and thus also $\lambda^{\sigma_1} = \lambda^{\sigma_2}$.*

Proof. Indukcí k délce $\pi_{\sigma,v}$.

For any accrued knowledge φ, this lemma gives us the right to consider only one of the experiments e_1, e_2 if $e_1 \sim \varphi e_2$.
Now, we would love an algorithm that would, for a given formula φ, generate a set of experiment, such that there is exaclty one expriment from every equivalence class in $E/\sim_\varphi$. ...
For the following sections, let us fix an experiment type t.

Phase 1

Phase 2

4 COBRA - COde BReaking Game Analyzer

4.1 Input language

4.2 Modes

5 Implementation details

6 Conclusions

Bibliography