Formal Reachability Analysis for Multi-Agent Reinforcement Learning Systems

XIAOYAN WANG\textsuperscript{1,2}, JUN PENG\textsuperscript{1}, SHUQIU LI\textsuperscript{1}, AND BING LI\textsuperscript{1}

\textsuperscript{1}College of Computer Science and Technology, Jilin University, Changchun 130012, China
\textsuperscript{2}Laboratory of Symbolic Computation and Knowledge Engineering, Ministry of Education, Jilin University, Changchun 130012, China

Corresponding author: Xiaoyan Wang (wangxy@jlu.edu.cn)

This work was supported in part by the Natural Science Foundation of Jilin Province under Grant 20200201290JC, and in part by the Fundamental Research Funds for the Central Universities, Jilin University (JLU), and Key Science and Technology Project of Jilin Province under Grant 20190304127YY.

\section*{ABSTRACT}
Reachability analysis is one of the most basic and challenging problems in verification. We investigate this problem in multi-agent reinforcement learning (MARL) system by transforming the reachability analysis to decision-making problem tackled by mixed integer linear programming (MILP) solver Gurobi. We define syntax and semantics for multi-role strategy logic (mrSL) which is used to describe the reachability specification. The logic mrSL is a variant of SL to express properties, which provide a holistic perspective to describe reachability properties instead of specifying a specific agent reaches a certain state in the MARL system. And the algorithms to translate reachability property into MILP constraints are provided. A tool called MAReachAnalysis is introduced, which uses Gurobi to solve the corresponding reachability problem and evaluate it on the predator-prey task of multi-agent deep deterministic policy gradient (MADDPG) example, discussion of the experimental results obtained on a range of test cases.

\section*{INDEX TERMS}
Multi-agent reinforcement learning, mixed integer linear program, multi-role strategy logic, verification.

\section*{I. INTRODUCTION}
As AI methods are increasingly used in safety-critical applications, concerns have been raised about the suitability of multi-agent reinforcement learning (MARL) for deployment in these applications. To ease this concern and gain users' trust, MARL system needs to be guaranteed that certain user-expected properties are satisfied. MARL systems are not programmed directly, instead, neural networks are first appropriately trained against data and then employed to conduct a particular task. So formal verification methods widely used in multi-agent system (MAS) verification cannot address the validation of AI applications due to the inherently different components [1], [2].

Generally speaking, a formal verification consists of two parts: (i) a neural network, (ii) a property to be checked, and its result is either a formal guarantee that the network satisfies the property, or a concrete input for which the property is violated. In the last few years, formal verification of neural network has grown up to be a very active field of research in computer science, with numerous success stories. Reference [3] provide a method of formally verifying desirable properties of neural networks, i.e., obtaining provable guarantees that neural networks satisfy specifications relating their inputs and outputs. Reference [4] submits a new direction for formally checking security properties of DNNs using interval arithmetic to compute rigorous bounds on the DNN outputs. Reference [5] introduce a general framework to certify robustness of neural networks with general activation functions. Reference [6] develop a method to verify automatically that no unwanted states are reached for the one agent reinforcement learning (RL) system. Reference [7] present a technique for the automatic verification of MAS populated by arbitrarily many agents adhering to different roles. Despite the emergence of various verification methods, reachability analysis is one of the most basic problems in verification, because it can be instantiated into several key decision problems, including safety verification [8]–[10], output range analysis [7], [11], and robustness comparison [12], [13]. But to our knowledge, there is little work in reachability analysis for MARL system.

To specify the verification property, alternating-time logic (ATL) and strategy logic (SL) are the well-known logical formalism. ATL is used to express cooperation and
competition among agents in order to achieve certain temporal goals, but it suffers from the strong limitation that strategies are treated only implicitly through modalities that refer to games between competing coalitions. To overcome this problem, SL is proposed in which strategies are explicitly referred to by using first-order quantifiers and bindings to agents. However, the policy specified by SL is assigned to each layer, if there are multiple agents in MAS, the property formula will be quite complex. So specifications concerning strategic interplay based on roles naturally are required in verification. For this reason we put forward multi-role strategy logic(mrSL), which is a variant of the SL and provides a holistic perspective to describe some properties of the system, regardless of which agent is responsible for the concrete task.

Due to the complex structure, manual reasoning for MARL system is impossible. Inspired by some MILP-based verification[14], [15], we transform the reachability analysis to decision-making problem tackled by MILP-based solver, which can answer queries about reachability properties by transforming these queries into constraint satisfaction problems. This article (i) provides a formal logic-mrSL to describe the reachability properties in MARL system; (ii) develops effective methods transforming the mrSL property into constraints;(iii) integrates the methods into a larger implementation and makes the experiment on MADDPG task predator-prey.

