

# Interpretable Strategy Synthesis for Competitive Games

## Thesis Defense Presentation

Abhijeet Krishnan

Department of Computer Science  
North Carolina State University

July 16, 2024

# Publications

- Krishnan, Abhijeet, Colin M. Potts, Arnav Jhala, Harshad Khadilkar, Shirish Karande and Chris Martens. "Learning Explainable Representations of Complex Game-playing Strategies." *Proceedings of the Eleventh Annual Conference on Advances in Cognitive Systems*. 2024.
- Villalobos-Arias, Leonardo, Derek Martin, Abhijeet Krishnan, Madeleine Gagné, Colin M. Potts and Arnav Jhala. "Modeling Risk in Reinforcement Learning: A Literature Mapping." *arXiv preprint arXiv:2312.05231*. 2023.
- Krishnan, Abhijeet and Chris Martens. "Synthesizing Chess Tactics from Player Games." In *Workshop on Artificial Intelligence for Strategy Games (SG) and Esports Analytics (EA), 18th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. 2022.
- Krishnan, Abhijeet and Chris Martens. "Towards the Automatic Synthesis of Interpretable Chess Tactics." In *Explainable Agency in Artificial Intelligence Workshop, 36th AAAI Conference on Artificial Intelligence*. 2022.
- Krishnan, Abhijeet, Aaron Williams, and Chris Martens. "Towards Action Model Learning for Player Modeling." *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. Vol. 16. No. 1. 2020.
- Krishnan, Abhijeet and Chris Martens. "Rule-based Cognitive Modeling via Human-Computer Interaction." Poster presented at: *5th LAS Research Symposium*; 2019 Dec 10; Raleigh, NC.

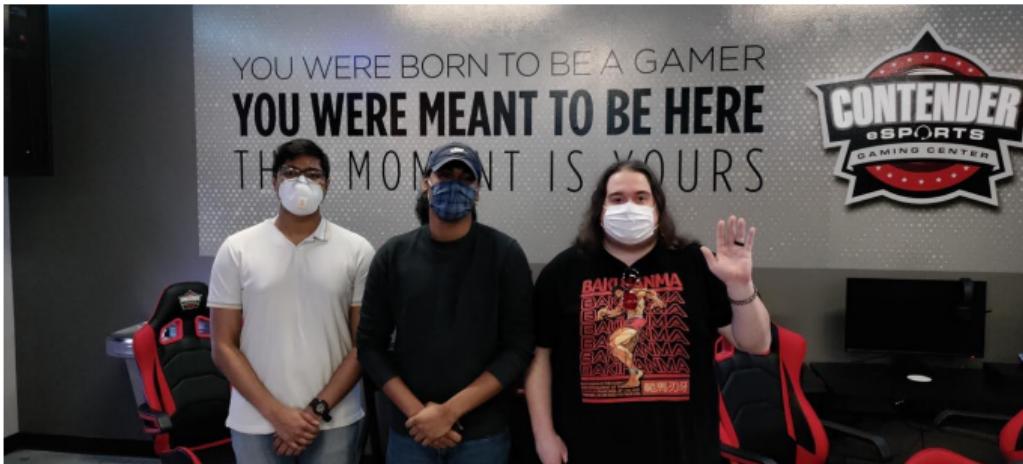
# Getting Into Competitive Gaming



Kevin Brown

Participating in a competitive *Tekken 7* tournament

# Wanting to Improve



Contender Esports

Placing 3rd

20 FEB 2022  
Tekken 7 at Tournament of Thrones 18  
Raleigh, NC 11 Entrants

2nd  
PLACE  
out of 11

1. HTL   Clam
2. Abhijeet
3. Ingress

15 JAN 2022  
Tekken 7 at Last Level: Raleigh 5  
Cary, NC 8 Entrants

3rd  
PLACE  
out of 8

1. HTL   Clam
2. DeL
3. Abhijeet

21 NOV 2021  
Tekken 7 at Tournament of Thrones 15  
Raleigh, NC 6 Entrants

2nd  
PLACE  
out of 6

1. HTL   Clam
2. STFU   Abhijeet
3. Ahematical

start.gg

Results of recent tournaments

# How to Improve?

Improving at a skill —

# How to Improve?

Improving at a skill —

- Play more

# How to Improve?

Improving at a skill —

- Play more → *time-consuming*

# How to Improve?

Improving at a skill —

- Play more → *time-consuming*
- Use a learning resource

# How to Improve?

Improving at a skill —

- Play more → *time-consuming*
- Use a learning resource → *not standardized*

# How to Improve?

Improving at a skill —

- Play more → *time-consuming*
- Use a learning resource → *not standardized*
- Find a coach

# How to Improve?

Improving at a skill —

- Play more → *time-consuming*
- Use a learning resource → *not standardized*
- Find a coach → *not scalable*

# How to Improve?

Improving at a skill —

- Play more → *time-consuming*
- Use a learning resource → *not standardized*
- Find a coach → *not scalable*
- Study better players

# How to Improve?

Improving at a skill —

- Play more → *time-consuming*
- Use a learning resource → *not standardized*
- Find a coach → *not scalable*
- Study better players → *difficult*

# Possible Solution

Could we *automatically* generate *explainable* strategies for players to learn from?

# Strategies are Mental Models of Games

## Claim

Strategies can be treated as mental models of games

# Strategies are Mental Models of Games

## Claim

Strategies can be treated as mental models of games

- Players develop *mental models* of games to explain them (Boyan & Sherry, 2011)

# Strategies are Mental Models of Games

## Claim

Strategies can be treated as mental models of games

- Players develop *mental models* of games to explain them (Boyan & Sherry, 2011)
- Players acquire and *refine* their mental models through practice, instruction and discussion (of strategies) (Boyan et al., 2018)

# Strategies are Mental Models of Games

## Claim

Strategies can be treated as mental models of games

- Players develop *mental models* of games to explain them (Boyan & Sherry, 2011)
- Players acquire and *refine* their mental models through practice, instruction and discussion (of strategies) (Boyan et al., 2018)
- Strategies can be learned from *other players* (Weintrop & Wilensky, 2013)

# Strategies are Mental Models of Games

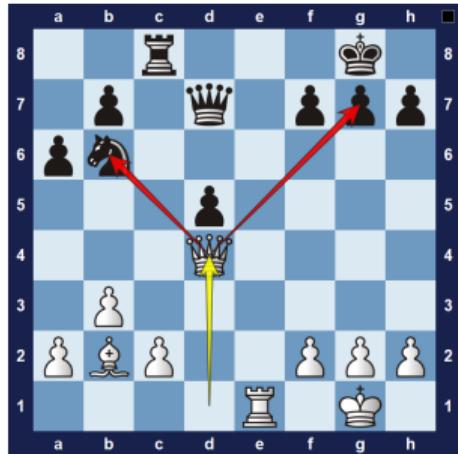
## Claim

Strategies can be treated as mental models of games

- Players develop *mental models* of games to explain them (Boyan & Sherry, 2011)
- Players acquire and *refine* their mental models through practice, instruction and discussion (of strategies) (Boyan et al., 2018)
- Strategies can be learned from *other players* (Weintrop & Wilensky, 2013)
- Learned strategies can be *transferred* to one's own gameplay (Paredes-Olay et al., 2002)

# Real-world Strategies

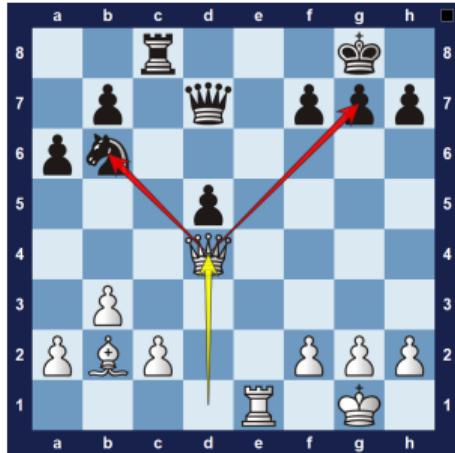
# Real-world Strategies



Chessfox

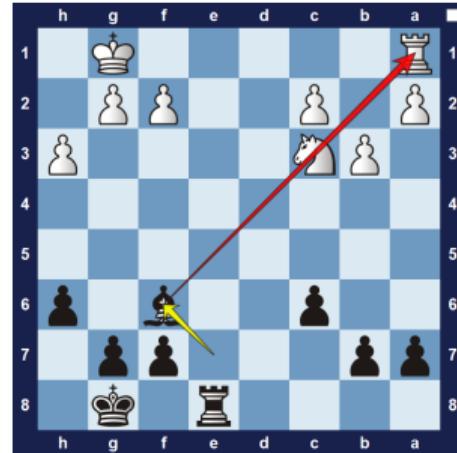
An example of the *fork* tactic in chess

# Real-world Strategies



Chessfox

An example of the *fork* tactic in chess



Chessfox

An example of the *pin* tactic in chess

# Real-world Strategies



That Blasted Salami

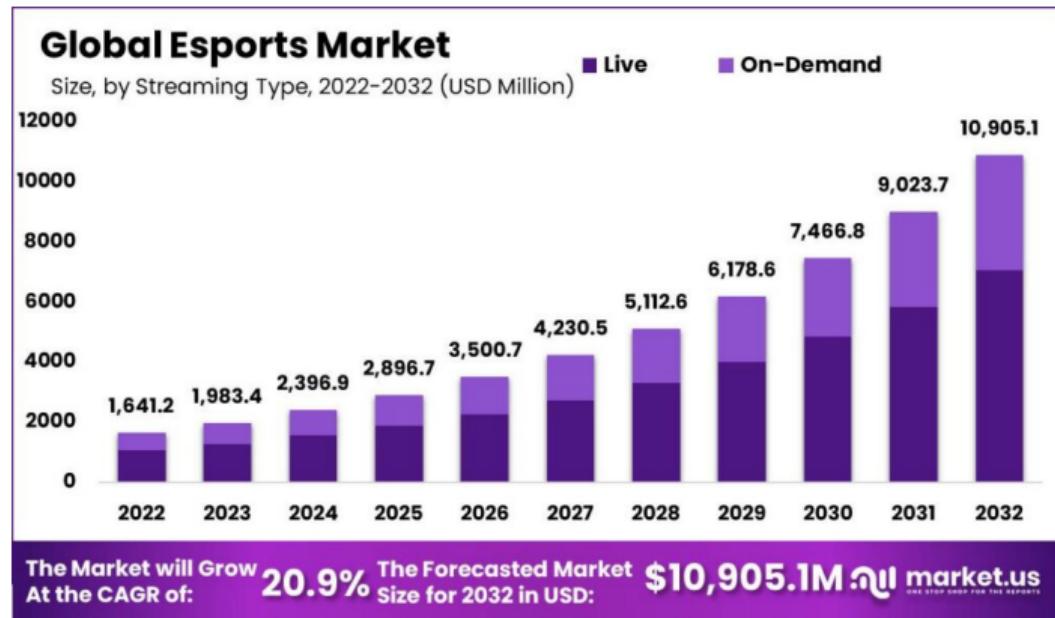
A mixed strategy in *Tekken*

# Benefits

# Benefits

- Competitive gaming → democratized access to strategies

# Esports Viewership



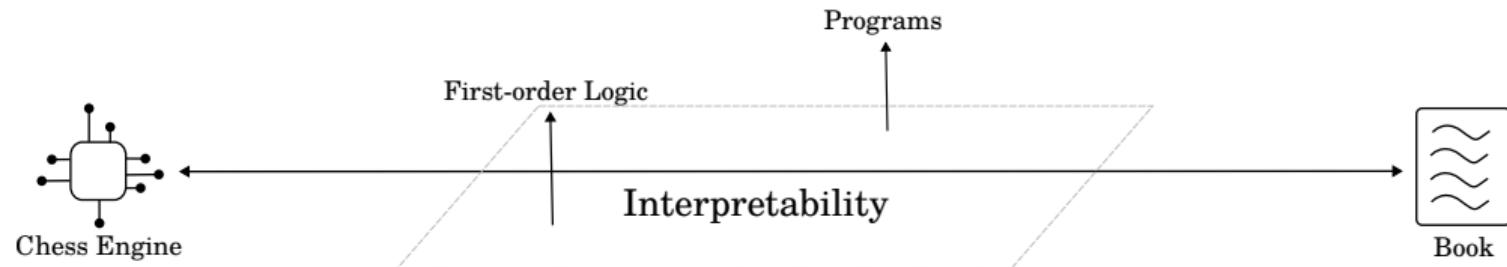
market.us

# Benefits

- Competitive gaming → democratized access to strategies
- Explainable AI → techniques for explainability

# Benefits

- Competitive gaming → democratized access to strategies
- Explainable AI → techniques for explainability



# Thesis Statement

## Thesis Statement

A computational model of a game strategy, along with a synthesis method, could meet the goals of discovering high-quality, interpretable strategies and impact the fields of competitive gaming and explainable AI.

# RQ1

## Thesis Statement

A *computational model* of a game strategy, along with a *synthesis method*, could meet the goals of discovering *high-quality*, *interpretable* strategies and impact the fields of competitive gaming and explainable AI.

## RQ1

How do we formally define the problem of *Interpretable Strategy Synthesis* (ISS)?

# RQ2

## Thesis Statement

A *computational model* of a game strategy, along with a synthesis method, could meet the goals of discovering high-quality, interpretable strategies and impact the fields of *competitive gaming* and explainable AI.

## RQ2

How do we approach the problem of ISS using a *logic-based* strategy model?

# RQ3

## Thesis Statement

A *computational model* of a game strategy, along with a synthesis method, could meet the goals of discovering high-quality, interpretable strategies and impact the fields of competitive gaming and *explainable AI*.

## RQ3

How do we approach the problem of ISS using a *program-based* strategy model?

# ISS Framework

RQ1

How do we formally define the problem of *Interpretable Strategy Synthesis* (ISS)?

# ISS Framework

## RQ1

How do we formally define the problem of *Interpretable Strategy Synthesis* (ISS)?

- Why do we need a framework?

# ISS Framework

## RQ1

How do we *formally define* the problem of Interpretable Strategy Synthesis (ISS)?

