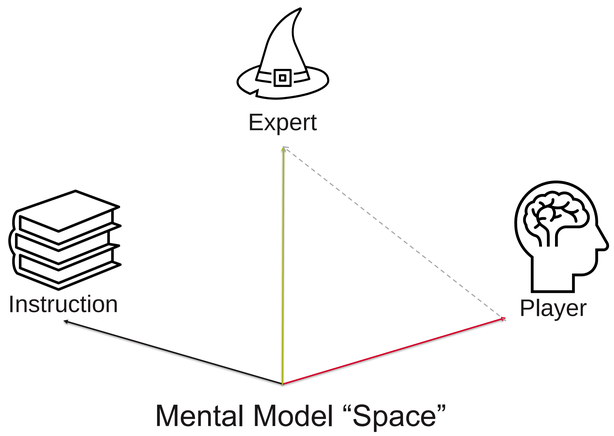
Learning Explainable Representations of Complex Game-playing Strategies

Abhijeet Krishnan, Colin M. Potts, Arnav Jhala, Harshad Khadilkar, Shirish Karande and Chris Martens
akrish13@ncsu.edu

Advances in Cognitive Systems
Palermo, Italy
18th June, 2024

Teaching as Mental Model Alignment



- **Mental Model**: player's representation of how actions affect state (Rouse et. al. 1992)
- **Improvement** = alignment of a player's mental model (Boyan et. al. 2018)
- Tools for alignment
  - Instruction
  - Practice
- Could we *automatically generate* instructions to help players improve?

Strategies in Games

- Examples –
  - Zerg rush in *Starcraft II*
  - Keepout in fighting games
  - Chess tactics – fork, pin, skewer
- **Strategy** = game-playing policy, and vocabulary/concepts to *explain* it
- Used in educational literature for games
- Could we generate *good*, *explainable* strategies for player improvement?

Interpretable Strategy Synthesis

- G: MDP for a game
- M: *strategy model*
- P: performance measure
- I: interpretability measure

Learn a *strategy* as an instance of a strategy model (M) that jointly optimizes a performance measure (P) and an interpretability measure (I).

$$\sigma^* = \underset{\sigma}{\operatorname{argmax}}\, P(\sigma)I(\sigma),\ \sigma \in M$$

Strategy Models

```
strategy(State, Action) ←
    feature_1 (· · · ),
    feature_2 (· · · ),
        ...
    feature_n (· · · )
```

First-order logic (FOL) rules

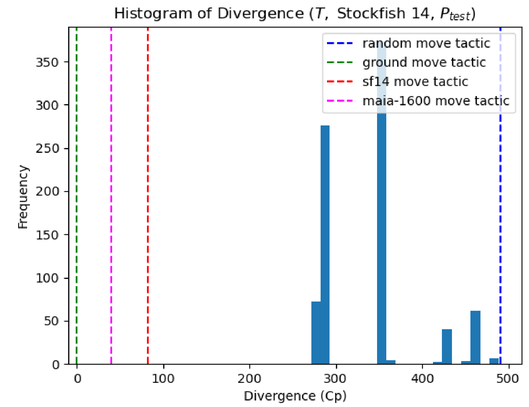
```
DEF run m(
  WHILE c( markersPresent c) w(
    turnRight
  w)
  move
  IF c( not c( markersPresent c) c) i(
    turnRight
    move
  i)
m)
```

Domain-specific language (DSL) script

- **Strategy model**: interpretable policy model
- Examples: first-order logic rules, domain-specific language scripts, decision trees, if-else rules, etc. (Puiutta, E., & Veith, E. M., 2020)

FOL Strategies for Chess

- Learned using *Inductive Logic Programming* (ILP) from chess gameplay data
- Strategies outperform random baseline in imitating human beginners



Programmatic Strategies

- Strategies learned for solving tasks in Karel, a programmatic grid-based environment
- Learned via offline reinforcement learning using a *decision transformer* (DT) model
- Learned strategies are competitive with a program synthesis baseline while requiring less data

Task	Mean Return		Unique Programs	
	LEAPS	DT	LEAPS	DT
cleanHouse	0.16 (0.13)	0.23	3627	59
fourCorners	0.35 (0.00)	0.35	9872	55
harvester	0.61 (0.21)	0.66	11708	28
randomMaze	0.97 (0.04)	1.0	295	63
stairClimber	0.74 (0.49)	1.1	298	49
topOff	0.80 (0.11)	0.66	30278	63