The rest of the paper is organized as following. We provide preliminaries related problem in Section II, along with a motivating example called predator-prey task in MADDPG. Section III introduces the syntax and semantics of mrSL. In Section IV, MILP based reachability analysis methods and the algorithms to translate reachability property into MILP constraints are introduced respectively. The example is given in Section V to illustrate our approach, related work and conclude are presented in Section VI and VII.

II. PRELIMINARIES

This section summaries and fixes the notation on some of the key notions used later in the paper.

A MARL system has several RL agents, each RL agent is a neural network(RLNN). Actually, RLNN usually is a fully connected feed forward neural network. Each RLNN is made up of a number of layers, each layer consists of multiple nodes, which are single computation units that combine input from the values of nodes in the previous layer, and then using these values to iteratively compute the assignments of nodes in each succeeding layer, finally producing an output.

Definition 1 (RLNN): a m-layer RLNN is a tuple RN=(L, L0, x, W, b, σ), where \( L^i \) denotes the \( i \)th layer of \( N \), \( L^0 \) is the input layer, \( L^m \) is the output layer, and all the other layers are so-called hidden layers. Each layer \( L^i \) has a weight matrix \( W^i \), a bias vector \( b^i \), and \( σ^i \) denotes an activation function in layer \( L^i \). Here we use \( x^i \) to denote the variables of layer \( L^i \).

\[
x^i = σ^i(W^{i-1}x^{i-1} + b^i)
\]

And the final output \( x^m = σ^m(W^i x^{i-1} + b^i) \)

Definition 2 (MARL System): MARL system is a tuple \( M = (RN_S, L, A, T, R) \), where

- \( RN \) is the set of RLNN agents
- \( S \) is the global state space
- \( L \) is the local state space
- \( A = \prod A_i \) is the joint action space
- \( R \) is the set of rewards
- Each agent \( i \) represented with \( RN_i \) executes action \( a_i \subseteq A_i \), where joint action \( a = (a_1, a_2, \ldots, a_n) \) causes environment state \( s \in S \) to transition \( T(s, a, s') \)

Supposing there are \( k \) roles in \( M, R = r_1 \cup r_2 \cup \ldots \cup r_k \) be the union of disjoint sets of role symbols, where each agent is associated with \( r_i \). Let \( \bar{n} = n^{k} \) be a vector to represent the agent set, and \( \bar{n}.i = \{RN^1_i, RN^2_i, \ldots, RN^n_i\} \) is the set of RL agents with role \( i \), where \( \bar{n}.i \) to be the \( i \)-th agent in \( \bar{n} \). For a global state \( s \in S \), we write \( l_i^s \) for local state of \( RN_i^s \) in \( s \), and

\[
S = \bigcup_{i=1}^{k} \bigcup_{l_i} l_i^s
\]

Our purpose is to use MILP to analyze the reachability of MARL system, so translating the reachability problem to a MILP model can boil down to expressing the constraints between the input and the output.

Definition 3 (Linear Function): a function \( f(x_1, x_2, \ldots, x_n) \) is said to be a linear function if for some \( c \in \mathbb{R}^n \), \( \sum_{i=1}^{n} c_i x_i \) is the set of \( \mathbb{R} \), \( b \in \mathbb{R} \), the expression \( f(x_1, x_2, \ldots, x_n) = b \), where \( \sim e \in \{=, \geq, \leq \} \) is said to be linear constraints.

Definition 4 (Linear Constraint): For any linear function \( f(x_1, x_2, \ldots, x_n) \) and any \( b \in \mathbb{R} \), the expression \( f(x_1, x_2, \ldots, x_n) \sim b \), where \( \sim e \in \{=, \geq, \leq \} \) is said to be linear constraints.

Definition 5 (MILP): A MILP is an optimization technique that uses to maximize or minimize a linear objective function subject to a set of linear constraints on the input values. The variables in MILP may be integer, real or binary type.

Definition 6 (Linear RLNN): an m-linear layer RLNN \( LRN = \{f(x)f^{(m)}(f^{(m-1)}(\ldots(f^{(1)}(x))))\)\), where each function \( f^{(i)} \) is a linear function.