- Why do we need a framework?
- Definitions of key elements —
  - ISS
  - Strategy Model
  - Strategy
  - Performance Measure
  - Interpretability Measure

# The Need for a Framework

Paper	Number Used			Interpretability
	Domains	Models	Algorithms	
Spronck, Sprinkhuizen-Kuyper, and Postma (2004)	2	1	1	✗
de Mesentier Silva, Isaksen, Togelius, and Nealen (2016)	1	1	4	✓
Butler, Torlak, and Popović (2017)	1	1	1	✗
Canaan et al. (2018)	1	1	1	✗
de Freitas, de Souza, and Bernardino (2018)	1	1	1	✗
Mariño, Moraes, Oliveira, Toledo, and Lelis (2021)	1	1	1	✗
Krishnan and Martens (2022a)	1	1	1	✗
Mariño and Toledo (2022)	1	1	1	✗
Medeiros, Aleixo, and Lelis (2022)	2	1	2	✗

Table: List of works in ISS

# Elements of a Good Framework

# Elements of a Good Framework

- Facilitates *comparison*
  - multiple *games*
  - multiple *strategy representations*
  - multiple *learning algorithms*

# Elements of a Good Framework

- Facilitates *comparison*
  - multiple *games*
  - multiple *strategy representations*
  - multiple *learning algorithms*
- Provides a clear definition of interpretability

# Interpretable Strategy Synthesis (ISS)

## Definition (ISS)

Formal

Given a —

# Interpretable Strategy Synthesis (ISS)

## Definition (ISS)

Formal

Given a —

- Game environment  $\mathcal{G}$

# Interpretable Strategy Synthesis (ISS)

## Definition (ISS)

Formal

Given a —

- Game environment  $\mathcal{G}$
- Strategy model  $\mathcal{M}$

# Interpretable Strategy Synthesis (ISS)

## Definition (ISS)

Formal

Given a —

- Game environment  $\mathcal{G}$
- Strategy model  $\mathcal{M}$
- Performance measure  $J$

# Interpretable Strategy Synthesis (ISS)

## Definition (ISS)

Formal

Given a —

- Game environment  $\mathcal{G}$
- Strategy model  $\mathcal{M}$
- Performance measure  $J$
- Interpretability measure  $\mathcal{I}$

# Interpretable Strategy Synthesis (ISS)

## Definition (ISS)

Formal

Given a —

- Game environment  $\mathcal{G}$
- Strategy model  $\mathcal{M}$
- Performance measure  $J$
- Interpretability measure  $\mathcal{I}$

The problem of ISS is to find a strategy  $\sigma^* \in \mathcal{M}$  that *maximizes both  $J$  and  $\mathcal{I}$* .

# Strategy Model ( $\mathcal{M}$ )

- Defines the *space* (or architecture) of strategies

# Strategy Model ( $\mathcal{M}$ )

- Defines the *space* (or architecture) of strategies
- Examples —
  - if-then rules
  - decision trees
  - programmatic scripts
  - neural networks

# Strategy ( $\sigma$ )

- A strategy is a *policy* that maps (some) states to actions
- A strategy is an *instance* of a strategy model

# Performance Measure ( $J$ )

- $J_{\mathcal{G}}(\sigma)$ : how *good* a strategy  $\sigma$  is

# Performance Measure ( $J$ )

- $J_{\mathcal{G}}(\sigma)$ : how *good* a strategy  $\sigma$  is
- Understanding the pros/cons of strategies is important

# Performance Measure ( $J$ )

- $J_{\mathcal{G}}(\sigma)$ : how *good* a strategy  $\sigma$  is
- Understanding the pros/cons of strategies is important
- Examples —
  - strategy win rate
  - material advantage (chess)
  - resources harvested (StarCraft II)

# Interpretability Measure ( $\mathcal{I}$ )

- $\mathcal{I}_{\mathcal{G}}(\sigma)$ : how *easily understood* a strategy  $\sigma$  is

# Interpretability Measure ( $\mathcal{I}$ )

- $\mathcal{I}_{\mathcal{G}}(\sigma)$ : how *easily understood* a strategy  $\sigma$  is
- *computational proxies* based on representation,  $\mathcal{I}_{\mathcal{M}}$

# Interpretability Measure ( $\mathcal{I}$ )

- $\mathcal{I}_{\mathcal{G}}(\sigma)$ : how *easily understood* a strategy  $\sigma$  is
- *computational proxies* based on representation,  $\mathcal{I}_{\mathcal{M}}$
- Examples —
  - number of statements (programmatic script)
  - number of nodes (decision tree)

# Interpretability Measure ( $\mathcal{I}$ )

- $\mathcal{I}_{\mathcal{G}}(\sigma)$ : how *easily understood* a strategy  $\sigma$  is
- *computational proxies* based on representation,  $\mathcal{I}_{\mathcal{M}}$
- Examples —
  - number of statements (programmatic script)
  - number of nodes (decision tree)
- Use *readable* representations

# Summary

## RQ1

How do we formally define the problem of Interpretable Strategy Synthesis (ISS)?

# Summary

## RQ1

How do we *formally define* the problem of Interpretable Strategy Synthesis (ISS)?

- Definition of ISS  $\langle \mathcal{G}, \mathcal{M}, \mathcal{J}, \mathcal{I} \rangle$  (Krishnan & Martens, 2022b)

# Introduction

## RQ2

How do we approach the problem of ISS using a *logic-based* strategy model?

# Introduction

## RQ2

How do we approach the problem of ISS using a *logic-based* strategy model?

- Application to *chess* —

# Introduction

## RQ2

How do we approach the problem of ISS using a *logic-based* strategy model?

- Application to *chess* —
  - RQ2(a): Applying the ISS framework using FOL-based strategies to chess (Krishnan & Martens, 2022b)

# Introduction

## RQ2

How do we approach the problem of ISS using a *logic-based* strategy model?

- Application to *chess* —
  - RQ2(a): Applying the ISS framework using FOL-based strategies to chess (Krishnan & Martens, 2022b)
  - RQ2(b): Learning chess strategies using ILP (Krishnan & Martens, 2022a)

# Introduction

## RQ2

How do we approach the problem of ISS using a *logic-based* strategy model?

- Application to *chess* —
  - RQ2(a): Applying the ISS framework using FOL-based strategies to chess (Krishnan & Martens, 2022b)
  - RQ2(b): Learning chess strategies using ILP (Krishnan & Martens, 2022a)
  - RQ2(c): Improving the ILP-based learning method (Krishnan et al., 2024)

# First-Order Logic (FOL)

- FOL is a *formal* language

# First-Order Logic (FOL)

- FOL is a *formal* language
- can represent *relationships* between objects

# First-Order Logic (FOL)

- FOL is a *formal* language
- can represent *relationships* between objects
- E.g., in chess, a king is *in check* if an opponent's piece can capture it

$$\forall k, b \text{ InCheck}(k, b) \iff \exists p \text{ CanMove}(p, \text{Square}(p), \text{Square}(k), b) \wedge p \neq \text{King} \quad (1)$$

# Prolog

- *Declarative* programming language

# Prolog

- *Declarative* programming language
- Allows specifying *facts* and *rules* in FOL

# Prolog

- *Declarative* programming language
- Allows specifying *facts* and *rules* in FOL
- *in check* in Prolog —

```
1 in_check(King , Board) :-  
2     can_move(Piece , Square(Piece) , Square(King) , Board) ,  
3     Piece \= king .
```

# Why use FOL-based Strategies?

# Why use FOL-based Strategies?

- Prior work in *relational reinforcement learning* and *pattern learning* in chess
  - Handle domains with *large state spaces*
  - *Generalize* to similar domains
  - *Interpretability* of learned rules

# Why use FOL-based Strategies?

- Prior work in *relational reinforcement learning* and *pattern learning* in chess
  - Handle domains with *large state spaces*
  - *Generalize* to similar domains
  - *Interpretability* of learned rules
- *Interpretability* of rule-based models (Kliegr et al., 2021; Zhang et al., 2021)

# ISS for Chess

## RQ2(a)

Could we represent known chess tactics as a strategy model for chess and develop metrics to show that they suggest better moves than a random baseline?

# ISS for Chess

## RQ2(a)

Could we represent known chess tactics as a *strategy model* for chess and develop metrics to show that they suggest better moves than a random baseline?

- *Strategy model* for chess

# ISS for Chess

## RQ2(a)

Could we represent known chess tactics as a strategy model for chess and develop *metrics* to show that they suggest better moves than a random baseline?

- Strategy model for chess
- *Performance measure* for chess

# ISS for Chess

## RQ2(a)

Could we represent known chess tactics as a strategy model for chess and develop metrics to *show that they suggest better moves than a random baseline?*

- Strategy model for chess
- Performance measure for chess

# Strategy Model for Chess

## First-Order (FO) Logic Rule

# Strategy Model for Chess

**First-Order (FO) Logic Rule**

**Predicate Vocabulary**

# Strategy Model for Chess

## First-Order (FO) Logic Rule

```
tactic(Position, Move) ←  
    feature_1(…),  
    feature_2(…),  
    :  
    feature_n(…)
```

## Predicate Vocabulary

Our chess strategy model expressed in Prolog pseudocode

# Strategy Model for Chess

## First-Order (FO) Logic Rule

```
tactic(Position, Move) ←  
    feature_1(…),  
    feature_2(…),  
    :  
    feature_n(…)
```

Our chess strategy model expressed in Prolog pseudocode

## Predicate Vocabulary

- **Position** =

- [contents(c2,pawn,white),  
 contents(g8,knight,black),  
 contents(e8,king,black),  
 turn(white),kingside\_castle(white),...]

# Strategy Model for Chess

## First-Order (FO) Logic Rule

```
tactic(Position, Move) ←  
    feature_1(…),  
    feature_2(…),  
    :  
    feature_n(…)
```

Our chess strategy model expressed in Prolog pseudocode

## Predicate Vocabulary

- **Position** =  
[contents(c2,pawn,white),  
 contents(g8,knight,black),  
 contents(e8,king,black),  
  
 turn(white),kingside\_castle(white),...]
- **Move** = [a7,a8,queen]

# Strategy Model for Chess

## First-Order (FO) Logic Rule

```
tactic(Position, Move) ←  
    feature_1(...),  
    feature_2(...),  
    :  
    feature_n(...)
```

Our chess strategy model expressed in Prolog pseudocode

## Predicate Vocabulary

- **Position** = [contents(c2,pawn,white), contents(g8,knight,black), contents(e8,king,black), turn(white), kingside\_castle(white), ...]
- **Move** = [a7, a8, queen]
- **Features** =
  - attacks (Pos, Sq1, Sq2)
  - in\_check (Pos, Side)
  - is\_empty (Pos, Squares)

# Example

```
fork(Position,Move) ←  
    legal_move(Position,Move),  
    move(Move,_,To,_),  
    make_move(Position,Move>NewPosition),  
    can_capture(NewPosition,To,ForkSquare1),  
    can_capture(NewPosition,To,ForkSquare2),  
    different(ForkSquare1,ForkSquare2).
```

An interpretation of the *fork* tactic from the chess literature using our predicate vocabulary.

# Example

```
fork(Position,Move) ←  
    legal_move(Position,Move),  
    move(Move,_,To,_),  
    make_move(Position,Move,NewPosition),  
    can_capture(NewPosition,To,ForkSquare1),  
    can_capture(NewPosition,To,ForkSquare2),  
    different(ForkSquare1,ForkSquare2).
```

An interpretation of the *fork* tactic from the chess literature using our predicate vocabulary.

# ISS for Chess

## RQ2(a)

Could we represent known chess tactics as a strategy model for chess and develop metrics to show that they suggest better moves than a random baseline?

- Strategy model for chess ✓
- Performance measure for chess

# ISS for Chess

## RQ2(a)

Could we represent known chess tactics as a strategy model for chess and develop *metrics* to show that they suggest better moves than a random baseline?

- Strategy model for chess ✓
- *Performance measure* for chess

# Performance Measure

## Divergence

Equation

- How do strategies *differ* in terms of strength? (Guid & Bratko, 2006, 2011)

# Performance Measure

## Divergence

Equation

- How do strategies *differ* in terms of strength? (Guid & Bratko, 2006, 2011)
- High divergence → large difference in playing strength

# Performance Measure

## Divergence

Equation

- How do strategies *differ* in terms of strength? (Guid & Bratko, 2006, 2011)
- High divergence → large difference in playing strength
- Low divergence → small difference in playing strength

# Performance Measure

## Divergence

Equation

- How do strategies *differ* in terms of strength? (Guid & Bratko, 2006, 2011)
- High divergence → large difference in playing strength
- Low divergence → small difference in playing strength
- Playing strength: *perceived evaluation* of moves

# Performance Measure

## Divergence

Equation

- How do strategies *differ* in terms of strength? (Guid & Bratko, 2006, 2011)
- High divergence → large difference in playing strength
- Low divergence → small difference in playing strength
- Playing strength: *perceived evaluation* of moves
- Who is “perceiving”?

# Performance Measure

## Divergence

Equation

- How do strategies *differ* in terms of strength? (Guid & Bratko, 2006, 2011)
- High divergence → large difference in playing strength
- Low divergence → small difference in playing strength
- Playing strength: *perceived evaluation* of moves
- Who is “perceiving”?
  - Chess-playing agents with an *evaluation function* (chess “engines”)

# Performance Measure

## Divergence

Equation

- How do strategies *differ* in terms of strength? (Guid & Bratko, 2006, 2011)
- High divergence → large difference in playing strength
- Low divergence → small difference in playing strength
- Playing strength: *perceived evaluation* of moves
- Who is “perceiving”?
  - Chess-playing agents with an *evaluation function* (chess “engines”)
  - e.g., Stockfish 14 (strong), Maia Chess (McIlroy-Young et al., 2020) (weak, human-like)

# ISS for Chess

## RQ2(a)

Could we represent known chess tactics as a strategy model for chess and develop metrics to show that they suggest better moves than a random baseline?

- Strategy model for chess ✓
- Performance measure for chess ✓

# ISS for Chess

## RQ2(a)

Could we represent known chess tactics as a strategy model for chess and develop metrics to *show that they suggest better moves than a random baseline?*

- Strategy model for chess ✓
- Performance measure for chess ✓

# Evaluation of Known Chess Tactics<sup>1</sup>

---

<sup>1</sup>Krishnan and Martens, 2022b.

