

# Model Decomposition for Forward Model Approximation

by Alexander Dockhorn, and Rudolf Kruse

Institute for Intelligent Cooperating Systems

Department for Computer Science, Otto von Guericke University Magdeburg  
Universitätsplatz 2, 39106 Magdeburg, Germany

Email: {alexander.dockhorn, rudolf.kruse}@ovgu.de

# Contents

- I. General Game-Playing vs. General Game-Learning
- II. Simulation-Based Search
- III. Forward Model Approximation
- IV. Evaluation
- V. Conclusion, Limitations and Future Work

# Why doing Research on Game-AI?

Games let us engage in a wide range of tasks.

- studying developed agents in multiple scenarios
- Games can model real life scenarios
- Solutions can be applied to all sorts of areas: robotics, finance, etc.

Games are (hopefully) well balanced by design.

- making the comparison of concurring algorithms feasible

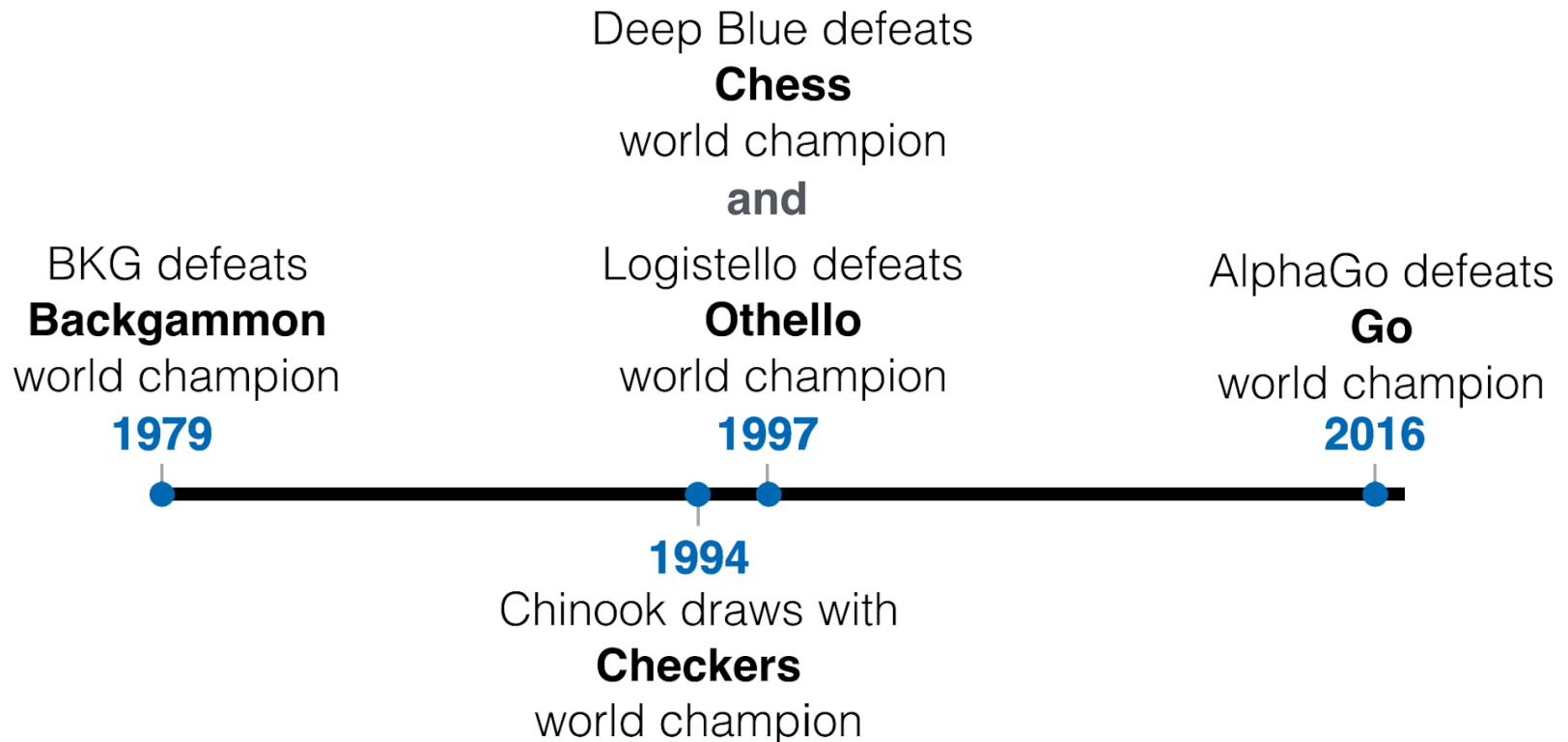
Games are hard because if their often vast finite state spaces.