But in MARL system, activation function introduces non-linearity into the output of a neuron. The most famous activation function is ReLU which is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. We called it a ReLU-RNN if the RLNN uses ReLU as its activation function. The method using binary variable to eliminate the non-linearities is discussed in [6], which can be done with the use of “big-M” encoding. In this article, the linear constraints encoding layer \( i \) is defined as follows:

Definition 7 (Linear Encoding for ReLU-RNN): Let \( N \) be a m-layer RLNN with computed function \( f \). Suppose \( x^{(i-1)} \) and \( x^{(i)} \) are vectors of real variables representing the input and output of layer \( i \), and \( δ^{(i)} \) is a vector of binary variable. Then the set of linear constraints encoding layer \( i \) is
defined as:

\[
\begin{align*}
    x_j^{(i)} &\geq W_j^{(i)} x^{(i-1)} + b_j^{(i)}, \\
    x_j^{(i)} &\leq W_j^{(i)} x^{(i-1)} + b_j^{(i)} + M \delta_j^{(i)}, \\
    x_j^{(i)} &\leq M(1 - \delta_j^{(i)})
\end{align*}
\]

where \( \delta_j^{(i)} = \begin{cases} 0 & \text{if } x_j^{(i)} > 0 \\ 1 & \text{otherwise} \end{cases} \)

Here, when \( x_j^{(i-1)} > 0 \), then \( \delta_j^{(i-1)} = 0 \), the constraints imply that \( x_j^{(i)} = W_j^{(i-1)} x^{(i-1)} + b_j^{(i-1)} \); but when \( x_j^{(i-1)} \leq 0 \), then \( \delta_j^{(i-1)} = 1, x_j^{(i)} = 0 \). So the MILP model can be obtained via the composition of constraints of the above form with appropriate linear constraints. We can apply this linear encoding to linearize all ReLU nodes and obtain a set of linear constraints.

In MARL system, the environment is stateful and updates its state in response to the action of the agent. Usually agents interact with non-linear environments. In this article, the training environments is modified to be linear so that we can use MILP to solve reachability problem.

**Definition 8 (Linear Environment):** \( LE = (S, t_E) \), where:

- \( S \) is the set of states of the environment
- \( t_E : S \times Act \rightarrow S \) is a transition function which given the current state of the environment and an action performed by the agent returns the next state of the environment.

**Definition 9 (Linear MARL System):** A linear MARL system \( LMS = (LRNs, LE, Init) \) is a tuple where:

- \( LRNs \) is the set of linear RLNN.
- \( LE \) is the linear environment.
- \( Init \) is the initial information, which may include initial state \( s_0 \in S \), initial observation \( o_0 \in O \), initial action \( a_0 \in A \).

MADDPG is one of the most advanced MARL methods that performs well in multi-agent environments [16], but after it has been trained, you don’t know if the training result is satisfying the need. We propose to study the reachability problem for the predator-prey task and report the results.

**Example 1:** MADDPG can be viewed as a multi-agent version of DDPG (deep deterministic policy gradient). Each agent has two networks: an actor network \( \pi \) and a critic network \( Q \). The actor network calculates the action to be executed based on the state acquired by the agent, while the critic network evaluates the action calculated by the actor network to improve the performance of the actor network.

In the training phase, the actor network only obtains observation information from itself, while the critic network acquires information such as the actions and observations of other agents. In the execution phase, the critic network does not participate, and each agent only needs an actor network.

The predator-prey task in MADDPG involves prey agents and a group of predator agents in figure 1. The prey is 30% faster than the predators, so aim of the predators is to learn how to team up in order to catch the prey, and the aim of the prey is to learn how to escape from predators. The task contains not only collaborative strategy, but also competitive strategy.

Because only actor network is used to calculate the action for each agent and the actor network is RELU-RLNN in MADDPG, the structures of these RELU-RLNNs are as follows if there are 3 predators \( (ag_1, ag_2 \text{ and } ag_3) \) and 1 prey \( (ag_4) \) in predator-prey task.

If the network is for predator, then the input layer makes up of 16 nodes; if the network is for prey, then the input layer consists of 14 nodes. There are 3 hidden layers, each layer consists of 64 hidden nodes and uses the ReLU activation function. The output layer consists of 5 nodes denoting the action.

In MADDPG example, the agent is not linear, because it uses Euclidean distance to compute if the collision happens between the two agents. In this article, Manhattan distance is used to calculate the distance between two agents. When the collision happens, \( |x_1 - x_2| \) and \( |y_1 - y_2| \) should be less than some fixed value \( v \). If the collision happens between the prey and the predator, the value is less than 0.125; if the collision between the two predators, the value is less than 0.1; and if the collision between the two preys, the value is less than 0.15.