# Evaluation of Known Chess Tactics<sup>1</sup>

- PAL (Morales, 1992)  $\xrightarrow{\text{learn}}$  *known* chess patterns (tactics) PAL

---

<sup>1</sup>Krishnan and Martens, 2022b.

# Evaluation of Known Chess Tactics<sup>1</sup>

- PAL (Morales, 1992)  $\xrightarrow{\text{learn}}$  known chess patterns (tactics) PAL
- tactics  $\xrightarrow{\text{translate}}$  chess strategy model

---

<sup>1</sup>Krishnan and Martens, 2022b.

# Evaluation of Known Chess Tactics<sup>1</sup>

- PAL (Morales, 1992)  $\xrightarrow{\text{learn}}$  known chess patterns (tactics) PAL
- tactics  $\xrightarrow{\text{translate}}$  chess strategy model
- Divergence(chess strategies, *human beginner*)

---

<sup>1</sup>Krishnan and Martens, 2022b.

# Evaluation of Known Chess Tactics<sup>1</sup>

- PAL <sub>(Morales, 1992)</sub>  $\xrightarrow{\text{learn}}$  known chess patterns (tactics) PAL
- tactics  $\xrightarrow{\text{translate}}$  chess strategy model
- Divergence(chess strategies, human beginner)
- Divergence(*random baseline*, human beginner)

---

<sup>1</sup>Krishnan and Martens, 2022b.

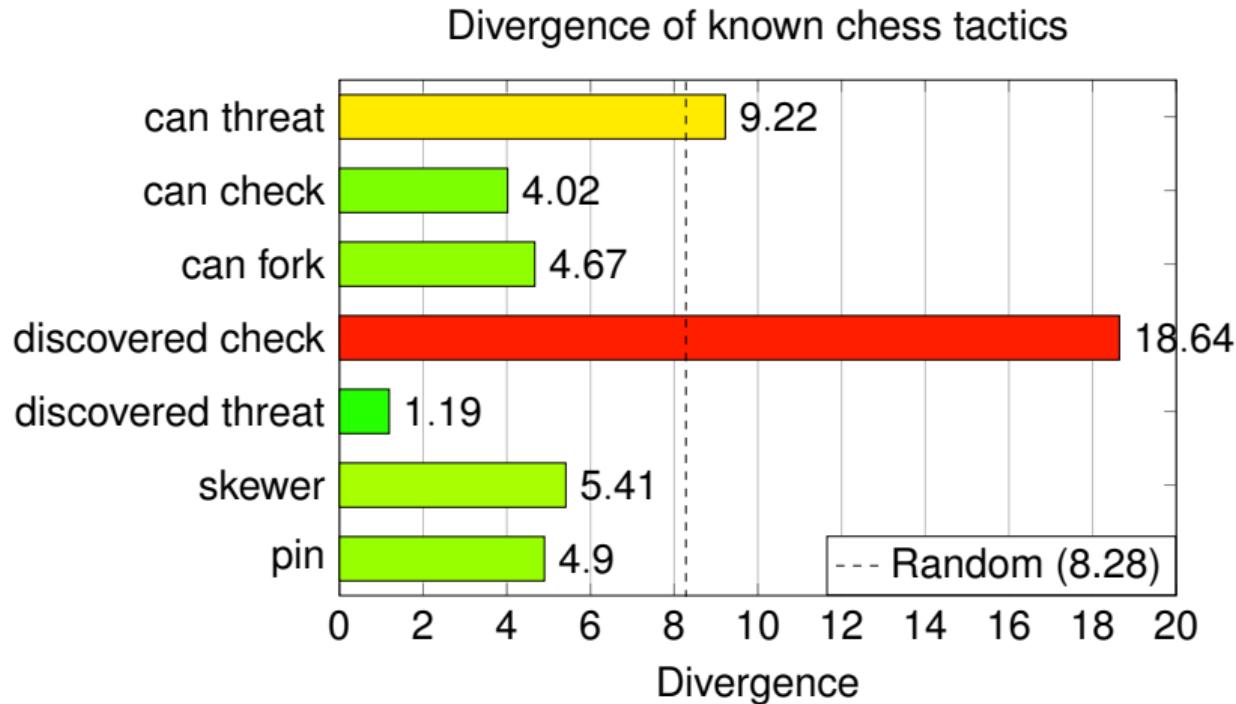
# Evaluation of Known Chess Tactics<sup>1</sup>

- PAL (Morales, 1992)  $\xrightarrow{\text{learn}}$  known chess patterns (tactics) PAL
- tactics  $\xrightarrow{\text{translate}}$  chess strategy model
- Divergence(chess strategies, human beginner)
- Divergence(random baseline, human beginner)
- Using a human-like engine (McIlroy-Young et al., 2020)

---

<sup>1</sup>Krishnan and Martens, 2022b.

# Results



# Learning Chess Strategies

## RQ2(b)

Do the chess strategies *learned* using inductive logic programming outperform a random baseline in how closely their divergence scores approximate a beginner player?

- Strategy model for chess ✓
- Performance measure for chess ✓

# Learning Chess Strategies

## RQ2(b)

Do the chess strategies learned using inductive logic programming outperform a random baseline in how closely their divergence scores approximate a beginner player?

- Strategy model for chess ✓
- Performance measure for chess ✓
- *Learning algorithm* for chess strategies

# Learning Chess Strategies

## RQ2(b)

Do the chess strategies learned using *inductive logic programming* outperform a random baseline in how closely their divergence scores approximate a beginner player?

- Strategy model for chess ✓
- Performance measure for chess ✓
- *Learning algorithm* for chess strategies

# Learning Chess Strategies

## RQ2(b)

Do the chess strategies learned using inductive logic programming  
*outperform a random baseline in how closely their divergence scores approximate a beginner player?*

- Strategy model for chess ✓
- Performance measure for chess ✓
- *Learning algorithm* for chess strategies

# Inductive Logic Programming (ILP)

# Inductive Logic Programming (ILP)

- *symbolic* machine learning technique

# Inductive Logic Programming (ILP)

- *symbolic* machine learning technique
- Learn *classifier* in the form of a FOL rule

# Inductive Logic Programming (ILP)

- *symbolic* machine learning technique
- Learn *classifier* in the form of a FOL rule
- FOL rule expressed using *background knowledge*

# Target Concept

- Want to learn `last` relating a list of characters to the last character in it

$$E^+ = \left\{ \begin{array}{l} \text{last}([m, a, c, h, i, n, e], e). \\ \text{last}([l, e, a, r, n, i, n, g], g). \\ \text{last}([a, l, g, o, r, i, t, h, m], m). \end{array} \right\}$$

$$E^- = \left\{ \begin{array}{l} \text{last}([m, a, c, h, i, n, e], m). \\ \text{last}([m, a, c, h, i, n, e], c). \\ \text{last}([l, e, a, r, n, i, n, g], x). \\ \text{last}([l, e, a, r, n, i, n, g], i). \end{array} \right\}$$

$$B = \left\{ \begin{array}{l} \text{empty}(A) :- \dots \\ \text{head}(A, B) :- \dots \\ \text{tail}(A, B) :- \dots \end{array} \right\}$$

# Possible Hypothesis

$$H = \left\{ \begin{array}{l} \text{last}(A, B) :- \text{head}(A, B), \text{tail}(A, C), \text{empty}(C). \\ \text{last}(A, B) :- \text{tail}(A, C), \text{last}(C, B). \end{array} \right\}$$

# Learning Rule-based Policies

# Learning Rule-based Policies

- Can be learned using ILP

# Learning Rule-based Policies

- Can be learned using ILP
- ISS problem  $\langle \mathcal{G}, \mathcal{M}, J, \mathcal{I} \rangle \xrightarrow{\text{translate}} \text{ILP problem } \langle E^+, E^-, B \rangle$

# Learning Rule-based Policies

- Can be learned using ILP
- ISS problem  $\langle \mathcal{G}, \mathcal{M}, J, \mathcal{I} \rangle \xrightarrow{\text{translate}} \text{ILP problem } \langle E^+, E^-, B \rangle$
- $E^+$ :  $\langle s, \pi(s) \rangle$  for some target policy  $\pi$
- $E^-$ :  $\langle s, a' \rangle$  where  $a' \in \mathcal{A}(s) - \pi(s)$
- $B$ : manual translation of  $\mathcal{G}$  to FOL

# Learning Rule-based Policies

- Can be learned using ILP
- ISS problem  $\langle \mathcal{G}, \mathcal{M}, J, \mathcal{I} \rangle \xrightarrow{\text{translate}} \text{ILP problem } \langle E^+, E^-, B \rangle$
- $E^+$ :  $\langle s, \pi(s) \rangle$  for some target policy  $\pi$
- $E^-$ :  $\langle s, a' \rangle$  where  $a' \in \mathcal{A}(s) - \pi(s)$
- $B$ : manual translation of  $\mathcal{G}$  to FOL
- ILP system( $\langle E^+, E^-, B \rangle$ )  $\xrightarrow{\text{learn}}$  rule-based strategies

# Learning Chess Strategies using ILP<sup>2</sup>

- Use *Popper* (Cropper & Morel, 2021) as ILP system

---

<sup>2</sup>Krishnan and Martens, 2022a.

# Learning Chess Strategies using ILP<sup>2</sup>

- Use *Popper* (Cropper & Morel, 2021) as ILP system
- Modifications —

---

<sup>2</sup>Krishnan and Martens, 2022a.

# Learning Chess Strategies using ILP<sup>2</sup>

- Use *Popper* (Cropper & Morel, 2021) as ILP system
- Modifications —
  - generate only strategies that produce legal moves

---

<sup>2</sup>Krishnan and Martens, 2022a.

# Learning Chess Strategies using ILP<sup>2</sup>

- Use *Popper* (Cropper & Morel, 2021) as ILP system
- Modifications —
  - generate only strategies that produce legal moves
  - prevent generation of strategies that don't match any position in training sets

---

<sup>2</sup>Krishnan and Martens, 2022a.

# Learning Chess Strategies using ILP<sup>2</sup>

- Use *Popper* (Cropper & Morel, 2021) as ILP system
- Modifications —
  - generate only strategies that produce legal moves
  - prevent generation of strategies that don't match any position in training sets
- Evaluate using *divergence* with Stockfish 14 (strong)

---

<sup>2</sup>Krishnan and Martens, 2022a.

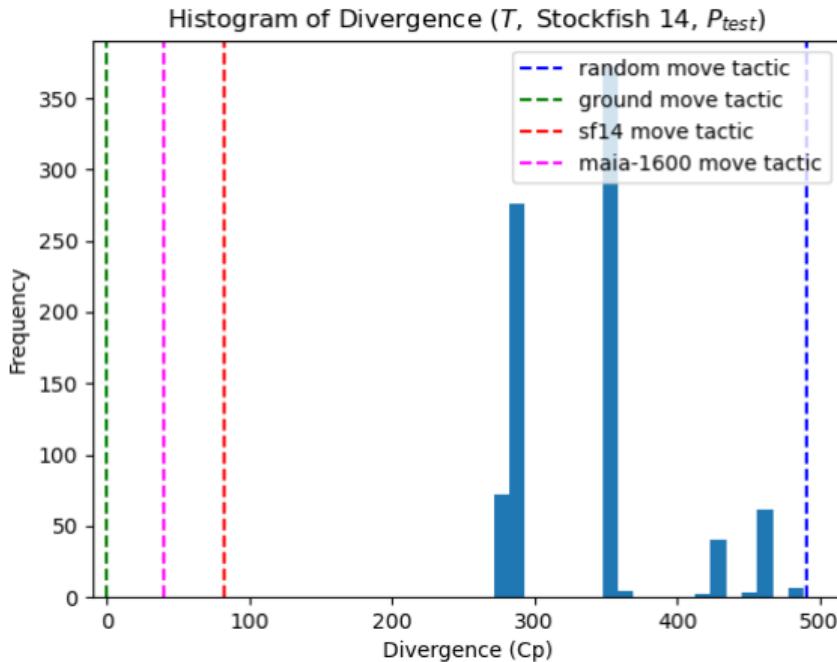
# Learning Chess Strategies using ILP<sup>2</sup>

- Use *Popper* (Cropper & Morel, 2021) as ILP system
- Modifications —
  - generate only strategies that produce legal moves
  - prevent generation of strategies that don't match any position in training sets
- Evaluate using *divergence* with Stockfish 14 (strong)
- Compare to random, strong/human-like engine baselines

---

<sup>2</sup>Krishnan and Martens, 2022a.

# Results



# Improving the ILP Learning Method

## RQ2(c)

Do the chess strategies learned by an ILP system incorporating the changes of the new predicate vocabulary and precision/recall-based constraints produce moves better than those learned by an ILP system without these modifications?

# Improving the ILP Learning Method

## RQ2(c)

Do the chess strategies learned by an ILP system incorporating the changes of the new predicate vocabulary and *precision/recall-based constraints* produce moves better than those learned by an ILP system without these modifications?

- *Constraints* to limit search space of chess strategies

# Improving the ILP Learning Method

## RQ2(c)

Do the chess strategies learned by an ILP system incorporating the changes of the new *predicate vocabulary* and precision/recall-based constraints produce moves better than those learned by an ILP system without these modifications?

- Constraints to limit search space of chess strategies
- *New* predicate vocabulary

# Improvements to ILP for Chess Strategies<sup>3</sup>

- Domain-specific constraints — Constraints

---

<sup>3</sup>Krishnan et al., 2024.

# Improvements to ILP for Chess Strategies<sup>3</sup>

- Domain-specific constraints — Constraints
  - *Limit* search space of chess strategies

---

<sup>3</sup>Krishnan et al., 2024.

# Improvements to ILP for Chess Strategies<sup>3</sup>

- Domain-specific constraints — Constraints
  - *Limit* search space of chess strategies
  - Based on precision and recall of strategy on training set

---

<sup>3</sup>Krishnan et al., 2024.

# Improvements to ILP for Chess Strategies<sup>3</sup>

- Domain-specific constraints — Constraints
  - *Limit* search space of chess strategies
  - Based on precision and recall of strategy on training set
- New predicate vocabulary — Vocabulary

---

<sup>3</sup>Krishnan et al., 2024.