- breadth and height of the search tree is often huge

Games are popular!

- more players, more games, more data

# What did AI in Games Achieve?



**All these solutions are based on online search algorithms.**

# Differences of Human and Computer Game-Playing

- humans can learn to play multiple games
- most AI agents focus on playing a single game

## **Next Step:** General Game Playing

- Requirements: unified state representation, unified forward model
- game definition languages offer both

But is this the same task?

- humans can learn to play games by playing them
- Can computers do the same?

## **Next Step:** General Game Learning

# General Video Game AI (GVGAI) Competition

Competition framework (<http://gvgai.net/>) including more than 100 games using Video Game Definition Language providing :

- general rules of a game and its forward model
- multiple levels per game
- visual representation





## Example game - Butterflies:

- control the small white fairy and catch all the butterflies.



# Example game - Butterflies:

Rule set of the game butterflies (see example level below):

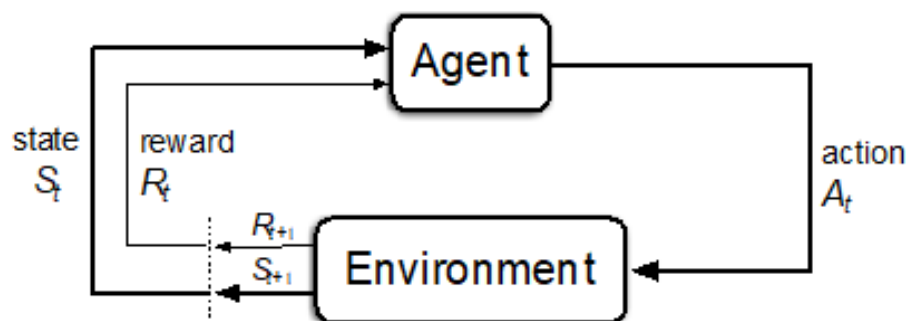
-  **Butterfly:** moves around randomly, is removed when it collides with the fairy
  -  **Fairy:** the agent's representation in the game, can be moved via actions
  -  **Cocoon:** static, when hit by a butterfly it transforms into another butterfly
  -  **Tree:** static, obstacle hindering movement of the fairy and the butterflies
- **Player actions:** {left, right, down, up} moves the fairy in the direction
  - **Win-Condition:** no butterflies left,    **Lose-Condition:** no cocoons left



# The Agent-Environment Interface

A general framework for studying games includes the following elements:

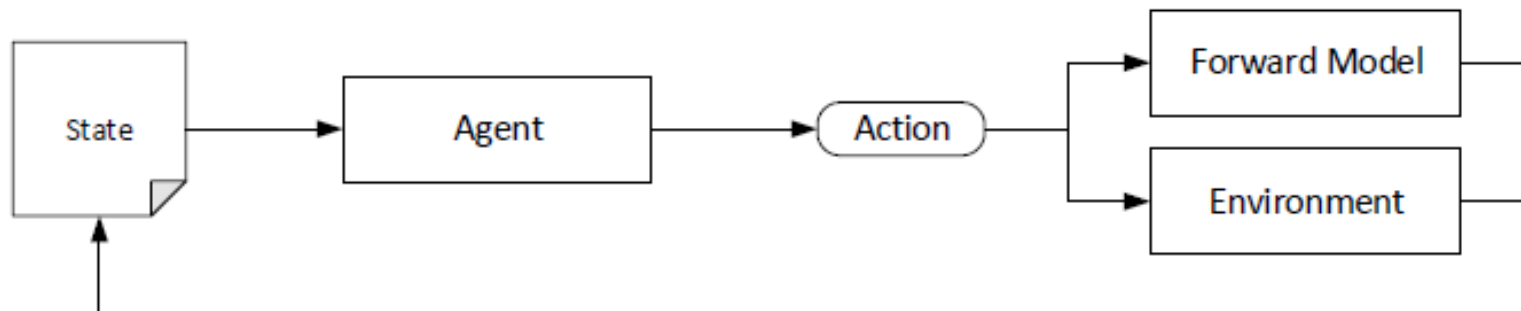
1. **Agent:** the learner and decision-maker
2. **Environment:** Anything the agent interacts with-everything outside the agent.
3. **Actions:** Agent and environment interact continually, the agent selecting actions and the environment responding to those actions and presenting new situations to the agent.
4. **Reward:** Special numerical values provided by the environment that the agent tries to maximize over time.