**III. MULTI-ROLE STRATEGY LOGIC**

SL is an extension of the classic linear-time temporal logic LTL along with the concepts of strategy quantifications and agent binding [17]. SL provides a decidable language for talking in a natural and uniform way about all kinds of properties on game graphs, including zero-sum, as well as nonzero-sum objectives.

However, SL assigns policies to each agent, but if there are multiple agents in MAS, the formula will be more complex. For example, suppose there are 4 agents in the predator-prey task, \( Ag_1, Ag_2 \text{ and } Ag_3 \) are predators, while \( Ag_4 \) is prey. Now we want to know if the agent \( Ag_4 \) can be caught by the agents of predators. Actually we don’t know whether the agent \( Ag_3 \) catches the agent \( Ag_4 \), or the other agents catch it.

The formula using SL is as follows:

\[
\exists x \left( (Ag_1, x) FC (Ag_1, Ag_4) \right) \lor \left( \exists y \left( (Ag_2, y) FC (Ag_2, Ag_4) \right) \right)
\]

\[
\lor \left( \exists z \left( (Ag_3, z) FC (Ag_3, Ag_4) \right) \right), \text{ where the function } C(a, b) \text{ means agent } a \text{ can catch agent } b.
\]

The more the number of agents is, the more complex for the formula. So we propose mrSL, in which roles are added. And it is a variant of the SL and provides a holistic perspective to describe some properties of the system, regardless of which agent is responsible for the concrete task. We also set the
boundary with natural numbers denoting the temporal depth up to which the formula is evaluated.

**Definition 10 (mrSL Syntax):** Given a set of environment states \( S \subseteq \mathbb{R}^m \), the mrSL specification language over linear inequalities is defined by the following BNF:

\[
\varphi ::= \alpha \mid \neg \varphi \land \varphi \lor \varphi \mid X^{\leq k} \varphi \mid U^{\leq k} \varphi \mid \varphi \psi
\]

\[
\psi ::= \psi \lor \psi \mid \varphi
\]

\[
\alpha ::= c_1p_1 + c_2p_2 + \cdots + c_mp_m \text{ op } c
\]

where \( \text{op} \in \{<,>,\} \), \( p_i \in \{1,\ldots,m\} \), \( c_i, c \in \mathbb{R} \), and \( k \in \mathbb{N} \).

In definition 10, two new symbols are introduced, which are quantification prefix \( \varphi \) and binding prefix \( b \). A quantification prefix over a set \( V \subseteq \text{Var} \) of variables is a finite word \( \varphi \in \{(x)\mid x \in V\}^{\mid V\mid} \) of length \( |V| \) such that each variable \( x \in V \) occurs just once in \( \varphi \), i.e., there is exactly one index \( i \in [0, |V|] \) such that \( (\varphi_i) \in \{x\} \). The existential and the universal \( \{x\} \), can be read as “there exists a strategy \( x' \)”, and “for all strategies \( x \)” respectively. A binding prefix over a set \( V \subseteq \text{Var} \) is a finite word \( b \in \{(r, x)\mid r \in \text{role}\mid x \in V\}^{\mid \text{role} \mid} \) of length \( |\text{role}| \) such that each role \( r \in \text{role} \) occurs just once in \( b \), i.e., there is exactly one index \( i \in [0, |\text{role}|] \) such that \( p_i \in \{(r, x)\mid x \in V\} \). The role binding \((r, x)\) can be read as “bind role \( r \) to the strategy associated with the variable \( x \)”. Atomic propositions \( \alpha \) are linear constraints on the components of a state, in which \( p_i \) represents the linear constraints of the component of \( r_i \).

Actually there are some other operation in other logic, such as F(Evendually), G(Alwayis), R(Release) in LTL, CTL. We don’t import these operators because G and F, R and U(until) are dual operators, and they can switch to each other. For example, \( F_{\leq k} \varphi \iff \neg G_{\leq k} \varphi \land \varphi [U_{\leq k} \varphi_2 \iff \neg \varphi R_{\leq k} \varphi_2 \iff X_{\leq k} \varphi \).

So the goals in the example above can be expressed in mrSL as follows:

\[
\langle x \rangle (r_1, x) [y] (r_2, y) F_{\leq 10}^d (r_1, r_2) < 0.5
\]

It means for all the strategies used by role \( r_2 \), there is a strategy used by role \( r_1 \) that the distance between the agents in \( r_1 \) and \( r_2 \) is less than 0.5 within 10 steps.

From the mrSL definition, it includes two strategy formulas \((x) (r, x) \varphi \land (r, x) \varphi \), where the binding operation \((r, x) \varphi \) expressing the agent in role \( r \) has a strategy to enforce the formula \( \varphi \). Intuitively, mrSL formulae quantify over the concrete agents. For instance, \((x) (r, x) \varphi \) expresses that every concrete agent from role \( r \) can enforce \( \varphi \) independently of the action performed by other agents whether they belong to role \( r \) or not.

