Solving Deductive Games

Antonín Kučera
(a joint work with Miroslav Klimoš)

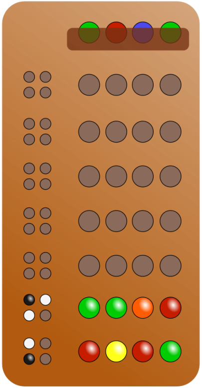
FI MU Brno
Czech Republic

Deductive games

- 2 players: **codemaker + codebreaker**
- codemakers selects a **secret code**
- codebreaker strives to reveal the code through **experiments**
- experiments provide **partial information** about the code
- the goal is to synthesize a **strategy** for the codebreaker s.t.
  - the secret code is eventually discovered;
  - the **worst** (or **average**) number of experiments is minimized.

Example 1: Mastermind

Mastermind



Known results

- Knuth (1976): 5 guesses in the worst case, 4.478 on average
- Irving (1978): 4.369 guesses on average
- Neuwirth (1982): 4.364 guesses on average
- Koyama& Lai (1993): 4.36 guesses on average (this is **optimal**)

Example 2: Counterfeit Coin Problem

Counterfeit Coin Problem

- n coins + balance scale
- All coins except one have the same weight
- Identify the odd-weight coin

Known results

- Dyson (1946): w weighings are sufficient iff $3 \leq N \leq (3^w - 3)/2$
- The average number of weighings has not been analyzed in greater detail
- Guy, Nowakowski (1995): an overview of existing variants and results

More Serious Examples

- Information leakage in security systems
  - Steel (2006), Bond & Zielinski (2003): API-level attacks in ATM hardware security modules
- String Matching Games
  - Erdös & Rényi (1963): Results on asymptotic worst-case complexity
  - Goodrich (2009), Gagneur et al. (2011): Applications in genetics

The Challenge

- Design a **generic** formalism for modeling deductive games.
- Invent algorithms for synthesizing **optimal** worst/average case strategies.
- Implement a **working software tool**.

Formal model

$\mathcal{G} = (X, \varphi_0, \Sigma, F, T)$

- X is a finite set of propositional variables,
- $\varphi_0 \in \mathit{form}(X)$ is a satisfiable **initial constraint**,
- Σ is a finite set of **parameters**,
- $F \subseteq X^\Sigma$ is a set of **attributes** with pairwise disjoint images,
- T is a finite set of **parameterized experiments** of the form (k, P, Φ) where
  - $k \in \mathbb{N}$ is the number of parameters,
  - $P \subseteq \Sigma^k$ is a set of **instances**,
  - Φ is a finite subset of $\mathit{form}(X \cup \{f(\$j) \mid f \in F, 1 \leq j \leq k\})$ whose elements are called **outcomes**.

Example – CCP with 4 coins

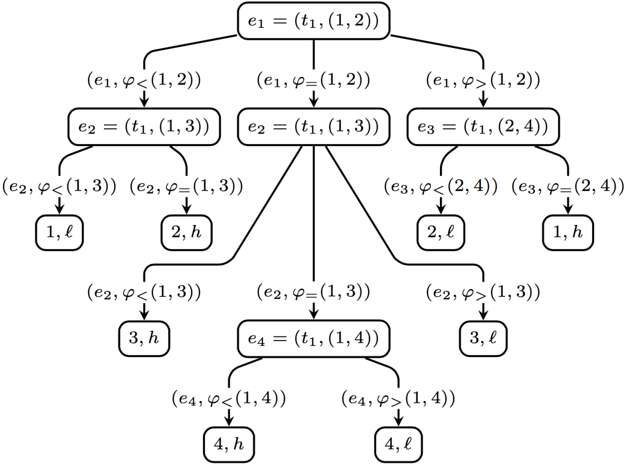
- $X = \{x_1, x_2, x_3, x_4, y\}$,
- $\varphi_0 = \text{EXACTLY}_1(x_1, x_2, x_3, x_4)$,
- $\Sigma = \{coin_1, coin_2, coin_3, coin_4\}$,
- $F = \{d\}$ where $d(coin_i) = x_i$ for every $1 \leq i \leq 4$,
- $T = \{ \ (2, \Sigma^{\langle 2 \rangle}, \{\varphi_<, \varphi_=, \varphi_>\}), \ (4, \Sigma^{\langle 4 \rangle}, \{\psi_<, \psi_=, \psi_>\}) \ \}$, and

$$
\begin{aligned}
\varphi_< &= (d(\$1) \wedge \neg y) \vee (d(\$2) \wedge y) \\
\varphi_= &= \neg d(\$1) \wedge \neg d(\$2) \\
\varphi_> &= (d(\$1) \wedge y) \vee (d(\$2) \wedge \neg y) \\
\psi_< &= ((d(\$1) \vee d(\$2)) \wedge \neg y) \vee ((d(\$3) \vee d(\$4)) \wedge y) \\
\psi_= &= \neg d(\$1) \wedge \neg d(\$2) \wedge \neg d(\$3) \wedge \neg d(\$4) \\
\psi_> &= ((d(\$1) \vee d(\$2)) \wedge y) \vee ((d(\$3) \vee d(\$4)) \wedge \neg y)
\end{aligned}
$$

