Summer Projects 2022-2023

Solving parity games in parallel using MATLAB and Mathematica. Parity games are games on graphs used in Computer Science for automated formal verification. From a computational complexity point of view, parity games are known to be in both NP and co-NP, but not known to be solvable in polynomial time, which has attracted a lot of research into the topic. In fact, many algorithms have been developed to solve parity games, most of which have been implemented in practice. Motivated by the desire to find new and efficient solutions for parity games in practical settings, *the goal of this project is to further study current algorithms for parity games using parallel implementations using MATLAB and Mathematica*, two powerful tools for mathematical analysis and calculations.

New parallel algorithms for parity games. Parity games are games on graphs used in Computer Science for automated formal verification. From a computational complexity point of view, parity games are known to be in both NP and co-NP, but not known to be solvable in polynomial time, which has attracted a lot of research into the topic. In fact, many algorithms have been developed to solve parity games, most of which have been implemented in practice. Motivated by the desire to find new and efficient solutions for parity games in practical settings, *the goal of this project is to further study parallel implementations of some of the best performing algorithms for parity games and evaluate their performance in practice with the state of the art in the literature,* specifically, against other parallel implementations.

A GUI for EVE. Equilibrium Verification Environment (EVE) is a formal verification tool for the automated analysis of temporal equilibrium properties of concurrent and multi-agent systems. In EVE, systems are modelled using the Simple Reactive Module Language (SRML) as a collection of independent system components (players/agents in a game) and players' goals are expressed using Linear Temporal Logic formulae. EVE can be used to automatically check the existence of pure strategy Nash equilibria in such concurrent and multi-agent systems and to verify which temporal logic properties are satisfied in the equilibria. *The goal of this project is to extend EVE by implementing a powerful and user-friendly Graphic User Interface (GUI) to enhance EVE's usability.*

Adding cooperative games to EVE. Equilibrium Verification Environment (EVE) is a formal verification tool for the automated analysis of temporal equilibrium properties of concurrent and multi-agent systems. In EVE, systems are modelled using the Simple Reactive Module Language (SRML) as a collection of independent system components (players/agents in a game) and players' goals are expressed using Linear Temporal Logic formulae. EVE can be used to automatically check the existence of pure strategy Nash equilibria in such concurrent and multi-agent systems and to verify which temporal logic properties are satisfied in the equilibria. Because Nash equilibrium is a non-cooperative solution concept, at present, EVE can only be used to analyse non-cooperative game-theoretic environments. *The goal of this project is to extend EVE through the implementation of algorithms already developed for the analysis of cooperative game-theoretic settings using the core as the main solution concept.*

Experimental evaluation of ALMANAC. Automaton/Logic Multi-Agent Natural Actor Critic (ALMANAC) is a recently developed multi-agent reinforcement learning algorithm used to learn to satisfy linear temporal logic specifications. ALMANAC has been implemented and some experiments have been produced to test its performance in some basic settings. However, a full experimental evaluation of ALMANAC has not been conducted yet. *The goal of this project is to evaluate ALMANAC in a wide range of learning environments so that its performance, optimality, and scalability can be studied much further than presently done*.

Hierarchical Reinforcement Learning with LTL Goals. Hierarchical Reinforcement Learning (HRL) can be used to decompose a complex learning task into a collection of subtasks, which may be simpler to learn or reason about. *The goal of this project is to develop HRL algorithms to learn to accomplish goals/tasks specified in Linear Temporal Logic (LTL)*, a language to express properties of concurrent and reactive systems.

Solving Automata using a Logic-Based Programming Approach. Several verification problems, especially those involving the use of temporal logics, can be solved by reducing such problems to questions about automata on infinite words, e.g., nonemptiness and containment. *The goal of this project is to use a state-of-the-art technology based on Logic Programming to solve those verification problems following the automata-theoretic approach to formal verification involving specifications in Linear Temporal Logic (LTL)*, a very expressive language to express properties of concurrent and reactive systems.

Lambda Calculi for Core Logic. In this project we will work on developing a lambda calculus that corresponds to a constructive logic known as Core Logic. Computation in this calculus is distinctive, since it is not guaranteed to preserve the type of a term. Rather, it can change the type in controlled ways. Moreover, computation acting on one part of a term can result in discarding other parts of the term. Despite these novel features, the implication-negation fragment of this calculus is known to be strongly normalising. *The aim of this project is to extend the calculus to include product and sum types as well, and demonstrate (weak or strong) normalisation for the extended calculus.*