# Improvements to ILP for Chess Strategies<sup>3</sup>

- Domain-specific constraints — Constraints
  - *Limit* search space of chess strategies
  - Based on precision and recall of strategy on training set
- New predicate vocabulary — Vocabulary
  - Rules to express *rare* situations – en passant, promotion

---

<sup>3</sup>Krishnan et al., 2024.

# Improvements to ILP for Chess Strategies<sup>3</sup>

- Domain-specific constraints — Constraints
  - *Limit* search space of chess strategies
  - Based on precision and recall of strategy on training set
- New predicate vocabulary — Vocabulary
  - Rules to express *rare* situations – en passant, promotion
  - Allows *more efficient* search due to introduction of more types

---

<sup>3</sup>Krishnan et al., 2024.

# Improvements to ILP for Chess Strategies<sup>3</sup>

- Domain-specific constraints — Constraints
  - *Limit* search space of chess strategies
  - Based on precision and recall of strategy on training set
- New predicate vocabulary — Vocabulary
  - Rules to express *rare* situations – en passant, promotion
  - Allows *more efficient* search due to introduction of more types
  - More domain-specific state features like *can\_threat* and *xrays*

---

<sup>3</sup>Krishnan et al., 2024.

# Evaluation

- Conduct *ablation study* to measure impact of modifications

# Evaluation

- Conduct *ablation study* to measure impact of modifications
  - Learn strategies using systems with/without constraints, predicate vocabulary

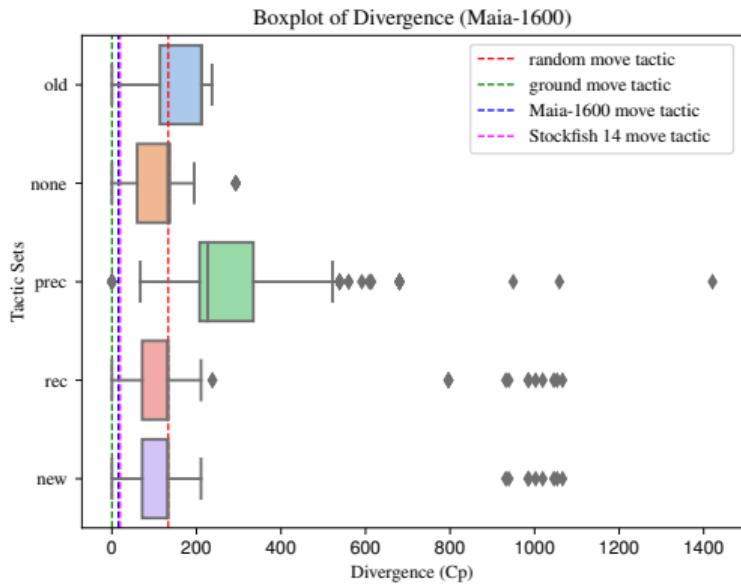
# Evaluation

- Conduct *ablation study* to measure impact of modifications
  - Learn strategies using systems with/without constraints, predicate vocabulary
  - Measure average strategy divergence (using both strong and human-like engines)

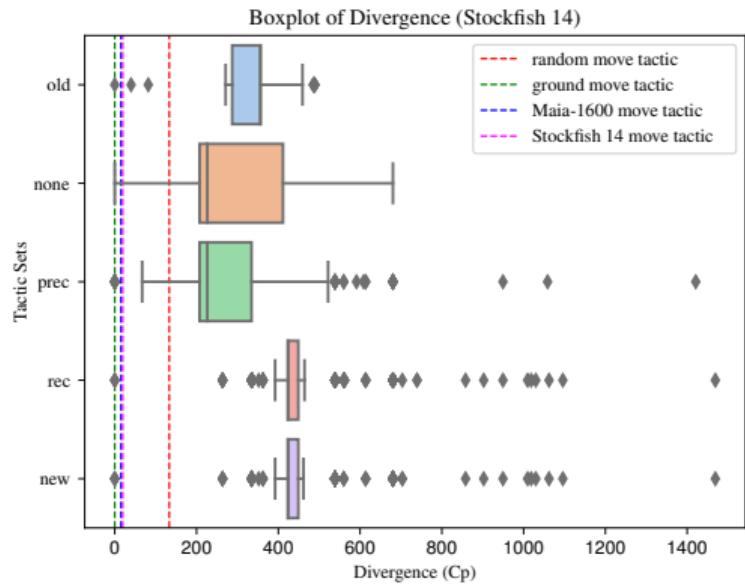
# Evaluation

- Conduct *ablation study* to measure impact of modifications
  - Learn strategies using systems with/without constraints, predicate vocabulary
  - Measure average strategy divergence (using both strong and human-like engines)
  - Test decrease vs. old system using a *one-sided Welch's t-test*

# Results



Boxplot of strategy divergence (evaluated using *human-like* engine) for each system



Boxplot of strategy divergence (evaluated using *strong* engine) for each system

# Summary

## RQ2

How do we approach the problem of ISS using a *logic-based* strategy model?

# Summary

## RQ2

How do we approach the problem of ISS using a *logic-based* strategy model?

- Application to *chess* —

# Summary

## RQ2

How do we approach the problem of ISS using a *logic-based* strategy model?

- Application to *chess* —
  - RQ2(a): Represented chess tactics as FOL rules and showed that known chess tactics resemble human beginner play (Krishnan & Martens, 2022b)

# Summary

## RQ2

How do we approach the problem of ISS using a *logic-based* strategy model?

- Application to *chess* —

- RQ2(a): Represented chess tactics as FOL rules and showed that known chess tactics resemble human beginner play (Krishnan & Martens, 2022b)
- RQ2(b): Learned chess strategies using ILP which learned strategies resembling human beginner playing strength (Krishnan & Martens, 2022a)

# Summary

## RQ2

How do we approach the problem of ISS using a *logic-based* strategy model?

- Application to *chess* —

- RQ2(a): Represented chess tactics as FOL rules and showed that known chess tactics resemble human beginner play (Krishnan & Martens, 2022b)
- RQ2(b): Learned chess strategies using ILP which learned strategies resembling human beginner playing strength (Krishnan & Martens, 2022a)
- RQ2(c): Improving ILP learning using precision/recall constraints (Krishnan et al., 2024)

# Challenges with FOL

Challenges with FOL-based strategies —

# Challenges with FOL

Challenges with FOL-based strategies —

- Knowledge engineering

# Challenges with FOL

Challenges with FOL-based strategies —

- Knowledge engineering
  - Improved predicate vocabulary represented *800+ LoC in Prolog*

# Challenges with FOL

Challenges with FOL-based strategies —

- Knowledge engineering
  - Improved predicate vocabulary represented *800+ LoC in Prolog*
- Performance

# Challenges with FOL

Challenges with FOL-based strategies —

- Knowledge engineering
  - Improved predicate vocabulary represented *800+ LoC in Prolog*
- Performance
  - Playing strength not close to state-of-the-art

# Motivation

Challenges with FOL-based strategies —

- Knowledge engineering
  - Programmatic strategies require *only a DSL* to be defined
- Performance
  - Success of RL methods (DeepMind, 2019; OpenAI et al., 2019; Silver et al., 2018)

# Strategies as Programs

RQ3

How do we approach the problem of ISS using a *programmatic* strategy model?

# Strategies as Programs

## RQ3

How do we approach the problem of ISS using a *programmatic* strategy model?

- RQ3(a): Leveraging reinforcement learning (RL) to learn programmatic strategies (Krishnan et al., 2024)

# Strategies as Programs

## RQ3

How do we approach the problem of ISS using a *programmatic* strategy model?

- RQ3(a): Leveraging reinforcement learning (RL) to learn programmatic strategies (Krishnan et al., 2024)
- RQ3(b): Showing RL is useful for learning programmatic strategies (Krishnan et al., 2024)

# Programmatic Strategy Model

# Programmatic Strategy Model

- Programmatic strategy modeled as *domain-specific language (DSL) script*

# Programmatic Strategy Model

- Programmatic strategy modeled as *domain-specific language (DSL) script*
- DSL defined as a context-free grammar (CFG)  
 $\langle V, \Sigma, R, S \rangle$

$$\begin{aligned}
 P &::= (\epsilon - \mathbf{hd}(h_i)) \\
 I &::= \mathbf{fold}(+, h_i) \\
 D &::= \mathbf{hd}(\mathbf{tl}(h_i)) - \mathbf{hd}(h_i) \\
 C &::= c_1 + c_2 * P + c_3 * I + c_4 * D \\
 B &::= c_0 + c_1 * \mathbf{hd}(h_1) + \cdots + c_k * \mathbf{hd}(h_k) > 0 \mid \\
 &\quad B_1 \text{ or } B_2 \mid B_1 \text{ and } B_2 \\
 E &::= C \mid \text{if } B \text{ then } E_1 \text{ else } E_2.
 \end{aligned}$$

The DSL for expressing PID controllers due to Verma et al. (2018).

# Programmatic Strategy Model

- Strategy = string in the *language* of the DSL

```
if (0.001 - hd(hTrackPos) > 0) and (0.001 + hd(hTrackPos) > 0)
  then 1.96 + 4.92 * (0.44 - hd(hRPM)) + 0.89 * fold(+, hRPM) + 49.79 * (hd(tl(hRPM)) - hd(hRPM))
  else 1.78 + 4.92 * (0.40 - hd(hRPM)) + 0.89 * fold(+, hRPM) + 49.79 * (hd(tl(hRPM)) - hd(hRPM))
```

A programmatic policy (strategy) for a PID controller.

# Learning Programmatic Strategies

- Learning programmatic strategies = program synthesis —

# Learning Programmatic Strategies

- Learning programmatic strategies = program synthesis —
  - based on  $\langle \text{input}, \text{output} \rangle$  pairs (X. Chen et al., 2019; Lázaro-Gredilla et al., 2019; Yang et al., 2021)

# Learning Programmatic Strategies

- Learning programmatic strategies = program synthesis —
  - based on  $\langle \text{input}, \text{output} \rangle$  pairs (X. Chen et al., 2019; Lázaro-Gredilla et al., 2019; Yang et al., 2021)
  - demonstrations (Burke et al., 2019; Sun et al., 2018; Xu et al., 2018)

# Learning Programmatic Strategies

- Learning programmatic strategies = program synthesis —
  - based on  $\langle \text{input}, \text{output} \rangle$  pairs (X. Chen et al., 2019; Lázaro-Gredilla et al., 2019; Yang et al., 2021)
  - demonstrations (Burke et al., 2019; Sun et al., 2018; Xu et al., 2018)

## RQ3(a)

How do I formally map the problem of ISS to that of programmatic policy synthesis?

# ISS as RL-based Program Synthesis

- Idea: learn a policy to *derive* programs from a CFG

# ISS as RL-based Program Synthesis

- Idea: learn a policy to *derive* programs from a CFG

$$S \rightarrow ASB|c$$

$$A \rightarrow aA|\epsilon$$

$$B \rightarrow bB|\epsilon$$

# ISS as RL-based Program Synthesis

- Idea: learn a policy to *derive* programs from a CFG

$$S \rightarrow ASB|c$$

$$A \rightarrow aA|\epsilon$$

$$B \rightarrow bB|\epsilon$$

$$S \Rightarrow ASB \Rightarrow SB$$

$$\Rightarrow S \Rightarrow c$$

# Program Synthesis MDP<sup>4</sup>

Define a program synthesis MDP  $\mathcal{M}_{\text{syn}}$ , where —

- States: partially expanded programs
- Actions: rule applications to valid non-terminals
- Transitions: effects of applying a rule
- Rewards: return from executing program in the target MDP

---

<sup>4</sup>Krishnan et al., 2024.

# Applying RL to Learn Programmatic Strategies

## RQ3b

Could a *decision transformer* model synthesize better strategies for a toy programming environment when compared to existing approaches?

# Learning Programmatic Strategies using Decision Transformers<sup>5</sup>

---

<sup>5</sup>Krishnan et al., 2024.

# Learning Programmatic Strategies using Decision Transformers<sup>5</sup>

- ISS for programmatic strategies —

---

<sup>5</sup>Krishnan et al., 2024.

# Learning Programmatic Strategies using Decision Transformers<sup>5</sup>

- ISS for programmatic strategies —
  - Strategy model = DSL

---

<sup>5</sup>Krishnan et al., 2024.

# Learning Programmatic Strategies using Decision Transformers<sup>5</sup>

- ISS for programmatic strategies —
  - Strategy model = DSL
  - Performance measure = reward from executing strategy in target environment

---

<sup>5</sup>Krishnan et al., 2024.

# Learning Programmatic Strategies using Decision Transformers<sup>5</sup>

- ISS for programmatic strategies —
  - Strategy model = DSL
  - Performance measure = reward from executing strategy in target environment
  - Learning method: Program synthesis MDP

---

<sup>5</sup>Krishnan et al., 2024.

# Decision Transformer (DT)<sup>6</sup>

- Application of a transformer model (Vaswani et al., 2017) to RL
- Models the RL problem as a sequence learning problem
- Given an *trajectory*  $\tau$ , predict the next action

$$\tau = (\hat{R}_1, s_1, a_1, \hat{R}_2, s_2, a_2, \dots, \hat{R}_T, s_T, a_T) \quad (2)$$

---

<sup>6</sup>Y. Chen, Wang, Bastani, Dillig, and Feng, 2020.

