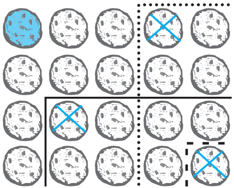
Automatic Verification of FSA Strategies via Counterexample-Guided Local Search for Invariants

Kailun Luo, Yongmei Liu
Dept. of Computer Science, Sun Yat-sen University

August 12, 2019

Motivation

- Strategy representation and reasoning receives much attention in KRR, e.g., Alternating-time Temporal Logic (ATL) and Strategy Logic (SL).
- Situation calculus game structure [De Giacomo, Lespérance, and Pearce 2010], automatic verification of Golog programs [Li and Liu 2015; Mo, Li, and Liu 2016]
- We consider general strategy representation and its automatic verification, e.g., see the Chomp game:



- Two-player, turn-based
- Size: NxM, top left: poisoned
- Rule: eat a cookie, together with all cookies to the right or below it.

Problem

A set of games **+** *a general strategy*

$\Rightarrow$ *Is it a winning strategy for all the games?*

Outline

1. FSA Strategies as General Strategies

2. Automatic Verification

The Situation Calculus

The Situation Calculus [John McCarthy 1969] (SitCal) is a many-sorted first-order logical language for representing dynamic worlds:

- Action: function, e.g., $eat(p, x, y)$
- Situation: action sequence, e.g., S_0 and $do(a, S_0)$
- Fluent: special predicate, e.g., $ch(x, y, s)$

Based on SitCal, a basic action theory [Reiter 2001] (BAT) $\mathcal{D}$

- consist five parts:

$$\Sigma \cup \mathcal{D}_{ap} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}$$

- represents a class (possibly infinite many) of games

BAT represents a class of games

Take Chomp NxN for example:

- Initial database:

$$ch(x, y, S_0) \equiv 0 < x \leq N \wedge 0 < y \leq M, N = M$$

- Precondition axioms: $Poss(eat(p, x, y), s)$

$$\equiv turn(p, s) \wedge ch(x, y, s)$$

- Successor state axioms: $ch(x, y, do(a, s))$

$$\equiv \exists p, i, j. a = eat(p, i, j) \wedge (i > x \vee j > y)$$

- Additional axioms:

$$win(p, s) \doteq turn(p); \qquad end(s) \doteq \neg ch(1, 1, s)$$

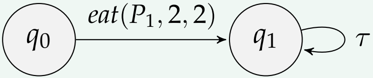
FSA Strategy

- An FSA strategies is a finite state automata except that its edge labels are single-step complex actions.
- FSA strategies represent general strategies, e.g.,

Strategy for Chomp $N \times N$

- eat position $(2,2)$
- If the opponent eat (x,y), eat (y,x)

FSA strategy representation



$\tau : \pi(x,y).last(x,y)?; eat(P_1,y,x)$

Winning strategy

- Complete strategy: always has a move until game ends;
- Composite strategy: all the possible plays between players;
- $\pi a.a$ strategy: do any possible action;
- $T^*_S(q, s, q', s')$: in (q, s) follow S, then (q', s') will be reached;

Definition

Given an BAT $\mathcal{D}$ and a complete FSA strategy S for player p, we say S is a winning strategy if the composition C of S and $\pi a.a$ strategy satisfies (second order theorem-proving task)

$$\mathcal{D} \models \forall q, s. T^*_C(Q_0, S_0, q, s) \wedge end(s) \supset win(p, s).$$

Intuition: FSA strategy S is winning if with S, player p always wins when the opponent adopts the $\pi a.a$ strategy.

Automatic Verification

Basic idea: Find sound invariant

From second order to first order:

- Let $\mathcal{X}$ be a labelling function: labels each FSA state with a first-order formula. (it characterizes state information of situations)
- $\mathcal{X}$ is a sound invariant for strategy S if
  - Invariant: for any edge $q \xrightarrow{\tau} q'$ in strategy S,

$$\mathcal{D} \models \forall s, s'. \mathcal{X}(q)[s] \wedge Do(\tau, s, s') \supset \mathcal{X}(q')[s'].$$

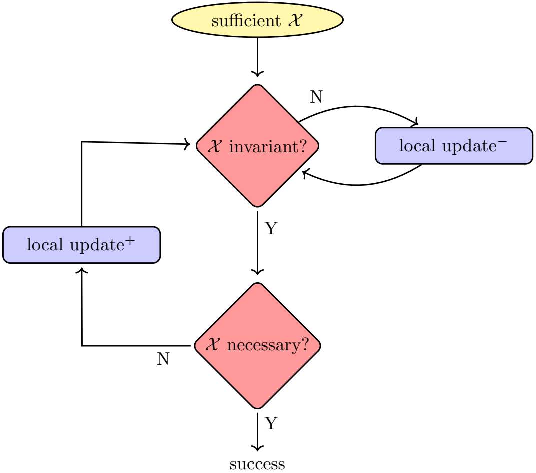
  - Necessary: (Let Q_0 be the starting state of strategy S)

$$\mathcal{D}_{S_0} \models \mathcal{X}(Q_0)[S_0].$$

  - Sufficient: for each state q in strategy S,

$$\mathcal{D} \models \mathcal{X}(q)[s] \supset [end(s) \supset win(p, s)].$$

Find sound invariant $\mathcal{X}$



A formula $\mathcal{X}(q)$ has the form $\forall^*.c_1 \wedge \ldots \wedge c_n$ (c_i is a clause)

local update$^-$ wrt model M

Because $\exists c_i$ s.t. $M \not\models c_i \implies M \not\models \mathcal{X}(q)$, we modify a clause to **exclude** model M

local update$^+$ wrt model M

Because $\forall c_i$ s.t. $M \models c_i \implies M \models \mathcal{X}(q)$, we modify all clauses to **include** model M

Tackle large formula space

- Local update (when updating c_i): Just 'flip' a few predicates inside c_i, *e.g.*, $\forall^*(P_1 \vee P_2 \vee P_3) \rightsquigarrow \forall^*(P_1 \vee P_2 \vee P_4)$.
- Predicates are extracted from specifications, and of the form $t = f(\vec{t})$ or $P(\vec{t})$, where $\vec{t}$ are terms, f is a function and P is a predicate.
- Use at most $m \geq 2$ different variables $x_1, \ldots, x_m$ in each generated predicate.

Experimental results

- SMT solver Z3 for first-order reasoning.
- Combinatorial games and planning domains are tested.

Name	C	P	U^-	U^+	B	R	T(s)
PickS123	3	59	3	3	0	0	7.3
PickS134	3	65	3	3	0	0	10.6
chp 2×N	4	41	46	17	5	2	817.6
chp N×N	4	58	11	4	0	0	99.9
Clobber*	3	92	53	11	13	2	1198.6
Clobber	3	92	-	-	-	-	-
Colouring	3	44	38	13	0	9	188.7
1d	3	34	4	3	0	0	6.2
Arith	3	34	5	3	0	0	6.5
Find	3	50	3	4	0	0	13.8
Sort	3	59	20	10	3	0	540.9
Add	4	41	7	2	0	0	8.5
PrizeA1	5	49	61	5	1	0	1300.1

Conclusion

- Provide a natural representation for general strategies.
- Propose a sound but incomplete method for verifying whether an FSA strategy is a winning strategy.
- Limitation: invariants considered are of the form CNF formula where variables are universally quantified.

Future works:

- Consider more expressive invariants which allow existential quantification;
- Explore automatic synthesis of FSA strategies.

Thank you for your listening!