DISCRETE GAMES ON GRAPHS MODELED IN GAME LOGIC

Anita Li, Brandon Bohrer, André Platzer

Carnegie Mellon University

Introduction

Game logic is a formal logic for proving safety and liveness properties of first order games, which are a programming language with discrete computations and adversarial dynamics. Since any game that can be modeled by game logic either has an Angel-win strategy or a Demon-win strategy, a formal proof in Game Logic has several implications in its applications, by modeling pursuit-evasion and defense-attack scenarios that are common in robotics and cybersecurity.

This work investigates how we can model graph games, taking Cops vs Robbers game as an example, in Game Logic and prove specific theorems for common families of graphs. The winning strategies for cops/robbers on these families of graphs serve as a good starting point for different variations of Cops vs Robbers game and can be combined to solve for more complex graph models.

Cops vs Robbers Game

Cops vs Robbers (also known as Search and Evasion) game has various implications in robotic graph search. The simple graphs we choose can model the physical environment, and the searcher/evader can model the robots in their working conditions.

Cops and Robbers is a classic graph game that has been widely discussed. The game rules are simple. Two players, a cop and a robber, compete on a simple graph G. The cop starts by choosing a start vertex, and then the robber chooses a start vertex. After that, the two players move to an adjacent node, by turns (starting with the cop). The two players can see all the moves. The cop wins the game if he catches the robber by being on the same vertex as the robber at some point. The robber wins if he never gets captured.

Model winning condition

Let G be our graph, $V(G)$ be the set of vertices and $E(G)$ be the set of edges. Each edge is denoted (i, j). Denote the cop's position c and the robber's position r where $c, r \in V(G)$. Below is the syntax of dGL.

$$a; b ::= x := e \mid ?Q \mid a \cup b \mid a; b \mid a^* \mid a^d$$

Define a game round **CR** as

$$\{c_o := c; c := c'; ?c_o c \in E(G)\}; \{?(c \neq r); r_o := r; r := r'; ?(r_o, r) \in E(G)\}^d$$

Define the cop's winning condition as

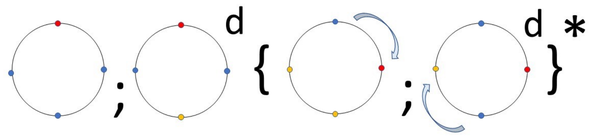
$$\langle c := c_0; ?c \in V(G); \{r := r_0; ?r \in V(G)\}^d; \mathbf{CR}^*\rangle c = r$$

and the robber's winning condition as

$$[c := c_0; ?c \in V(G); \{r := r_0; ?r \in V(G)\}^d; \mathbf{CR}^*]c \neq r$$

The $\langle\rangle$ here denotes Angel's move, and $[]$ denotes Demon's move. The cop is Angel because he will not end the game until he catches the robber and therefore determines the round of the game. Since the game starts with the cop making decisions, he only needs 'one' certain strategy to win the game rather than 'every' single scenario.

We can use the following example to illustrate. The cop is colored red and the robber is colored yellow. The left two graphs are initialization, and the right two graphs are an example game round. Although the graphs show only the assignment steps, the effect of tests are implicitly restricting the moves to be legal.



Assignment: $x := e$. This is used to assign a new position for c or r.
Test: $?Q$. A test does not change the game state, but fails immediately if Q is not passed.
Sequential: $\alpha; \beta$. Sequential games are a reflection of time flow. The active player plays α first, and then plays β.
Dual: α^d. The active player switches (in our case to robber), and plays game α.
Repetition: α^*. The active player chooses when to terminate the game (in our case cop).

Proof Sketch

We will need the `loop` rule to understand the proof for robber-win graphs:

$$\frac{P \vdash J \quad J \vdash [a]J \quad J \vdash Q}{P \vdash [a^*]Q}\, loop$$

Although in robber-win games, cops will never win no matter how long the game is played, we can break down the $*$ rule by proving an invariant J always holds and guarantees the cop's winning condition cannot be met.

- Start by picking arbitrary vertex for the cop to start on. Pick a specific vertex for the robber.
- Prove that a game invariant J holds.
- After a game round of **CR**, prove that J holds.
- Prove that when J holds, we can prove $c \neq r$.

We will need `ind` rule to understand the proof for cop-win graphs:

$$\frac{G \vdash J(c) \quad c_o = c, J(c), c > k \vdash \langle a\rangle(c < c_o \wedge J(c)) \quad c \leq k \wedge J(c) \vdash P}{G \vdash \langle a^*\rangle P}\, ind$$

The cop-win strategy must terminate in some indefinite number of steps. Therefore, we need an invariant J to hold after each round of the game, and a variable c to be strictly decreasing until it reaches minimum k. When c is at minimum the cop's winning condition can be proved.

- Start by picking a vertex for the cop to start on. Pick arbitrary vertex for the robber.
- Prove that a game invariant J holds.
- After a game round of **CR**, prove that J holds and a variable v is strictly decreasing.
- Prove that once v reaches a lower bound, we can use J to prove $c = r$.

We can use the previous example to illustrate. In order to prove the 4-cycle is cop-win, we would set $J := r = c + 2 \bmod 4$. To prove this holds after a game round, we can case on two possible cop moves and move the robber respectively.

Remark: Each step above is a game logic rule, and the entire proof can be expanded into a formal proof tree. For the sake of simplicity, we will omit the details and focus only on the key steps in the following examples.

Examples

Graph	Win Type	Init Position	Invariant	Termination
Cycle $C_n(n \geq 4)$	Robber-win	$r = c + 2 \bmod n$	$r = c + 2 \bmod n$	N/A
Path P_n	Cop-win	Anywhere	$r \leq c$	c
Tree T_n	Cop-win	Anywhere	True	$S(c, r) := \{v : c \notin p(v, r)\}$
DAG[1]	Robber-win	$\forall c, \exists r, d(r, c) = \infty$	$d(r, c) = \infty$	N/A
DAG[2]	Cop-win	$\exists c, \forall r, d(r, c) \neq \infty$	True	$S(c, r) := \{v : c \notin p(v, r)\}$

Remark: DAG denotes Directed Acyclic Graph. We use a slightly modified version of the game, by allowing the cop and the robber to stay on the original position on each round. DAG^1 denotes the family of DAGs that has a vertex that can reach any other vertices. DAG^2 denotes the family of DAGs that are not DAG^1. We defined $p(u, v)$ to be the vertices on the path from u to v. Since this is only used on acyclic graphs, this path is unique.

Conclusion

We have used Game Logic to prove the winning strategies for cycles, paths, trees, DAGs and chordal graphs (for simplicity omitted here). These examples can lead to further work on investigating more complex graph models and game variations to better model real-life conditions.

[1] Brian Alspach. "Searching and sweeping graphs: A brief survey". In: *Le Matematiche; Vol 59, No 1,2 (2004); 5-37* 59 (Nov. 2004).

[2] Anthony Bonato and Richard Nowakowski. *The Game of Cops and Robbers on Graphs*. Sept. 2011. DOI: 10.1090/stml/061.

[3] André Platzer. *Logical Foundations of Cyber-Physical Systems*. Cham: Springer, 2018. ISBN: 978-3-319-63587-3. DOI: 10.1007/978-3-319-63588-0. URL: http://www.springer.com/978-3-319-63587-3.