We can encode predator-prey tasks as a MILP model \( H \) composed of some linearized RLNN models, some of which are preys, the others are predators. The input of these RLNN network is the global states of each agent, and the output is the action for each agent. The reachability property for preys may be that they can escape from predators, but for predators is that they can catch the preys.

**Definition 11 (Path):** A path in a linear MARL system is a finite sequence of states \( \rho \in \mathbb{R}^t \), such that, for all \( j \in [0, |\rho| - 1] \), there exists a decision \( dc \in \mathbb{D}^c \) such that \( \rho(j + 1) = f_s(\rho(j), dc) \), where \( D_c = A_c \times Ag_j \), which means the choices of an action for each agent \( i \) of role \( r \). We use \( \rho_i^j(s) \) to represent the path starting at a state \( s \) for agent \( Ag_i^j \), and \( \rho(i) \) to denote the \( i \)-th state in \( \rho \).

**Definition 12 (Strategy):** A strategy for a concrete agent \( Ag_i^j \) is a function \( \Theta : \rho \rightarrow \bigcup_{r \in \text{role}} \bigcup_{c \in Ag_i^j} \). The two possible choices of actions depending upon the path. \( \Theta(s, Ag_i^j) \) denote all the strategies from state \( s \) of the agent \( i \) of role \( r \).

A valuation is a function \( v : \text{Var} \rightarrow \Theta \) that maps variables to strategies. A binding is a function \( \beta : Ag_i^j \rightarrow \text{Var} \) that maps agents to variables.

**Definition 13 (Assignment):** An assignment in a \( H \) is a partial function \( \chi : \text{Var} \cup Ag_i^j \rightarrow \Theta \) mapping variables, agents to a strategy.

Actually we can use \( v(x) \) and the composition function \( \nu(\beta(Ag_i^j)) \) to represent the assignment from vars and agents to strategies respectively. These two expressions can be abbreviated as \( \chi(l) \), where \( l \in \text{Var} \cup Ag_i^j \).

Given an assignment \( \chi \in \text{Asg} \), an agent, variable \( l \in \text{Var} \cup Ag_i^j \), and a strategy \( f \in \Theta \), we define a notation to represent a new assignment as following:

\[
\chi[l \rightarrow f](l') \overset{\Delta}{=} f
\]

the return is \( f \) if on domain \( l \), otherwise the value is equal to \( \chi \) on the remaining part of its domain.

We now give the semantics of the satisfaction relation for the logic.

**Definition 14 (mrSL Semantics):** Given a linear MARL system \( H \), for all mrSL formulas \( \varphi \), states \( s \in S \), and assignments \( \chi \), the modeling relation \( H, s, \chi \models \varphi \) is inductively defined as follows.

\[
H, s, \chi \models \varphi \text{ iff } \big( \sum_{i=1}^{m} c_i \times s.i \big) \text{ opc } \text{ and } l_i \text{ is the local state for role } r
\]

\[
H, s, \chi \models \neg \varphi \text{ iff } H, s, \models \varphi
\]

\[
H, s, \chi \models \varphi_1 \land \varphi_2 \text{ iff } H, s, \chi \models \varphi_1 \text{ and } H, s, \chi \models \varphi_2
\]

\[
H, s, \chi \models \varphi_1 \lor \varphi_2 \text{ iff } H, s, \chi \models \varphi_1 \text{ or } H, s, \chi \models \varphi_2
\]

\[
H, s, \chi \models X^{\leq k} \varphi \text{ iff } \exists \rho' \in \rho(s), \exists i < k, \rho'(i) \models \varphi
\]

\[
H, s, \chi \models U^{\leq k} \varphi_2 \text{ iff } \exists \rho' \in \rho(s), \exists i + k \leq j, \rho'(j) \models \varphi
\]

\[
H, s, \chi \models (a, x) \varphi \text{ iff } \forall f \in \Theta(s), H, s, \chi[a \rightarrow (x)] \models \varphi
\]

**IV. REACHABILITY ANALYSIS**

In this section, we outline the ideas and details of our reachability analysis method. Firstly, the reachability problem in linear MARL system is formally introduced, then construction of MILP models for MADDPG is described.
Finally the multi-agents multi-Step state reachability (MMSR) algorithm is provided.