# Methodology

Modifications to original decision transformer (DT) architecture —

# Methodology

Modifications to original decision transformer (DT) architecture —

- *Action Masking* to deal with invalid actions (Huang & Ontañón, 2022)

# Methodology

Modifications to original decision transformer (DT) architecture —

- *Action Masking* to deal with invalid actions (Huang & Ontañón, 2022)
- *Cross-entropy loss* and *sampling from action distribution* to deal with discrete action spaces

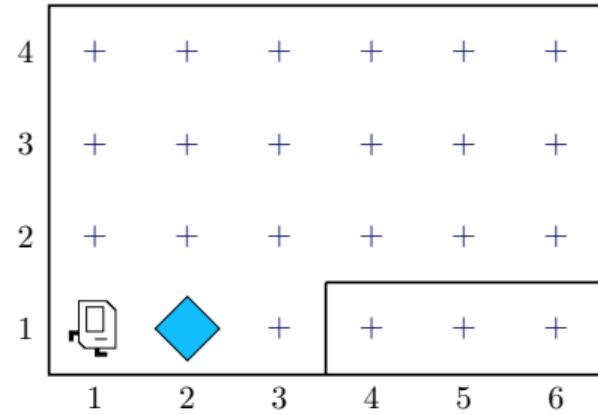
# Methodology

Modifications to original decision transformer (DT) architecture —

- *Action Masking* to deal with invalid actions (Huang & Ontañón, 2022)
- *Cross-entropy loss* and *sampling from action distribution* to deal with discrete action spaces
- *GRU-based state embedding* to better encode state information

# Karel Domain

- Simple programming language and environment for teaching
- Discrete action space
- Use task definitions from Trivedi, Zhang, Sun, and Lim (2021)



A sample Karel world

# Training & Evaluation

- Randomly generated program dataset

# Training & Evaluation

- Randomly generated program dataset
- Programs  $\xrightarrow{\text{convert}}$  trajectories in  $\mathcal{M}_{\text{syn}}$

# Training & Evaluation

- Randomly generated program dataset
- Programs  $\xrightarrow{\text{convert}}$  trajectories in  $\mathcal{M}_{\text{syn}}$
- Train modified DT on generated trajectories

# Training & Evaluation

- Randomly generated program dataset
- Programs  $\xrightarrow{\text{convert}}$  trajectories in  $\mathcal{M}_{\text{syn}}$
- Train modified DT on generated trajectories
- Parameterize top- $k\%$  of the programs based on reward to train on (prioritized experience replay)

# Training & Evaluation

- Randomly generated program dataset
- Programs  $\xrightarrow{\text{convert}}$  trajectories in  $\mathcal{M}_{\text{syn}}$
- Train modified DT on generated trajectories
- Parameterize top- $k\%$  of the programs based on reward to train on (prioritized experience replay)
- Compare performance with LEAPS (Trivedi et al., 2021)

# Sample Program

```
1 DEF run m(
2     WHILE c( noMarkersPresent c) w(
3         turnRight
4         move
5     w)
6     IF c( rightIsClear c) i(
7         WHILE c( frontIsClear c) w(
8             move
9         w)
0     i )
1 m)
```

A program learned by the DT

# Program Description

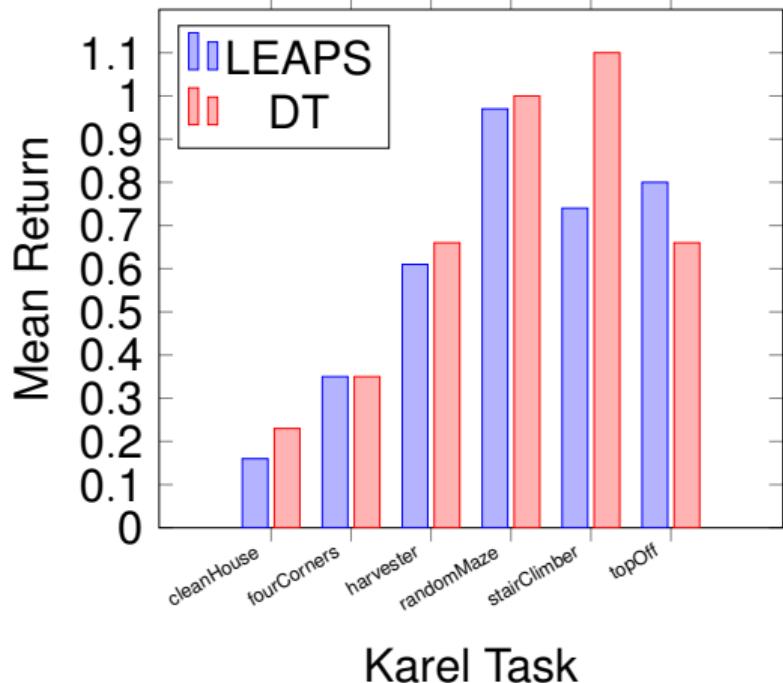
Strategy description as generated by Claude 3.5 Sonnet (Anthropic, 2024)

## Claude 3.5 Sonnet Description

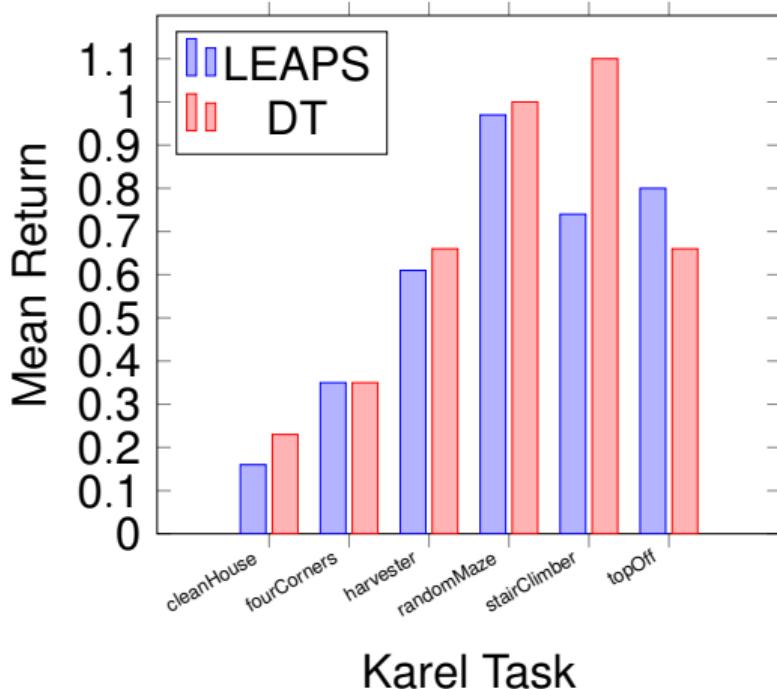
In summary, the strategy is:

- ➊ Climb the stairs by repeatedly turning right and moving forward until a marker is found.
- ➋ Once at the top (indicated by the marker), if there's clear space to the right, move forward along the top of the stairs until reaching an obstacle.

# Results



# Results



Task	Unique Programs	
	LEAPS	DT
cleanHouse	3627	59
fourCorners	9872	55
harvester	11708	28
randomMaze	295	63
stairClimber	298	49
topOff	30278	63

**Table:** Number of unique programs explored by LEAPS and the DT during their search (sampling) phases.

# Sample Efficiency

- Search
  - LEAPS uses CEM search
  - DT uses random sampling
  - DT explores orders of magnitude fewer programs

# Sample Efficiency

- Search
  - LEAPS uses CEM search
  - DT uses random sampling
  - DT explores orders of magnitude fewer programs
- Training
  - Optimal top- $k\%$  = 0.1%, implies DT trained on 500 programs
  - LEAPS trained on 35,000 programs

# Sample Efficiency

- Search
  - LEAPS uses CEM search
  - DT uses random sampling
  - DT explores orders of magnitude fewer programs
- Training
  - Optimal  $\text{top-}k\% = 0.1\%$ , implies DT trained on 500 programs
  - LEAPS trained on 35,000 programs
- Both achieve comparable performance on almost all tasks

# Summary

## RQ3

How do we approach the problem of ISS using a *programmatic* strategy model?

# Summary

## RQ3

How do we approach the problem of ISS using a *programmatic* strategy model?

- RQ3(a): Program synthesis MDP (Krishnan et al., 2024)

# Summary

## RQ3

How do we approach the problem of ISS using a *programmatic* strategy model?

- RQ3(a): Program synthesis MDP (Krishnan et al., 2024)
- RQ3(b): Decision transformer matches SOTA performance with less training data and search (Krishnan et al., 2024)

# Conclusion

## Thesis Statement

A computational model of a game strategy, along with a synthesis method, could meet the goals of discovering high-quality, interpretable strategies and impact the fields of competitive gaming and explainable AI.

## Contributions —

# Conclusion

## Thesis Statement

A *computational model* of a game strategy, along with a synthesis method, could meet the goals of discovering *high-quality*, *interpretable* strategies and impact the fields of competitive gaming and explainable AI.

## Contributions —

- RQ1: Defined a *framework* for ISS (Krishnan & Martens, 2022b)

# Conclusion

## Thesis Statement

A computational model of a game strategy, along with a synthesis method, could meet the goals of discovering high-quality, interpretable strategies and impact the fields of *competitive gaming* and explainable AI.

## Contributions —

- RQ2: Applied *FOL-based* strategy model to solve ISS for chess

# Conclusion

## Thesis Statement

A *computational model* of a game strategy, along with a synthesis method, could meet the goals of discovering *high-quality*, *interpretable* strategies and impact the fields of *competitive gaming* and explainable AI.

## Contributions —

- RQ2: Applied *FOL-based* strategy model to solve ISS for chess
  - RQ2(a): Represented chess tactics as FOL rules (Krishnan & Martens, 2022b)

# Conclusion

## Thesis Statement

A computational model of a game strategy, along with a *synthesis method*, could meet the goals of discovering high-quality, interpretable strategies and impact the fields of *competitive gaming* and explainable AI.

## Contributions —

- RQ2: Applied *FOL-based* strategy model to solve ISS for chess
  - RQ2(a): Represented chess tactics as FOL rules (Krishnan & Martens, 2022b)
  - RQ2(b): Learned chess strategies using ILP (Krishnan & Martens, 2022a)

# Conclusion

## Thesis Statement

A computational model of a game strategy, along with a *synthesis method*, could meet the goals of discovering *high-quality*, interpretable strategies and impact the fields of *competitive gaming* and explainable AI.

## Contributions —

- RQ2: Applied *FOL-based* strategy model to solve ISS for chess
  - RQ2(a): Represented chess tactics as FOL rules (Krishnan & Martens, 2022b)
  - RQ2(b): Learned chess strategies using ILP (Krishnan & Martens, 2022a)
  - RQ2(c): Improving ILP learning (Krishnan et al., 2024)

# Conclusion

## Thesis Statement

A *computational model* of a game strategy, along with a *synthesis method*, could meet the goals of discovering *high-quality*, interpretable strategies and impact the fields of competitive gaming and *explainable AI*.

## Contributions —

- RQ3: Applied *programmatic* strategy model to solve ISS using RL

# Conclusion

## Thesis Statement

A computational model of a game strategy, along with a *synthesis method*, could meet the goals of discovering *high-quality*, *interpretable* strategies and impact the fields of competitive gaming and *explainable AI*.

## Contributions —

- RQ3: Applied *programmatic* strategy model to solve ISS using RL
  - RQ3(a): Program synthesis MDP (Krishnan et al., 2024)

# Conclusion

## Thesis Statement

A computational model of a game strategy, along with a *synthesis method*, could meet the goals of discovering *high-quality*, interpretable strategies and impact the fields of competitive gaming and *explainable AI*.

## Contributions —

- RQ3: Applied *programmatic* strategy model to solve ISS using RL
  - RQ3(a): Program synthesis MDP (Krishnan et al., 2024)
  - RQ3(b): Decision transformer matches SOTA performance with less training data and search (Krishnan et al., 2024)

# Publications

- Krishnan, Abhijeet, Colin M. Potts, Arnav Jhala, Harshad Khadilkar, Shirish Karande and Chris Martens. "Learning Explainable Representations of Complex Game-playing Strategies." *Proceedings of the Eleventh Annual Conference on Advances in Cognitive Systems*. 2024.
- Villalobos-Arias, Leonardo, Derek Martin, Abhijeet Krishnan, Madeleine Gagné, Colin M. Potts and Arnav Jhala. "Modeling Risk in Reinforcement Learning: A Literature Mapping." *arXiv preprint arXiv:2312.05231*. 2023.
- Krishnan, Abhijeet and Chris Martens. "Synthesizing Chess Tactics from Player Games." In *Workshop on Artificial Intelligence for Strategy Games (SG) and Esports Analytics (EA), 18th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. 2022.
- Krishnan, Abhijeet and Chris Martens. "Towards the Automatic Synthesis of Interpretable Chess Tactics." In *Explainable Agency in Artificial Intelligence Workshop, 36th AAAI Conference on Artificial Intelligence*. 2022.
- Krishnan, Abhijeet, Aaron Williams, and Chris Martens. "Towards Action Model Learning for Player Modeling." *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. Vol. 16. No. 1. 2020.
- Krishnan, Abhijeet and Chris Martens. "Rule-based Cognitive Modeling via Human-Computer Interaction." Poster presented at: *5th LAS Research Symposium*; 2019 Dec 10; Raleigh, NC.

*Thank You!*

# Questions?

# References I

- Anthropic. (2024). The claude 3.5 sonnet model card addendum.  
Retrieved from [https://www-cdn.anthropic.com/fed9cc193a14b84131812372d8d5857f8f304c52/Model\\_Card\\_Claude\\_3\\_Addendum.pdf](https://www-cdn.anthropic.com/fed9cc193a14b84131812372d8d5857f8f304c52/Model_Card_Claude_3_Addendum.pdf)
- Beau, P., & Bakkes, S. (2016, September). Automated game balancing of asymmetric video games. In *2016 ieee conference on computational intelligence and games (cig)* (pp. 1–8). doi:10.1109/CIG.2016.7860432

# References II

- Bergdahl, J., Gordillo, C., Tollmar, K., & Gisslén, L. (2020, August). Augmenting automated game testing with deep reinforcement learning. In *2020 ieee conference on games (cog)* (pp. 600–603). doi:10.1109/CoG47356.2020.9231552
- Berliner, H. J. (1975). *A representation and some mechanisms for a problem solving chess program*. Carnegie-Mellon Univ Pittsburgh PA Dept of Computer Science.