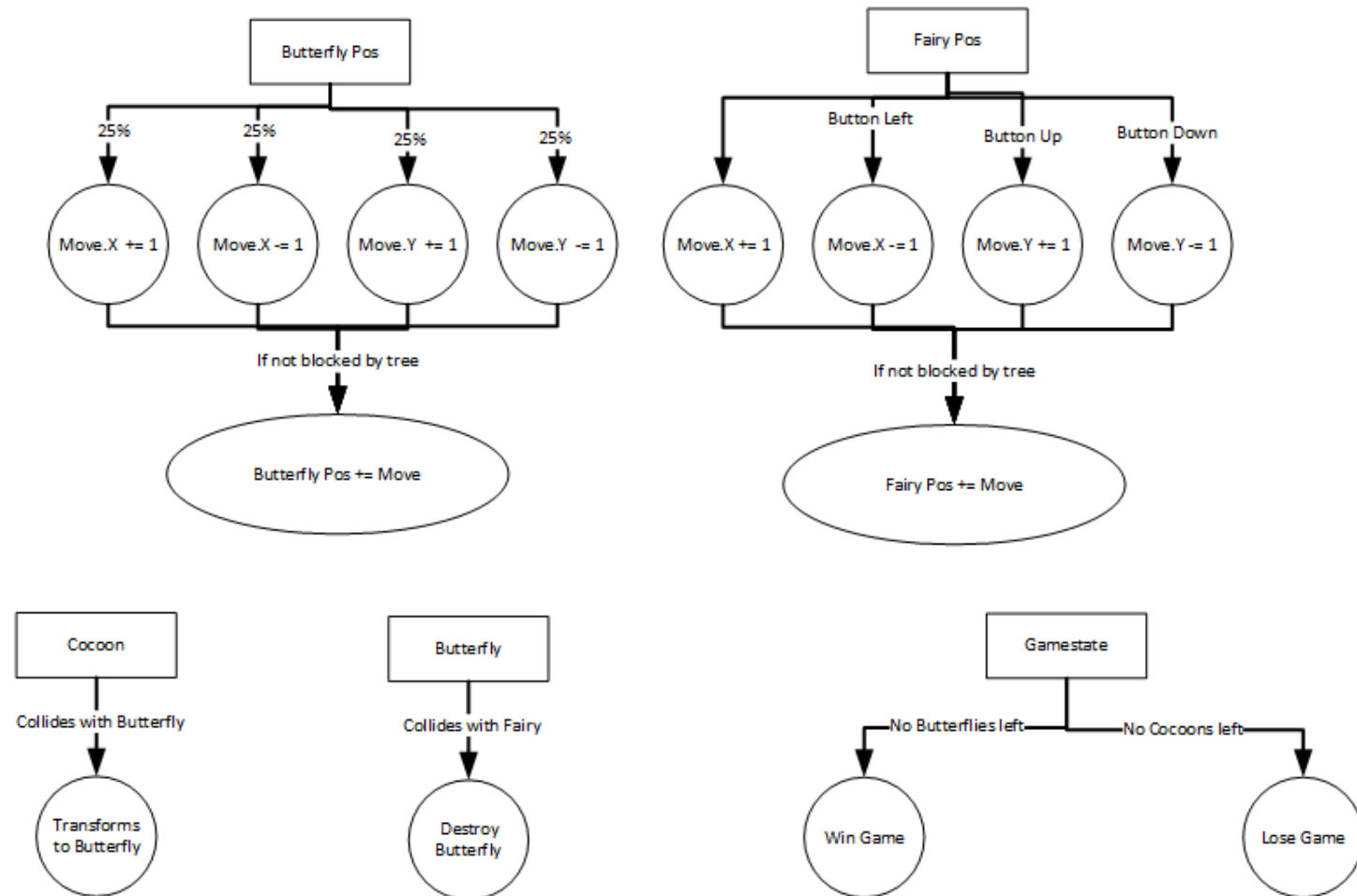


# Forward Models

- Using a forward model we can predict the outcome of an action.
  - defined by the rules of the game being played
- Search algorithms use the forward model to traverse the state-space
  - Actions (edges of the game tree) are chosen based on their outcome
- **Examples**
  - Robotics: activating a motor adjusts the leg, colliding with a ball moves it
  - Gameplaying Chess: taking the king wins the game




# Example game - Butterflies:



# Comparison of GVGAI Tracks

Planning Track	Learning Track
forward model available, 10 trials for training	no forward model, 5 minutes of training
simulation-based search is widely applied	reinforcement learning and heuristics are widely applied
algorithm performance close to or better than human	algorithm performance on par with random decision-making



huge performance gap of proposed solutions

# Restrictions of Simulation-Based Search

Simulation-Based search has two requirements:

- 1) the Forward Model needs to be known to the agent for forecasting the outcome of its actions
- 2) the current game state needs to be known, such that the agent can apply the forward model to it

Access to can sometimes be assured in (computer) games.

**What do we do in case the requirements are not fulfilled?**

**Do those scenarios exist?**

**Can simulation-based search still be applied?**

# Forward Model Approximation

Instead of learning a value function we try to learn a forward model.

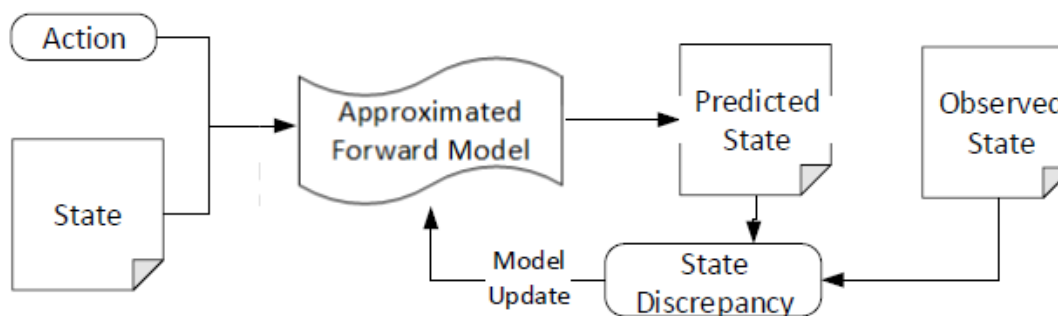
- apply simulation-based search using the learned model

## Idea:

In case the game fulfills the Markov Property the next state is only dependent on the observation of the current state and our action.

Classification task:

- map the current state and an action to the upcoming state



# Characteristics of the Forward Model

Possible models:

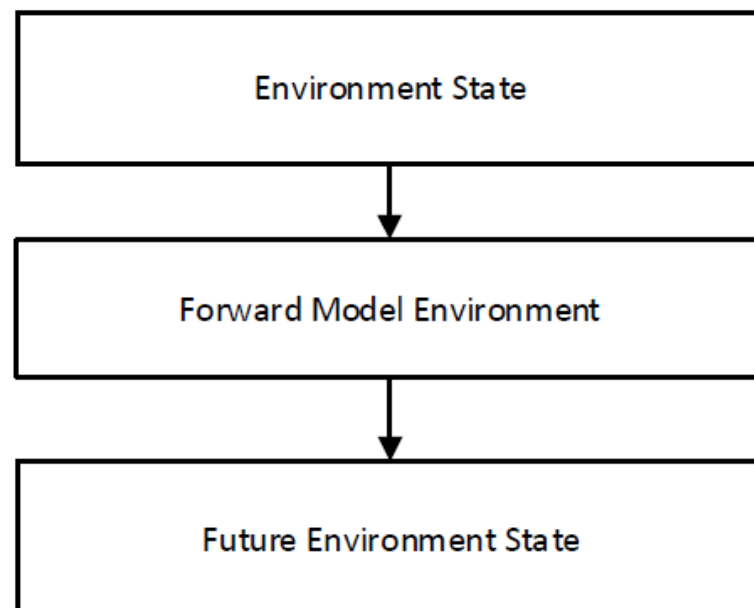
- Model the state change as a single transition, considering the history of previous game states, player actions and rewards

$$Pr\{S_{t+1} = s' |$$
$$S_0, A_0, R_1, \dots, S_{t-1}, A_{t-1}, R_t, S_t, A_t\}$$

- Markov property: only consider the latest interaction

$$Pr\{S_{t+1} = s' | S_t = s, A_t = a\}$$

- Problem: the state can be arbitrarily complex, how to reduce the number of possible models

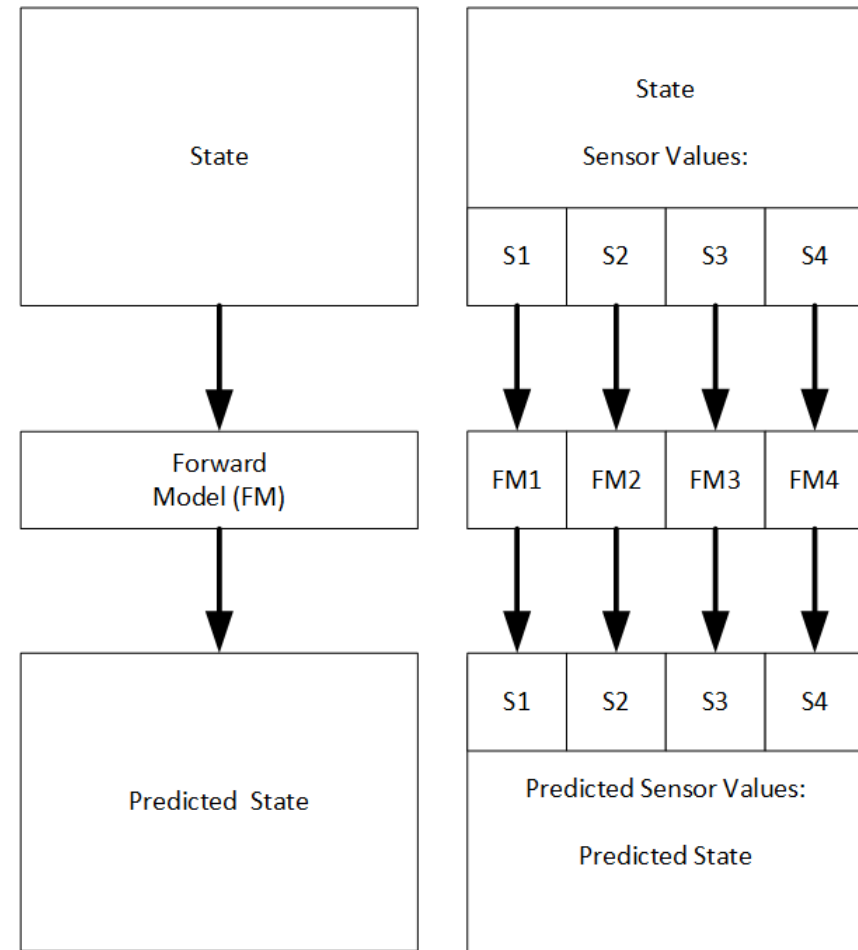


# Model Decomposition / Composite Model

- Modelling the complete state transition is very complex for a lot of applications
- Decomposing the forward model and learning a composite model (set of independent sub-models)