A. PROBLEM FORMULATION

The general reachability problem of MARL system is to determine whether some states on the output holds for all or some input in a bounded input domain. Section 2 shows that a trained RL neural network can be represented with MILP model regardless of its layer depth and number of neurons. So the reachability problem in MARL system can be translated into using MILP to analyze the reachability problem.

Definition 15 (Reachability of Linear MARL System): Let \( X_0 \) be an input subspace of agent \( i \), \( F^i(X_0) = f^{(m)}f^{(m-1)}(\ldots f(X_0)) \) be a linearized RLNN network, \( \{C_j\} \) be the constraints for agent \( j \) in layer \( i \), and \( Y^i \) denotes the output for agent \( i \). The reachability of linearized MARL system is to compute if \( \exists y' \in Y^i, x_0 \in X_0, y' = F^i(x_0) \) under the constraint set \( C \) for each agent, where

\[
C = \bigcap_{i=1}^{m} \left( \left( C^i_m \right)^T \left( f^{(m)}f^{(m-1)}(\ldots f(X_0)) \right) + d_i \leq 0 \right)
\]

From the definition above, we can transform the reachability objective to constraints, that is to say, for the agents in MARL system their objective can be described as \( z = 0 \) and \( C_{reach} = C \). Hence we can reduce the reachability analysis to solving a linear program defined on the constraints \( C_{reach} \), which is a MILP problem.

We have introduced how to transform the ReLU function to be a linear constraint set \( C_i \) in definition 7. Let \( C_R = \bigcup_{j=2}^{m} C_i \) denote the linear constraint set for all the layer of each RLNN.

Definition 16 (Linear Model for MARL System): Let \( C_l \) and \( C_o \) to denote the linear constraints on input and output for each RLNN, the linear encoding for MARL system is defined by the set of constraints and variables \( C_{lp} = \sum_{n=1}^{n} \{X_0\} \cup C_l \cup \{X\} \cup C_R \cup \{Y\} \cup C_o \), where \( n \) is the number of agents.

Actually, the target constraints \( C_o = \sum_{i=1}^{n} C^i_o \) can be expressed in disjunction and conjunctive form. The conjunctive expression can be used in MILP model directly, but for the disjunctive expression, we turn it into linear expression using binary variables as \( \delta \) does.

Supposing agent \( i \) and agent \( j \) have the constraints \( C_i^j \) and \( C_j^i \) respectively, the disjunctive form for the target constraints \( C_o = C_i^j \cup C_j^i \), we can use a binary variable \( \delta \) to indicate which constraint is satisfied.

\[
C^i_o \cup C^j_o = \left( (C^i_o = \text{true}) \cup (C^j_o = \text{true}) \right) \cup \left( (\delta = 1) \land (C^i_o = \text{true}) \right) \cup \left( (\delta = 1) \land (C^j_o = \text{true}) \right)
\]

And the multi-agent disjunctive constraints can be expressed as follows:

\[
C^1_o \cup C^2_o \ldots C^n_o = \bigcup_{i=1}^{n} \left( (\delta_i = 1) \land (C^i_o = \text{true}) \right) \cup \sum_{i=1}^{n} \delta_i = 1
\]

Algorithm 1 formula (1) translation

Procedure AAF (\( ar1, ar2, d \))
1: Input: \( ar1, ar2, d \)
2: Output: constraints
3: \( C = \emptyset \)
4: for \( i \) in \( ar1 \) do
5: \( \text{ag1} = \text{Agt}_r^1 \)
6: for \( j \) in \( ar2 \) do
7: \( \text{ag2} = \text{Agt}_r^2 \)
8: \( c = \text{Distance}(\text{ag1}, \text{ag2}) < d \)
9: \( C \cup c \)
10: return \( C \)

B. REACHABILITY SPECIFICATIONS WITH mrSL

We consider the following specifications:

\( S1 \): Is it ever the case that each prey can be caught by the predator within \( k \) steps. It can be described by mrSL as follows:

\[
\forall x (\text{prey}, x) \exists y (\text{pred}, y) F<\delta^k d (\text{prey}, pred) < 0.125
\]

\( S2 \): Is it ever the case that the preys hasn’t been caught from the initial states within \( k \) steps.

\[
\forall x (\text{prey}, x) \forall y (\text{pred}, y) F<\delta^k d (\text{prey}, pred) > 0.125
\]

\( S3 \): Is it ever the case that there exists one prey has been caught from the initial states within \( k \) steps.

\[
\exists x (\text{prey}, x) \exists y (\text{pred}, y) F<\delta^k d (\text{prey}, pred) < 0.125
\]