Example – Mastermind with n pegs and m colors

- $X = \{x_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq m\}$
- φ_0 says that each peg has precisely one color
- $\Sigma = \{color_1, \ldots, color_m\}$
- $F = \{peg_1, \ldots, peg_n\}$
- T contains just one experiment with "many" outcomes

Decision tree for a simple strategy (CCP with 4 coins)



Problems

Consider CCP with 60 coins

- There are more than 10^{63} ways of instantiating the weighing of $20 + 20$ coins.
- If we spent 1 ns with processing each instance, we need more than 10^{46} years to go over all of them (the estimated age of our Universe is about 10^9 years).
- Cobra **can** analyze CCP with more than 60 coins...

How It Works

When assembling the next experiment, we exploit symmetries.

- **Phase 1:** Generate a list of experiments by "intelligent backtracking".
- **Phase 2:** Go over the list and try to identify and eliminate "symmetric" experiments (here we employ tools for checking graph isomorphism).
- **Phase 3:** Evaluate all experiments, select the most promising one (here we employ SAT solvers).

COBRA (COde-BReaking game Analyzer)

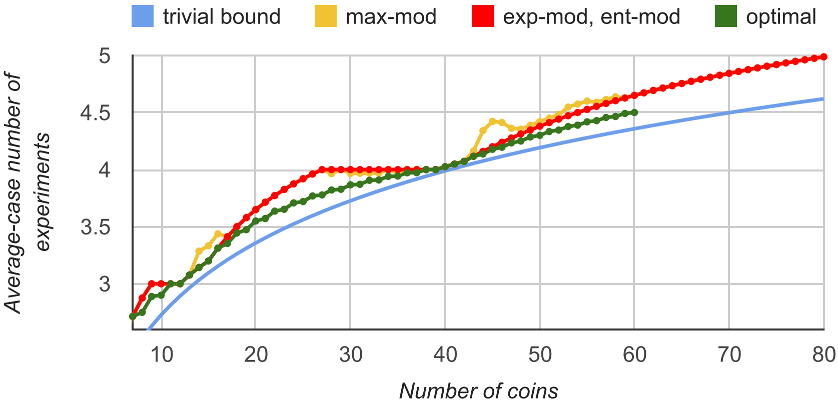
- command-line tool written in C++
- takes game specification (language based on Python)
- two modes:
  - compute the complexity of a given ranking strategy
  - compute worst/average-case optimal strategy

Game specification example (CCP)

```
n = 4
xvars = ["x1", "x2", "x3", "x4"]
VARIABLES(xvars + ["y"])
CONSTRAINT("Exactly-1(%s)" % ",".join(xvars))
ALPHABET(xvars)
MAPPING("X", xvars)

for m in range(1, n//2 + 1):
  EXPERIMENT("weighing" + str(m), 2*m)
  PARAMS_DISTINCT(range(1, 2*m + 1))
  OUTCOME("lighter", "((%s) & !y) | ((%s) & y)" ...
  OUTCOME("heavier", "((%s) & y) | ((%s) & !y)" ...
  OUTCOME("same", "!(%s)" % params(1, 2*m))
```

Results I



trivial bound
max-mod
exp-mod, ent-mod
optimal
Average-case number of experiments
5
4.5
4
3.5
3
10
20
30
40
50
60
70
80
Number of coins

Results II

Average-case			
Size	MM	MM+col	MM+pos
2x8	3.67187	3.64062	2
3x6	3.19444	3.18981	3
4x4	2.78516	2.74609	2.78516
Worst-case			
Size	MM	MM+col	MM+pos
2x8	5	5	2
3x6	4	4	3
4x4	3	3	3

Two phases of symmetry breaking

- Phase 1: Generate parameters
  - Generate one-by-one, investigate parameter prefixes
  - A prefix can be completely *dominated by* another prefix under some conditions
- Phase 2: Eliminate symmetric experiments

Two phases of symmetry breaking

- Phase 1: Generate parameters
  - Generate one-by-one, investigate parameter prefixes
  - A prefix can be completely *dominated by* another prefix under some conditions
- Phase 2: Eliminate symmetric experiments

CCP 26 ($\approx 10^{26}$ exp.)		**CCP 39** ($\approx 10^{46}$ exp.)		**CCP 50** ($\approx 10^{64}$ exp.)	
Phase 1	Phase 2	Phase 1	Phase 2	Phase 1	Phase 2
13.0	13.0	19.0	19.0	25.0	25.0
4,365.0	861.7	26,638.7	3,318.0	83,625.0	8,591.0
603.0	36.4	2,263.0	88.1	5,733.4	172.2
76.3	4.2	214.7	7.2	405.1	10.4
-	-	-	-	153.2	4.1

Conclusion

- Formal model based on propositional logic
- Generic tool to analyze deductive games
- Main advantage: **versatility**

- Next challenge: push the boundaries of what is feasible