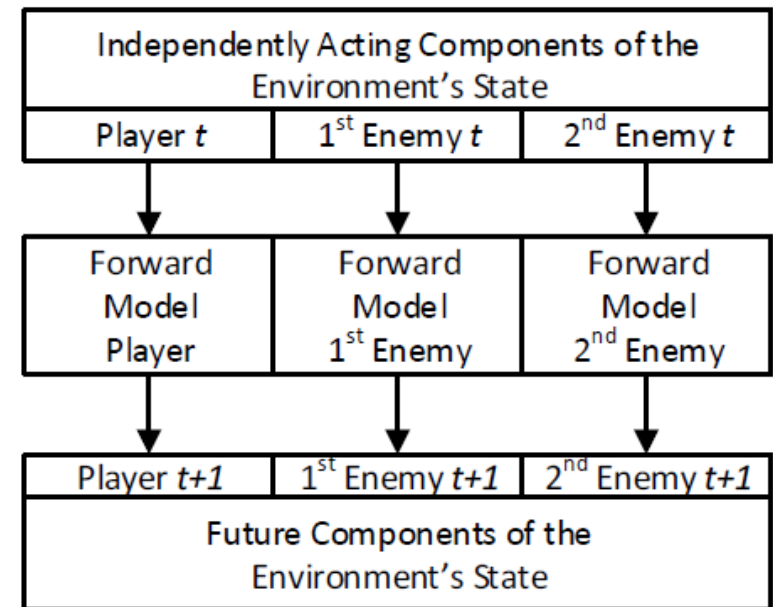
## Algorithm

- 1) Determine independent components or sensors
- 2) Create a model for each component
- 3) Predict each component
- 4) Aggregate the result of the composite model



# Characteristics of the Forward Model

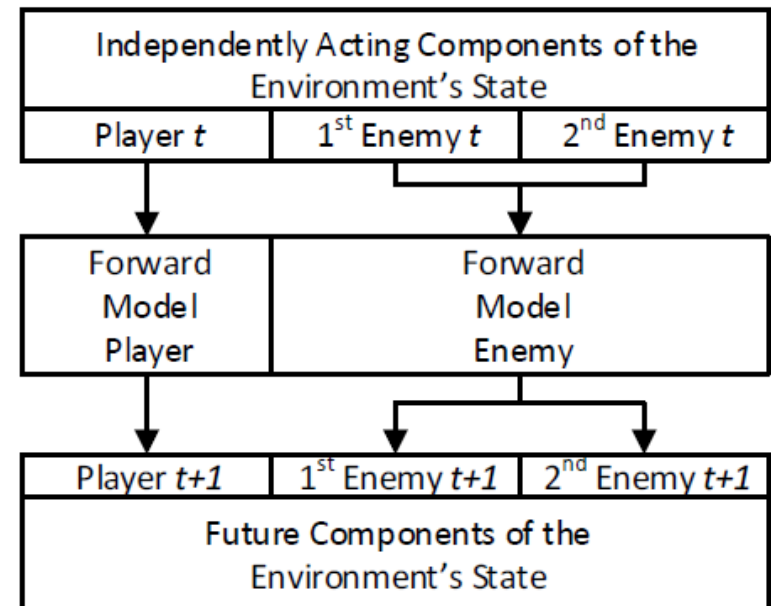
- Property of the Video Game Definition Language: model components individually and combine the result of each component's model.
  - Even if each component has a simple model complex behaviour can emerge
- Idea:
  - reduce the number of considered sensor values to an interesting subset
  - Predict the change of all components/sensors individually





# Characteristics of the Forward Model

- Learning a forward model includes finding a mapping for each component of the game
- Further reducing the model building by grouping instances
  - more training data per group
  - less models to learn