To analyze reachability, the above specifications should be converted into constraints in the MILP model. The logic mrSL contains the strategy formula \( \forall x (a, x) \) and \( \exists x (a, x) \), so there are four types of combinations,

\[
\begin{align*}
\forall x (a, x) \forall y (b, y) \varphi & \quad (1) \\
\forall x (a, x) \exists y (b, y) \varphi & \quad (2) \\
\exists x (a, x) \forall y (b, y) \varphi & \quad (3) \\
\exists x (a, x) \exists y (b, y) \varphi & \quad (4)
\end{align*}
\]

The semantics of formula (1) is that all the strategy \( x \) using by any agent in role \( a \) and all the strategy \( y \) using by any agent in role \( b \) satisfy \( \varphi \). The semantics of formula 4 is that there is a strategy \( x \) using by an agent in role \( a \) and strategy \( y \) using by an agent in role \( b \) satisfy \( \varphi \). We think the semantics of formula (2) and formula (3) are the same, it means for all the strategy \( x \) using by any agent in role \( a \) and there exists a strategy \( y \) using by an agent in role \( b \) satisfy \( \varphi \). So we need to translate formula (1), (4) and (2) to the MILP constraints, and the algorithms are described as following.

In algorithm 1, the input includes roles \( ar1, ar2 \), and the distance value to be compared \( d \). The formula (1) is transformed into the set of conjunctive constraints that must be satisfied.

In algorithm 2, the formula (2) is transformed into the disjunctive and conjunctive set of all constraints that must be
Algorithm 2: formula (2) translation

Procedure AEF(ar1, er2, d)
1: Input: ar1, er2, d
2: Output: constraints
3: C = ∅
4: i = num(ar1); j = num(er2)
5: if i > j
6: while (ar1(i) ! = null) do
7: C = C ∨ AEF(ar1(i), er2, d)
8: else
9: while (ar2(i) ! = null) do
10: C = C ∨ AEF(ar1, er2(i), d)
11: return C

Algorithm 3: formula (4) translation

Procedure EEF(er1, er2, d)
1: Input: er1, er2, d
2: Output: constraints
3: C = ∅; k = 0
4: for i in er1 do
5: ag1 = Agt_i
6: for j in er2 do
7: ag2 = Agt_j
8: C = C ∨ Distance(ag1, ag2) < d
9: C = C ∨ c
10: return C

satisfied. The function ar1(j) returns the set of j objects of role ar1. The meaning of er2(i) is same.

In algorithm 3, the formula (4) is transformed into the disjunctive set of all constraints that must be satisfied.

C. MULTI-AGENTS MULTI-STEP STATE REACHABILITY

In fact, the three specifications above can be reduced to the multi-agent multi-step state reachability (MMSR) decision problem, that is to say, if the agents can arrive at the specified states within multi-steps.

Definition 17 (MMSR decision problem): Let n ∈ N and S" ∈ S be the target states, where S'" ∈ S" represent the target state of agent ag_i. The MMSR decision problem is to determine whether there is a case for each agent ag_i within k step, that ∃X'"_i ∈ I, such that S'"_i = F^k_i(X'"_i) for each S'"_i.

So the MMSR problem can be solved by k-step iteration.

C_{lp} = \sum_{j=1}^{k} \sum_{i=1}^{n} \{X'_{i,j} \cup C'_{i,j} \cup \{X'\} \cup C'_{R} \cup \{Y'\} \cup C'_{O} \}

The MMSR analysis procedure is given by Algorithm 4. In it the method spec2constraints will call the algorithm 1, 2 and 3 according to the formula type and returns constraint set.

V. EXPERIMENTS

We implemented the methods described in the previous section in a toolkit, called MAReachAnalysis, that solves the reachability decision problem in MARL system. MAReachAnalysis takes as input several linear RLNNs and a mrSL specification ϕ, the toolkit returns whether or not the specification ϕ holds. The set init of initial states is given in the form of a hyper-rectangle [l_1, u_1] × ⋯ × [l_n, u_n]. We experimented with ReLU-RLNNs using tensorflow, but other choices are possible.

To evaluate the correctness and the performance of MAReachAnalysis, we analyzed the predator-prey task described in Example 1 with different agent number. The neural networks used in this article are trained 10000 iterations. The experiments were run on an Intel Core i7-9750 CPU (2.600GHz, 8 cores) running Ubuntu 18.04 upon which we invoked Gurobi version 9.0.