# References III

- Boyan, A., McGloin, R., & Wasserman, J. A. (2018). Model matching theory: A framework for examining the alignment between game mechanics and mental models. *Media and Communication*, 6(2), 126–136.  
doi:<https://doi.org/10.17645/mac.v6i2.1326>
- Boyan, A., & Sherry, J. L. (2011). The challenge in creating games for education: Aligning mental models with game models. *Child Development Perspectives*, 5(2), 82–87.  
doi:10.1111/j.1750-8606.2011.00160.x

# References IV

- Bramer, M. A. (1977). *Representation of knowledge for chess endgames towards a self-improving system.* (Doctoral dissertation, Open University (United Kingdom)).
- Bratko, I. (1982). Knowledge-based problem-solving in al3. *Machine intelligence*, 10, 73–100.
- Burke, M., Penkov, S. V., & Ramamoorthy, S. (2019, June). From explanation to synthesis: Compositional program induction for learning from demonstration. In *Robotics: Science and systems XV*. doi:10.15607/rss.2019.xv.015

# References V

- Butler, E., Torlak, E., & Popović, Z. (2017). Synthesizing interpretable strategies for solving puzzle games. In *Proceedings of the 12th international conference on the foundations of digital games*. doi:10.1145/3102071.3102084
- Canaan, R., Shen, H., Torrado, R., Togelius, J., Nealen, A., & Menzel, S. (2018). Evolving agents for the hanabi 2018 cig competition. In *2018 ieee conference on computational intelligence and games (cig)* (pp. 1–8). doi:10.1109/CIG.2018.8490449

# References VI

- Chen, X., Liu, C., & Song, D. (2019). Execution-guided neural program synthesis. In *International conference on learning representations*. Retrieved from <https://openreview.net/forum?id=H1gfOiAqYm>
- Chen, Y., Wang, C., Bastani, O., Dillig, I., & Feng, Y. (2020). Program Synthesis Using Deduction-Guided Reinforcement Learning. In S. K. Lahiri & C. Wang (Eds.), *Computer Aided Verification* (pp. 587–610). doi:10.1007/978-3-030-53291-8\_30

# References VII

- Chitayat, A. P., Block, F., Walker, J., & Drachen, A. (2023). Beyond the meta: Leveraging game design parameters for patch-agnostic esport analitics. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 19(11), 116–125. doi:10.1609/aiide.v19i1.27507
- Clark, N., Macdonald, B., & Kloo, I. (2020). A bayesian adjusted plus-minus analysis for the esport dota 2. *Journal of Quantitative Analysis in Sports*, 16(4), 325–341. doi:10.1515/jqas-2019-0103

# References VIII

Cropper, A., & Morel, R. (2021). Learning programs by learning from failures. *Machine Learning*, 110(4), 801–856.

DeepMind. (2019, January). Alphastar: Mastering the real-time strategy game starcraft ii. Retrieved September 21, 2022, from <https://www.deepmind.com/blog/alphastar-mastering-the-real-time-strategy-game-starcraft-ii>

# References IX

- de Freitas, J. M., de Souza, F. R., & Bernardino, H. S. (2018). Evolving controllers for mario ai using grammar-based genetic programming. In *2018 ieee congress on evolutionary computation (cec)* (pp. 1–8). doi:10.1109/CEC.2018.8477698
- DeLaurentis, D. A., Panchal, J. H., Raz, A. K., Balasubramani, P., Maheshwari, A., Dachowicz, A., & Mall, K. (2021, August). Toward automated game balance: A systematic engineering design approach. In *2021 ieee conference on games (cog)* (pp. 1–8). doi:10.1109/CoG52621.2021.9619032

# References X

- de Mesentier Silva, F., Isaksen, A., Togelius, J., & Nealen, A. (2016). Generating heuristics for novice players. In *2016 ieee conference on computational intelligence and games (cig)* (pp. 1–8). IEEE.
- Evans, R., & Grefenstette, E. (2018). Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research*, *61*, 1–64.

# References XI

Frommel, J., Gugenheimer, J., & Rogers, K. (2019). Opportunities and challenges of using game video stream data for games research. In *All the (world wide) webs a stage: A workshop on live streaming, workshop at chi 2019.*

Gebser, M., Harrison, A., Kaminski, R., Lifschitz, V., & Schaub, T. (2015). Abstract gringo. *Theory and Practice of Logic Programming*, 15(4-5), 449–463.

# References XII

- Gelfond, M., & Lifschitz, V. (1988). The stable model semantics for logic programming.. In *Iclp/slp* (Vol. 88, pp. 1070–1080). Cambridge, MA.
- Gobet, F., & Jansen, P. J. (2006). Training in chess: A scientific approach. *Education and chess*.
- Guid, M., & Bratko, I. (2006). Computer analysis of world chess champions. *ICGA journal*, 29(2), 65–73.

# References XIII

- Guid, M., & Bratko, I. (2011). Using heuristic-search based engines for estimating human skill at chess. *ICGA journal*, 34(2), 71–81.
- Hodge, V. J., Devlin, S., Sephton, N., Block, F., Cowling, P. I., & Drachen, A. (2021). Win prediction in multiplayer esports: Live professional match prediction. *IEEE Transactions on Games*, 13(4), 368–379. doi:10.1109/TG.2019.2948469

# References XIV

- Huang, S., & Ontañón, S. (2022). A Closer Look at Invalid Action Masking in Policy Gradient Algorithms. *The International FLAIRS Conference Proceedings, 35*. arXiv:2006.14171 [cs, stat]. doi:10.32473/flairs.v35i.130584
- Huberman, B. J. (1968, July). *A program to play chess end games.* (Doctoral dissertation, Department of Computer Science, Stanford University.).

# References XV

- Jennings-Teats, M., Smith, G., & Wardrip-Fruin, N. (2010). Polymorph: Dynamic difficulty adjustment through level generation. In *Proceedings of the 2010 workshop on procedural content generation in games* (pp. 1–4).
- Kliegr, T., Bahník, Š., & Fürnkranz, J. (2021). A review of possible effects of cognitive biases on interpretation of rule-based machine learning models. *Artificial Intelligence*, 295, 103458. doi:<https://doi.org/10.1016/j.artint.2021.103458>

# References XVI

Krishnan, A., & Martens, C. (2022a, October). Synthesizing interpretable chess tactics from player games. In *Proceedings of the workshop on artificial intelligence for strategy games (sg) and esports analytics (ea), 18th aaai conference on artificial intelligence and interactive digital entertainment*, American Association for Artificial Intelligence.

# References XVII

Krishnan, A., & Martens, C. (2022b, March). Towards the automatic synthesis of interpretable chess tactics. In *Proceedings of the explainable agency in artificial intelligence workshop, 36th aaai conference on artificial intelligence* (pp. 91–97). American Association of Artificial Intelligence.

# References XVIII

- Krishnan, A., Potts, C. M., Jhala, A., Khadilkar, H., Karande, S., & Martens, C. (2024, June). Learning explainable representations of complex game-playing strategies. In *Proceedings of the eleventh annual conference on advances in cognitive systems*. (to appear).
- Lázaro-Gredilla, M., Lin, D., Guntupalli, J. S., & George, D. (2019). Beyond imitation: Zero-shot task transfer on robots by learning concepts as cognitive programs. *Science Robotics*, 4(26), eaav3150. doi:10.1126/scirobotics.aav3150

# References XIX

- Mariño, J. R. H., Moraes, R. O., Oliveira, T. C., Toledo, C., & Lelis, L. H. S. (2021). Programmatic strategies for real-time strategy games. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(1), 381–389.  
doi:10.1609/aaai.v35i1.16114
- Mariño, J. R., & Toledo, C. F. (2022). Evolving interpretable strategies for zero-sum games. *Applied Soft Computing*, 122, 108860.

# References XX

- Maymin, P. Z. (2021). Smart kills and worthless deaths: Esports analytics for league of legends. *Journal of Quantitative Analysis in Sports*, 17(1), 11–27. doi:10.1515/jqas-2019-0096
- McIlroy-Young, R., Sen, S., Kleinberg, J., & Anderson, A. (2020). Aligning superhuman ai with human behavior: Chess as a model system. In *Proceedings of the 26th acm sigkdd international conference on knowledge discovery & data mining* (pp. 1677–1687). doi:10.1145/3394486.3403219

# References XXI

Medeiros, L. C., Aleixo, D. S., & Lelis, L. H. S. (2022). What can we learn even from the weakest? learning sketches for programmatic strategies. (arXiv:2203.11912).

arXiv:2203.11912 [cs]. Retrieved from  
<http://arxiv.org/abs/2203.11912>

Morales, E. (1992). *First order induction of patterns in chess*. (Doctoral dissertation, PhD thesis, The Turing Institute-University of Strathclyde).

# References XXII

Mousavinasab, E., Zarifsanaiey, N., R. Niakan Kalhor, S., Rakhshan, M., Keikha, L., & Ghazi Saeedi, M. (2021). Intelligent tutoring systems: A systematic review of characteristics, applications, and evaluation methods. *Interactive Learning Environments*, 29(1), 142–163. doi:10.1080/10494820.2018.1558257

# References XXIII

- Nagel, G. L. (2017, January). *Use of eye tracking for esports analytics in a moba game*. (Master thesis, The University of Bergen). Accepted: 2017-05-18T12:37:39Z. Retrieved from <https://bora.uib.no/bora-xmlui/handle/1956/15875>
- Novak, A. R., Bennett, K. J., Pluss, M. A., & Fransen, J. (2020). Performance analysis in esports: Modelling performance at the 2018 league of legends world championship. *International Journal of Sports Science & Coaching*, 15(56), 809–817. doi:10.1177/1747954120932853

# References XXIV

- OpenAI, : Berner, C., Brockman, G., Chan, B., Cheung, V., ...  
Zhang, S. (2019). Dota 2 with large scale deep reinforcement learning. arXiv: 1912.06680 [cs.LG]
- Paredes-Olay, C., Abad, M. J. F., Gámez, M., & Rosas, J. M. (2002). Transfer of control between causal predictive judgments and instrumental responding. *Animal Learning & Behavior*, 30(3), 239–248. doi:10.3758/BF03192833

# References XXV

Pfau, J., Smeddinck, J. D., & Malaka, R. (2017). Automated game testing with icarus: Intelligent completion of adventure riddles via unsupervised solving. In *Extended abstracts publication of the annual symposium on computer-human interaction in play* (pp. 153–164). doi:10.1145/3130859.3131439

Pitrat, J. (1977). A chess combination program which uses plans. *Artificial Intelligence*, 8(3), 275–321.

# References XXVI

- Qiu, W., & Zhu, H. (2022). Programmatic reinforcement learning without oracles. In *International conference on learning representations*. Retrieved from <https://openreview.net/forum?id=6Tk2noBdvxt>
- Schubert, M., Drachen, A., & Mahlmann, T. (2016). Esports analytics through encounter detection. In *Proceedings of the mit sloan sports analytics conference* (Vol. 1, p. 2016). MIT Sloan Boston, MA.

# References XXVII

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., ... Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419), 1140–1144. Publisher: American Association for the Advancement of Science. doi:10.1126/science.aar6404

# References XXVIII

- Smith, A. M., Lewis, C., Hullet, K., Smith, G., & Sullivan, A. (2011). An inclusive view of player modeling. In *Proceedings of the 6th international conference on foundations of digital games* (pp. 301–303).
- Spronck, P., Sprinkhuizen-Kuyper, I., & Postma, E. (2004). Online adaptation of game opponent ai with dynamic scripting. *International Journal of Intelligent Games and Simulation*, 3(1), 45–53.

# References XXIX

Stepanov, A., Lange, A., Khromov, N., Korotin, A., Burnaev, E., & Somov, A. (2019, July). Sensors and game synchronization for data analysis in esports. In *2019 ieee 17th international conference on industrial informatics (indin)* (Vol. 1, pp. 933–938). doi:10.1109/INDIN41052.2019.8972249

# References XXX

- Sun, S.-H., Noh, H., Somasundaram, S., & Lim, J. (2018, July). Neural program synthesis from diverse demonstration videos. In J. Dy & A. Krause (Eds.), *Proceedings of the 35th international conference on machine learning* (Vol. 80, pp. 4790–4799). PMLR. Retrieved from <https://proceedings.mlr.press/v80/sun18a.html>
- Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

# References XXXI

Szabo, A. (1984). *Computer chess tactics and strategy*.  
(Doctoral dissertation, University of British Columbia).  
doi:<http://dx.doi.org/10.14288/1.0051870>

# References XXXII

Trivedi, D., Zhang, J., Sun, S.-H., & Lim, J. J. (2021). Learning to Synthesize Programs as Interpretable and Generalizable Policies. In *Advances in Neural Information Processing Systems* (Vol. 34, pp. 25146–25163). Curran Associates, Inc. Retrieved June 7, 2023, from <https://proceedings.neurips.cc/paper/2021/hash/d37124c4c79f357cb02c655671a432fa-Abstract.html>

# References XXXIII

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 30), Curran Associates, Inc. Retrieved from [https://proceedings.neurips.cc/paper\\_files/paper/2017/file/3f5ee243547dee91fb053c1c4a845aa-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fb053c1c4a845aa-Paper.pdf)

# References XXXIV

Verma, A., Murali, V., Singh, R., Kohli, P., & Chaudhuri, S. (2018, July). Programmatically interpretable reinforcement learning. In J. Dy & A. Krause (Eds.), *Proceedings of the 35th international conference on machine learning* (Vol. 80, pp. 5045–5054). PMLR. Retrieved from <https://proceedings.mlr.press/v80/verma18a.html>

# References XXXV

- Weintrop, D., & Wilensky, U. (2013). Know your enemy: Learning from in-game opponents. In *Proceedings of the 12th international conference on interaction design and children* (pp. 408–411). doi:10.1145/2485760.2485789
- Wilkins, D. E. (1979). *Using patterns and plans to solve problems and control search*. Stanford University.
- Xenopoulos, P., Rulff, J., & Silva, C. (2022). Ggviz: Accelerating large-scale esports game analysis. *Proc. ACM Hum.-Comput. Interact.*, 6(CHI PLAY), 238:1–238:22. doi:10.1145/3549501

# References XXXVI

Xu, D., Nair, S., Zhu, Y., Gao, J., Garg, A., Fei-Fei, L., & Savarese, S. (2018). Neural task programming: Learning to generalize across hierarchical tasks. In *2018 ieee international conference on robotics and automation (icra)* (pp. 3795–3802). doi:10.1109/ICRA.2018.8460689

# References XXXVII

- Xue, S., Wu, M., Kolen, J., Aghdaie, N., & Zaman, K. A. (2017). Dynamic difficulty adjustment for maximized engagement in digital games. In *Proceedings of the 26th international conference on world wide web companion* (pp. 465–471). doi:10.1145/3041021.3054170

# References XXXVIII

- Yang, Y. D., Inala, J. P., Bastani, O., Pu, Y., Solar-Lezama, A., & Rinard, M. (2021, November). Program Synthesis Guided Reinforcement Learning for Partially Observed Environments. arXiv:2102.11137 [cs]. arXiv. Retrieved June 28, 2023, from <http://arxiv.org/abs/2102.11137>
- Zhang, Y., Tio, P., Leonardis, A., & Tang, K. (2021). A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 5(5), 726–742. doi:10.1109/TETCI.2021.3100641