# Final Model for Forward Model Approximation

- Learn a model for each type of sensor value in the game
  - Filter input sensors of each model using knowledge on the Video Game Definition Language,
  - Combine training instances of similar behaving units and their sensor values, e.g. position information of similar units
- For every observable variable in the game, apply its respective model
- Aggregate the result of each model on each sensor value to predict the next gamestate
  - composite approximated forward model

# Evaluation

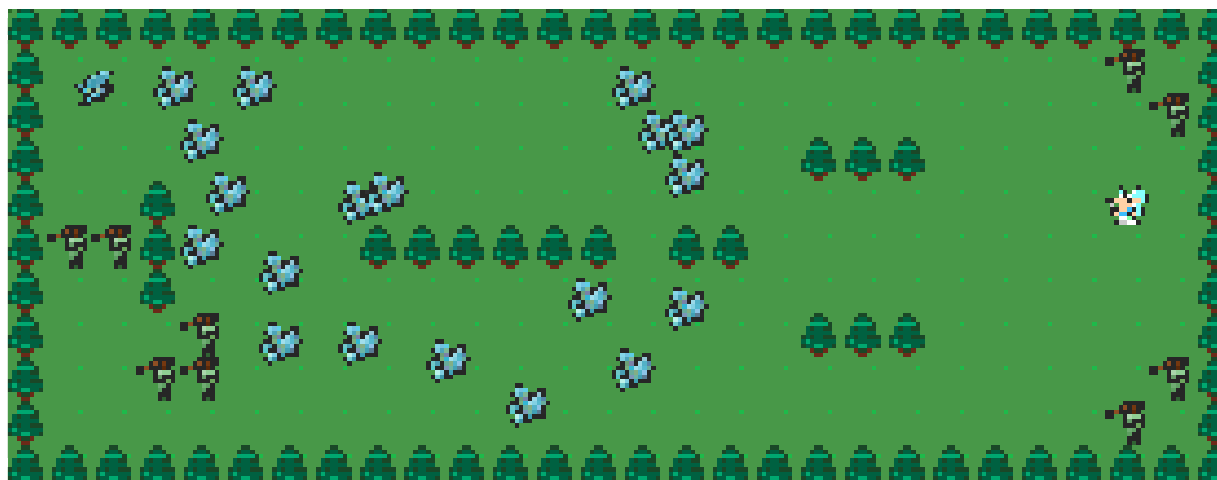
- Each game of the GVGA framework offers 5 levels
  - 2 for training, 3 for validation
- Train a new composite model after playing a level
- Test the composite approximated forward model by measuring the accuracy of its predictions after playing a single level
  - The next level might contain instances that were not present
  - The game may contain random behavior that reduces the baseline

# Summary

- Learning an approximated forward model converges much faster than learning the inherent value of an action
  - A high prediction accuracy can be achieved using just a few minutes/trials for training
  - The agent learns to play the game (to describe its environment) just by continuously interacting with it
- Decomposition into multiple sub-models and aggregation of their result speeds up learning
- Resulting composite model can be used as an interpretable representation of the environment's rules
  - The agent is modelling the outcome of its own actions

# Summary

- Forward Model Approximation is a basic framework in which we try to predict the outcome of an action rather than the value of it
- A high prediction accuracy can be achieved using just a few minutes/trials for training
  - High playing performance after just a few minutes of training!



# Limitations and Open Research Questions

- Local influence of variables is necessary to restrict the practical model space
  - Background knowledge on the application was used to restrict the model space to a smaller set of practical models
  - Dependency analysis may help to automatize this process
- Restrictions of the applied classifier still apply.
  - Which classifier should be used for which application?
- How does the accuracy of the approximated forward model influence the simulation-based search relying on it.
  - How good does the model need to be?
  - Can sensor values be left out without reducing the agent's playing performance?

# Thank you for your attention!

by Alexander Dockhorn, Tim Tippelt, and Rudolf Kruse

Institute for Intelligent Cooperating Systems  
Department for Computer Science, Otto von Guericke University Magdeburg  
Universitätsplatz 2, 39106 Magdeburg, Germany

Email: {alexander.dockhorn, tim.tippelt, rudolf.kruse}@ovgu.de