# References XXXIX

Zheng, Y., Xie, X., Su, T., Ma, L., Hao, J., Meng, Z., ... Fan, C. (2019, November). Wuji: Automatic online combat game testing using evolutionary deep reinforcement learning. In *2019 34th ieee/acm international conference on automated software engineering (ase)* (pp. 772–784). doi:10.1109/ASE.2019.00077

# Interpretable Strategy Synthesis (ISS)

## Definition (ISS)

Given a —

- Game environment  $\mathcal{G}$  (i.e., an MDP  $\langle \mathcal{S}, \mathcal{A}(s), \mathcal{P}, \mathcal{R}, \gamma \rangle$ )
- Strategy model  $\mathcal{M}$
- Performance measure  $J: \mathcal{M} \times \mathcal{G} \rightarrow \mathbb{R}$
- Interpretability measure  $\mathcal{I}: \mathcal{M} \times \mathcal{G} \rightarrow \mathbb{R}$

The problem of ISS is to find a strategy  $\sigma^*$  s.t. —

$$\sigma^* \doteq \arg \max_{\sigma} J(\sigma, \mathcal{G}) \mathcal{I}(\sigma, \mathcal{G}), \sigma \in \mathcal{M}$$

# Strategy Model ( $\mathcal{M}$ )

## Definition (strategy model)

A strategy model ( $\mathcal{M}$ ) is a function  $\mathcal{M}: \mathcal{S} \times \Theta \rightarrow \mathcal{A}$ , where —

- $\mathcal{S}$  is the state space of  $\mathcal{G}$ ,
- $\Theta$  is the associated parameter space of  $\mathcal{M}$ , and
- $\mathcal{A}$  is the action space of  $\mathcal{G}$

◀ Return

# Strategy ( $\sigma$ )

## Definition (strategy)

A strategy ( $\sigma$ ) for  $\mathcal{G}$  is a probability distribution over the available actions in a state, for a *subset* of states in the state space ( $\mathcal{S}$ ), and parameterized by  $\theta \in \Theta$ . The states for which  $\sigma$  is defined are termed as the applicable states set, and are given by  $A_\sigma \subseteq \mathcal{S}$ .

$$\sigma: \mathcal{S} \times \mathcal{A} \times \Theta \rightarrow [0, 1] \quad (3)$$

[◀ Return](#)

# Performance Measure ( $J$ )

## Definition (performance measure)

The performance measure ( $J$ ) is a function  $J: \mathcal{M} \times \mathcal{G} \rightarrow \mathbb{R}$  that evaluates the performance of a strategy  $\sigma \in \mathcal{M}$ , in a game environment  $\mathcal{G}$ , by assigning it a numerical score.

◀ Return

# Interpretability Measure ( $\mathcal{I}$ )

## Definition (interpretability measure)

Formal

The interpretability measure ( $\mathcal{I}$ ) is a function  $\mathcal{I}: \mathcal{M} \times \mathcal{G} \rightarrow \mathbb{R}$  that evaluates the interpretability of a strategy ( $\sigma$ ) in a game environment ( $\mathcal{G}$ ) by assigning it a numerical score.

◀ Return

# Divergence

## Move Evaluation Function

Given chess engine  $E$  with position evaluation function  $v_E(s)$ , we can obtain a move evaluation function  $q_E(s, a)$  as —

$$q_E(s, a) = \sum_{s', r} \mathcal{P}(s', r | s, a) [r + v_E(s')] \quad (4)$$

$$= v_E(s'), s' \text{ is non-terminal} \quad (5)$$

Equation 5 follows from 4 since rewards in chess are 0 for non-terminal states,  $\gamma = 1$ , and chess rules are deterministic.

# Divergence

## Difference Function

Given two moves  $a_1, a_2$  made in a position  $s$ , we can calculate their difference  $d_E(s, a_1, a_2)$  as —

$$d_E(s, a_1, a_2) \doteq | q_E(s, a_1) - q_E(s, a_2) | \quad (6)$$

◀ Return

# Metrics

## Definition

**Coverage** is the fraction of positions in a given set  $P$  to which the chess strategy model is *applicable*.

$$P_A \doteq P \cap A_\sigma \quad (7)$$

$$\text{Coverage}(\sigma, P) \doteq \frac{|P_A|}{|P|} \quad (8)$$

# Divergence

## Definition (Divergence)

*Divergence* of a tactic from a set of examples  $P$  is the average difference in *evaluation* between the moves suggested by the tactic and the ground truth move.

$$\text{Divergence}_E(\sigma, P) \doteq \frac{1}{|P_A|} \sum_{(s, a_1) \in P_A} \sum_{a_2 \in \mathcal{A}(s)} \sigma(a_2 | s) d_E(s, a_1, a_2) \quad (9)$$

[◀ Return](#)

# PAL

- Patterns and Learning (Morales, 1992)
- ILP system to learn chess *patterns*
- Predicate vocabulary
- *rIgg* algorithm + heuristics to learn patterns
- Automatic *example generator* to learn target concepts

◀ Return

# Precision/Recall-based Constraints

## Definition (Precision constraint)

A precision constraint prunes the specializations of a hypothesis if its precision on a set of examples is less than some pre-defined lower limit.

## Definition (Recall constraint)

A recall constraint prunes specializations of a hypothesis if its recall on a set of examples is less than some pre-defined lower limit.

◀ Return

# Precision/Recall-based Constraints

## Theorem

*Given hypotheses  $H_1, H_2 \in \mathbb{H}$  with  $H_1 \preceq H_2$  and having recall values of  $r_1$  and  $r_2$  on a training set respectively, then  $r_1 \leq r_2$ .*

◀ Return

# Precision/Recall-based Constraints

Proof.

$$\text{Recall} = \frac{tp}{tp + fn} \quad (10)$$

$$H_1 \preceq H_2 \quad (11)$$

$$r_1 = \frac{tp_1}{tp_1 + fn_1} \quad (12)$$

$$r_2 = \frac{tp_2}{tp_2 + fn_2} \quad (13)$$

By the definition of theory subsumption, we have

$$tp_1 \leq tp_2, \text{ and} \quad (14)$$

$$fp_1 \leq fp_2 \quad (15)$$

Since the number of positive examples is constant for the same population,

$$\therefore r_1 \leq r_2 \quad (16)$$

[◀ Return](#)

# Predicate Vocabulary

- Allows more *situational rule* expression – en passant, promotion
- Allows *more efficient* unification due to introduction of more types
- Adds higher-order state features like *can\_threat* and *xrays*

◀ Return

# Interpretability Measure

# Interpretability Measure

- *No explicit interpretability measure!* Only qualitative arguments

# Interpretability Measure

- *No explicit interpretability measure!* Only qualitative arguments
- Human players *think* and *train* using chess tactics (Gobet & Jansen, 2006;  
Szabo, 1984)

# Interpretability Measure

- *No explicit interpretability measure!* Only qualitative arguments
- Human players *think* and *train* using chess tactics (Gobet & Jansen, 2006; Szabo, 1984)
- FO-logic used extensively to model chess patterns (Berliner, 1975; Bramer, 1977; Bratko, 1982; Huberman, 1968; Morales, 1992; Pitrat, 1977; Wilkins, 1979)

# Interpretability Measure

- *No explicit interpretability measure!* Only qualitative arguments
- Human players *think* and *train* using chess tactics (Gobet & Jansen, 2006; Szabo, 1984)
- FO-logic used extensively to model chess patterns (Berliner, 1975; Bramer, 1977; Bratko, 1982; Huberman, 1968; Morales, 1992; Pitrat, 1977; Wilkins, 1979)
- Logic rules are *acknowledged to be interpretable* (Zhang et al., 2021)

# Answer Set Programming

- *Declarative programming* paradigm based on *stable models* (Gelfond & Lifschitz, 1988)
- ASP language (Gebser, Harrison, Kaminski, Lifschitz, & Schaub, 2015) allows using rules to —
  - *model* a design space
  - *restrict* it using integrity constraints
  - *generate* instances in the newly restricted space

◀ Return

# Example

```
1 #const width=10.
2
3 param("width",width).
4
5 dim(1..width).
6
7 tile((X,Y)) :- dim(X), dim(Y).
8
9 adj((X1,Y1),(X2,Y2)) :- tile((X1,Y1)), tile((X2,Y2)), \
0   #abs(X1-X2)+#abs(Y1-Y2) == 1.
1
2 start((1,1)). finish((width,width)).
3
4 % tiles have at most one named sprite
5 0 { sprite(T,wall;gem;altar) } 1 :- tile(T).
6
7 % there is exactly one altar and one gem in the whole level
8 :- not 1 { sprite(T,altar) } 1. :- not 1 { sprite(T,gem) } 1.
```

An ASP program which can generate maze-like levels with integrity constraints that specify the number of game objects.

# Transformer

- Neural-network based model
- Great success at learning sequences
- Uses an *attention* mechanism to focus on relevant parts of a sequence

$$z_i = \sum_{j=1}^n \text{softmax}(\{\langle q_i, k_{j'} \rangle\}_{j'=1}^n)_j \cdot v_j \quad (17)$$

# Rise of Esports



Colin Young-Wolff/Riot Games

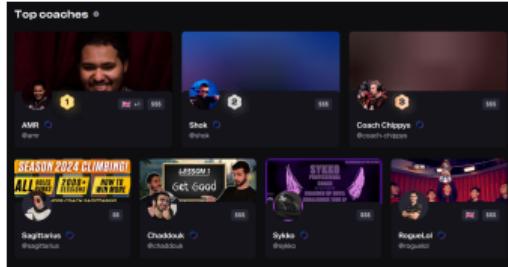
T1 huddles with the trophy onstage after their victory at the League of Legends World Championship 2023 Finals at the Gocheok Sky Dome on November 19, 2023 in Seoul, South Korea.

# Coaching in Esports



Robert Paul

Professional *Tekken 7* player Arslan Ash receives on-stage coaching during the *Tekken World Tour 2023 Finals*



Metafy

Top *League of Legends* coaches available for hire on the Metafy platform



RyderDie

A beginner guide for *Counter Strike 2* on YouTube

# Difficulties with Esports Coaching

Esports coaching is –

# Difficulties with Esports Coaching

Esports coaching is –

- not *standardized*: no “textbook” to follow

# Difficulties with Esports Coaching

Esports coaching is –

- not *standardized*: no “textbook” to follow
  - coaches follow *ad-hoc* methods

# Difficulties with Esports Coaching

Esports coaching is –

- not *standardized*: no “textbook” to follow
  - coaches follow *ad-hoc* methods
  - learning resources are *scattered* and *unstructured*

# Difficulties with Esports Coaching

Esports coaching is –

- not *standardized*: no “textbook” to follow
  - coaches follow *ad-hoc* methods
  - learning resources are *scattered* and *unstructured*
- *confined* to popular games

# Difficulties with Esports Coaching

Esports coaching is –

- not *standardized*: no “textbook” to follow
  - coaches follow *ad-hoc* methods
  - learning resources are *scattered* and *unstructured*
- *confined* to popular games
- *difficult* due to a shifting “*meta*”

# Difficulties with Esports Coaching

Esports coaching is –

- not *standardized*: no “textbook” to follow
  - coaches follow *ad-hoc* methods
  - learning resources are *scattered* and *unstructured*
- *confined* to popular games
- *difficult* due to a shifting “*meta*”
- not *scalable*

# Difficulties with Esports Coaching

Esports coaching is –

- not *standardized*: no “textbook” to follow
  - coaches follow *ad-hoc* methods
  - learning resources are *scattered* and *unstructured*
- *confined* to popular games
- *difficult* due to a shifting “*meta*”
- not *scalable*

## Solution

A *principled* and *automated* approach to esports coaching could democratize access to it for all interested players.

# Summary

Research Thrust	RQ	Sub-RQ	Publication
ISS Framework	RQ1	–	EAAI '22 <small>(Krishnan &amp; Martens, 2022b)</small>
		RQ2(a)	
ISS using Logic	RQ2	RQ2(b)	SG+EA Workshop @ AIIDE '22 <small>(Krishnan &amp; Martens, 2022a)</small>
		RQ2(c)	
ISS using Programs	RQ3	RQ3(a)	ACS '24 <small>(Krishnan et al., 2024)</small>
		RQ3(b)	

# Summary

Research Thrust	RQ	Sub-RQ	Publication
<i>ISS Framework</i>	RQ1	–	EAAI '22 <small>(Krishnan &amp; Martens, 2022b)</small>
		RQ2(a)	
ISS using Logic	RQ2	RQ2(b)	SG+EA Workshop @ AIIDE '22 <small>(Krishnan &amp; Martens, 2022a)</small>
		RQ2(c)	
ISS using Programs	RQ3	RQ3(a)	ACS '24 <small>(Krishnan et al., 2024)</small>
		RQ3(b)	

# Summary

Research Thrust	RQ	Sub-RQ	Publication
ISS Framework	RQ1	–	EAAI '22 <small>(Krishnan &amp; Martens, 2022b)</small>
		RQ2(a)	
<i>ISS using Logic</i>	RQ2	RQ2(b)	SG+EA Workshop @ AIIDE '22 <small>(Krishnan &amp; Martens, 2022a)</small>
		RQ2(c)	
ISS using Programs	RQ3	RQ3(a)	ACS '24 <small>(Krishnan et al., 2024)</small>
		RQ3(b)	

# Summary

Research Thrust	RQ	Sub-RQ	Publication
ISS Framework	RQ1	–	EAAI '22 <small>(Krishnan &amp; Martens, 2022b)</small>
		RQ2(a)	
ISS using Logic	RQ2	RQ2(b)	SG+EA Workshop @ AIIDE '22 <small>(Krishnan &amp; Martens, 2022a)</small>
		RQ2(c)	
<i>ISS using Programs</i>	RQ3	RQ3(a)	ACS '24 <small>(Krishnan et al., 2024)</small>
		RQ3(b)	

# Why Chess?

# Why Chess?

- *Popular* game with a *long* competitive history

# Why Chess?

- *Popular* game with a *long* competitive history
- Has a large number of *player-discovered strategies*

# Why Chess?

- *Popular* game with a *long* competitive history
- Has a large number of *player-discovered strategies*
- Extensive use as a *testbed for AI*
  - Convenient software implementations
  - Work in learning chess patterns
  - *Engines* for evaluation

# Dataset

- Randomly generated program dataset (Trivedi et al., 2021)
- Convert programs into a trajectory in the program synthesis MDP

---

Transitions	870,782
Programs	50,000
Program Length	$19.30 \pm 5.96$ (min = 7, max = 50)
Episode Length	$17.42 \pm 4.33$ (min = 5, max = 30)

---

Table: Statistics for the LEAPS program dataset.

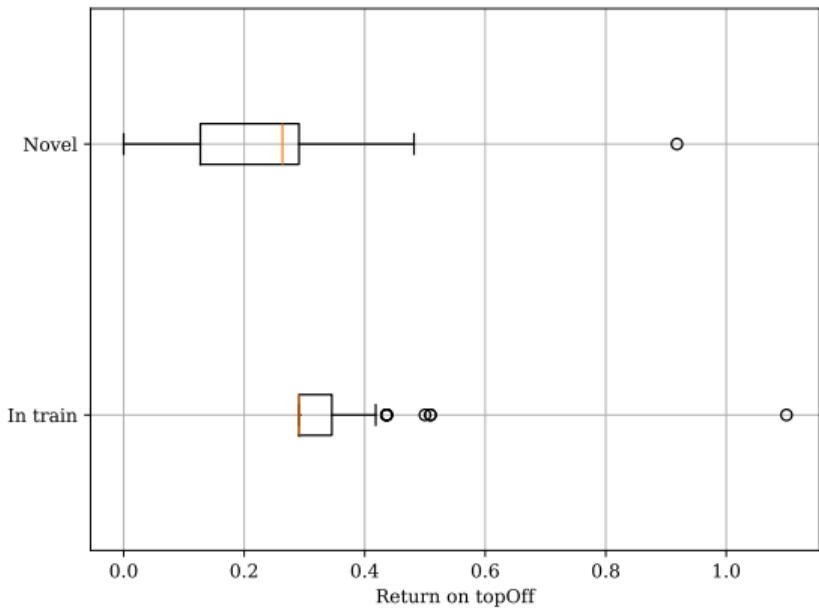
# Novelty of Generated Programs

# Novelty of Generated Programs

- Plotted distribution of rewards for novel and non-novel programs

# Novelty of Generated Programs

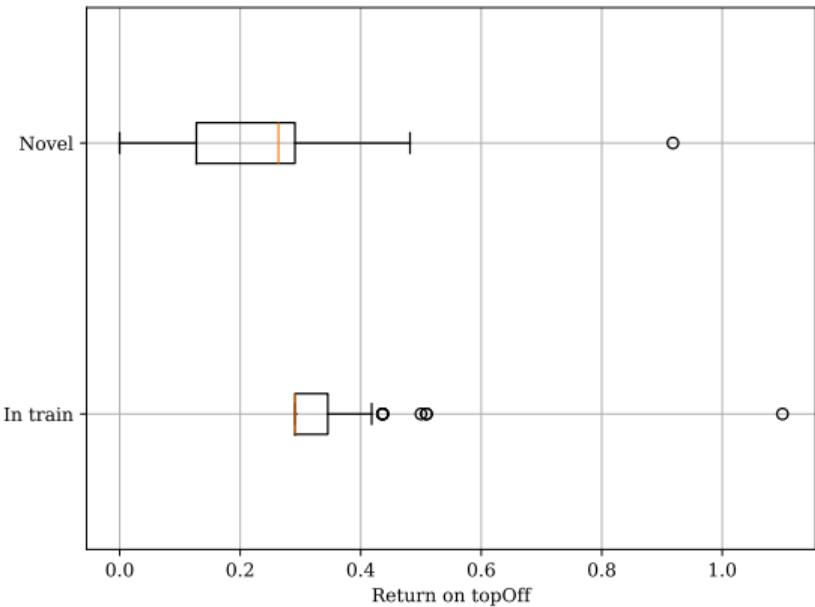
- Plotted distribution of rewards for novel and non-novel programs



Box plot of returns for programs sampled from the decision transformer model.

# Novelty of Generated Programs

- Plotted distribution of rewards for novel and non-novel programs
- Mean reward for novel programs is *lower* than non-novel programs ( $p < 0.01$ )



Box plot of returns for programs sampled from the decision transformer model.

# Limitations and Future Work

# Limitations and Future Work

- High cost of *knowledge engineering* – improved BK required  
~ 800 LoC of Prolog!

# Limitations and Future Work

- High cost of *knowledge engineering* – improved BK required  
~ 800 LoC of Prolog!
  - *general* policy BK

# Limitations and Future Work

- High cost of *knowledge engineering* – improved BK required ~ 800 LoC of Prolog!
  - *general* policy BK
- Performance not competitive with state-of-the-art

# Limitations and Future Work

- High cost of *knowledge engineering* – improved BK required ~ 800 LoC of Prolog!
  - *general* policy BK
- Performance not competitive with state-of-the-art
  - experiment with *task-specific* BK

# Limitations and Future Work

- High cost of *knowledge engineering* – improved BK required ~ 800 LoC of Prolog!
  - *general* policy BK
- Performance not competitive with state-of-the-art
  - experiment with *task-specific* BK
  - use *neural* ILP systems (Evans & Grefenstette, 2018)

# Limitations and Future Work

- High cost of *knowledge engineering* – improved BK required ~ 800 LoC of Prolog!
  - *general* policy BK
- Performance not competitive with state-of-the-art
  - experiment with *task-specific* BK
  - use *neural* ILP systems (Evans & Grefenstette, 2018)
- Interpretability for FOL-based strategies is poorly understood

# Limitations and Future Work

- High cost of *knowledge engineering* – improved BK required ~ 800 LoC of Prolog!
  - *general* policy BK
- Performance not competitive with state-of-the-art
  - experiment with *task-specific* BK
  - use *neural* ILP systems (Evans & Grefenstette, 2018)
- Interpretability for FOL-based strategies is poorly understood
  - study factors affecting interpretability (Kliegr et al., 2021)

# Limitations and Future Work

# Limitations and Future Work

- Different DSLs required for different domains

# Limitations and Future Work

- Different DSLs required for different domains
  - *general* policy DSL (Qiu & Zhu, 2022; Verma et al., 2018)

# Limitations and Future Work

- Different DSLs required for different domains
  - *general* policy DSL (Qiu & Zhu, 2022; Verma et al., 2018)
- Interpretability for programmatic strategies is poorly understood

# Limitations and Future Work

- Different DSLs required for different domains
  - *general* policy DSL (Qiu & Zhu, 2022; Verma et al., 2018)
- Interpretability for programmatic strategies is poorly understood
  - study factors affecting interpretability of programmatic strategies

# Expected Outcomes

- Benefit esports industry → *better coaching* for players
- Benefit explainable AI research → generate *policy explanations*

# Background and Related Work

## Automated Coaching for Esports

# Background and Related Work

## Automated Coaching for *Esports*

- Esports Analytics
- Automated Game Analysis

# Esports Analytics

# Esports Analytics

- *Data-driven* analysis of esports games to provide *support* to players

# Esports Analytics

- *Data-driven* analysis of esports games to provide *support* to players
- win prediction (Clark et al., 2020; Hodge et al., 2021; Maymin, 2021; Novak et al., 2020; Schubert et al., 2016)

# Esports Analytics

- *Data-driven* analysis of esports games to provide *support* to players
- win prediction (Clark et al., 2020; Hodge et al., 2021; Maymin, 2021; Novak et al., 2020; Schubert et al., 2016)
- patch-agnostic methods (Chitayat et al., 2023)

# Esports Analytics

- *Data-driven* analysis of esports games to provide *support* to players
- win prediction (Clark et al., 2020; Hodge et al., 2021; Maymin, 2021; Novak et al., 2020; Schubert et al., 2016)
- patch-agnostic methods (Chitayat et al., 2023)
- eye-tracking (Nagel, 2017; Stepanov et al., 2019)

# Esports Analytics

- *Data-driven* analysis of esports games to provide *support* to players
- win prediction (Clark et al., 2020; Hodge et al., 2021; Maymin, 2021; Novak et al., 2020; Schubert et al., 2016)
- patch-agnostic methods (Chitayat et al., 2023)
- eye-tracking (Nagel, 2017; Stepanov et al., 2019)
- data collection (Frommel et al., 2019; Xenopoulos et al., 2022)

# Automated Game Analysis

# Automated Game Analysis

- automated *testing* to discover bugs (Bergdahl et al., 2020; Pfau et al., 2017; Zheng et al., 2019)

# Automated Game Analysis

- automated *testing* to discover bugs (Bergdahl et al., 2020; Pfau et al., 2017; Zheng et al., 2019)
- automated *game balancing* to ensure fairness (Beau & Bakkes, 2016; DeLaurentis et al., 2021)

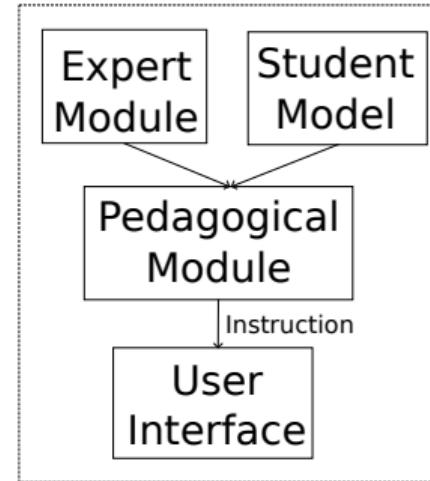
# Background and Related Work

## Automated *Coaching* for Esports

- Intelligent Tutoring System
- Player Modeling

# Intelligent Tutoring System (ITS)

- *Automated, personalized* education system
- No ITS for esports (Mousavinasab et al., 2021)



ITS architecture

# Player Modeling

- To provide *personalized* experiences in games
  - dynamic difficulty adjustment (Xue, Wu, Kolen, Aghdaie, & Zaman, 2017)
  - generated levels (Jennings-Teats, Smith, & Wardrip-Fruin, 2010)
- Strategy = player model (Smith, Lewis, Hullet, Smith, & Sullivan, 2011)

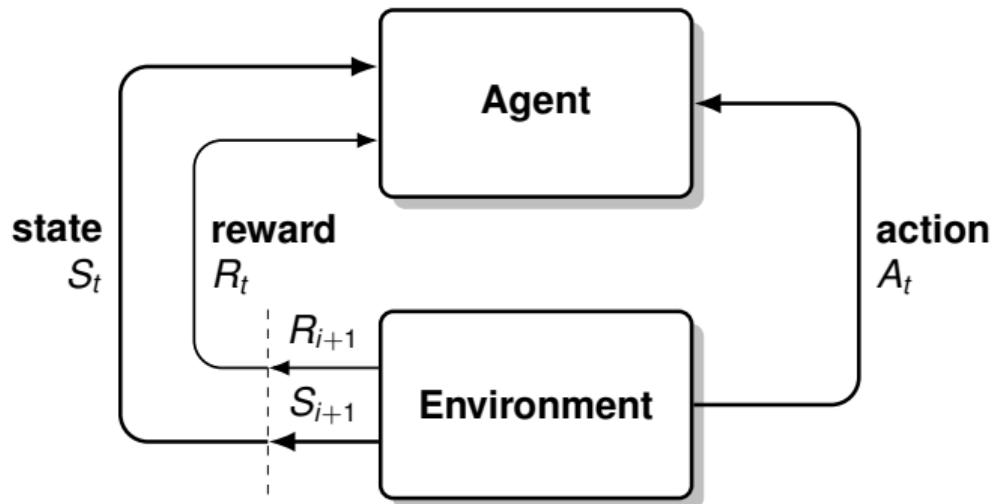
# Background and Related Work

## *Automated* Coaching for Esports

- Reinforcement Learning (RL)
- Explainable RL
- Strategy Synthesis

# Reinforcement Learning (RL)

- ML paradigm where an agent *learns to act* in an environment
- Game strategies = RL *policies*



The agent-environment interaction interface in a MDP. Reproduced from Figure 3.1 of Sutton and Barto (2018).

# Explainable Reinforcement Learning (XRL)

# Explainable Reinforcement Learning (XRL)

- Modern RL agents are *difficult* to understand compared to earlier approaches

# Explainable Reinforcement Learning (XRL)

- Modern RL agents are *difficult* to understand compared to earlier approaches
- Agents are being used in *high-stakes* applications

# Explainable Reinforcement Learning (XRL)

- Modern RL agents are *difficult* to understand compared to earlier approaches
- Agents are being used in *high-stakes* applications
- Necessary to *interpret* an agent and *explain* their decisions

# Explainable Reinforcement Learning (XRL)

- Modern RL agents are *difficult* to understand compared to earlier approaches
- Agents are being used in *high-stakes* applications
- Necessary to *interpret* an agent and *explain* their decisions
- XRL: field of study to devise techniques to interpret RL agents

# Explainable Reinforcement Learning (XRL)

- Modern RL agents are *difficult* to understand compared to earlier approaches
- Agents are being used in *high-stakes* applications
- Necessary to *interpret* an agent and *explain* their decisions
- XRL: field of study to devise techniques to interpret RL agents
- ISS = XRL for game environments

# Strategy Synthesis

# Strategy Synthesis

- Early work uses rule-based models (Butler et al., 2017; Canaan et al., 2018; de Freitas et al., 2018; de Mesentier Silva et al., 2016; Spronck et al., 2004)

# Strategy Synthesis

- Early work uses rule-based models (Butler et al., 2017; Canaan et al., 2018; de Freitas et al., 2018; de Mesentier Silva et al., 2016; Spronck et al., 2004)
- Subsequent work uses programmatic representations (Mariño et al., 2021; Mariño & Toledo, 2022; Medeiros et al., 2022)

# Strategy Synthesis

- Early work uses rule-based models (Butler et al., 2017; Canaan et al., 2018; de Freitas et al., 2018; de Mesentier Silva et al., 2016; Spronck et al., 2004)
- Subsequent work uses programmatic representations (Mariño et al., 2021; Mariño & Toledo, 2022; Medeiros et al., 2022)
- No unifying framework to synthesize goals and approaches