A. SCENARIO 1 WITH 3 PREDATORS AND 1 PREY

In this scenario, we describe the global states of the predator-prey task using the set of position of each agent S = {obs_1, obs_2, obs_3, obs_4}, where obs_1, obs_2 and obs_3 are for predators, and obs_4 for prey. Actually the structure of obs_1 to obs_4 is also the set of tuples.

obs_i = (p_vel, p_pos, entity_pos, other_pos, other_vel)

• p_vel is the velocity of the agent, it’s a two-dimensional vector
• p_pos is the position of the agent, it’s a two-dimensional vector
• entity_pos is the relative position between the agent and the landmarks, it’s a two-dimensional vector multiplying the number of the landmarks
• other_pos is the relative position between the agent and the other agents, it’s a two-dimensional vector multiplying the number of the agent but minus one
• other_vel is used for communication, it’s not used in this example.

The length of obs_1, obs_2 and obs_3 is 16, but for obs_4 is 14. A joint action Act is a tuple of local actions for all the agents in the system. The set Act = (Act_1, Act_2, Act_3, Act_4, Act_E), where Act_1 is a 5-dimension Act_1 = (up, down, left, right, reserved). The first 4 items specify the desired movement in Cartesian coordinates and the last item is reserved. In MADDPG example, the environment has one dummy action ε because it does nothing to the agent, so the actions for environment Act_E is equal to ε.

Similarly, a transition is a tuple of local transition for all the agents in the system T=(t_1, t_2, t_3, t_4), where each
local transition function is defined as follows. \( t_i (s (i), a_i) = p_i \_pos + p_i \_vel \times \Delta t \), where \( p_i \_vel \) is also obtained from \( obs_i \).

We attempt to perform reachability analysis on predator-prey task and report the analysis results. The MAReachAnalysis toolkit builds the set of linear constraints necessary based on Definition 7 and Definition 15. Each MILP model consists of three different groups of constraints: (1) constraints on the input to ensure that its value stays within bounds, that’s the \( \mathcal{C}_I \) constraints; (2) constraints on the output of each layer ensuring that it is in the safe level of floating precision, that is the \( \mathcal{C}_R \) constraints; (3) constraints on the final output result, that is the \( \mathcal{C}_O \) constraints. We define the range for \( p \_pos \) is \([-1, 1]\), for output value is \([0,1]\), and we add total perturbation tolerance for the RLNN, which is the sum of perturbation in each layer belonging to \( \mathcal{C}_R \).

The constraints are then passed to Gurobi, which is used to determine whether the generated linear program admits a solution. Following the output from Gurobi, a result is then presented to the user as either a feasible result, indicating that an output state is reachable from the initial states or an infeasible result representing unreachability of the output states.

\( S^1 \): Is it ever the case that each prey can be caught by the predator within \( k \) steps.

In predator-prey task, the condition of the predators catching the preys is the distance between them less than 0.125 (The maximum collision distance is the sum of the sizes of predator and prey. The size of predator is 0.075, prey is 0.05, landmark is 0.2). Using algorithm 2, \( S^1 \) can be transformed into the following conjunctive set:

\[
\begin{align*}
\text{Distance} (A^{0}_{\text{pred}}, A^{0}_{\text{prey}}) & \leq 0.125 \\
\lor \text{Distance} (A^{1}_{\text{pred}}, A^{0}_{\text{prey}}) & \leq 0.125 \\
\lor \text{Distance} (A^{2}_{\text{pred}}, A^{0}_{\text{prey}}) & \leq 0.125
\end{align*}
\]

In this article, we use Manhattan distance to calculate whether there is a collision between the two agents. The linear constraints are added to the model as following:

\[
\begin{align*}
gmodel.addConstr(C_1) \\
gmodel.addConstr(C_2) \\
gmodel.addConstr(C_3)
\end{align*}
\]

where

\[
\begin{align*}
C_1: & \text{delta03} \_x \times d_1 + \text{delta13} \_x \times d_2 + \text{delta23} \_x \times d_3 \\
& \leq 0.125 \\
C_2: & \text{delta03} \_y \times d_1 + \text{delta13} \_y \times d_2 + \text{delta23} \_y \times d_3 \\
& \leq 0.125 \\
C_3: & d_1 + d_2 + d_3 == 1
\end{align*}
\]

The variables \( \text{delta03} \_x \) and \( \text{delta03} \_y \) are the X-axis and Y-axis distance between predator \( A^{0}_{\text{pred}} \) and prey \( A^{0}_{\text{prey}} \) which are the linear constraint. Similarly, \( \text{delta13} \_x \) and \( \text{delta13} \_y \) are distance between \( A^{1}_{\text{pred}} \) and prey \( A^{0}_{\text{prey}} \).

We denote by \((obs_1, obs_2, obs_3, obs_4)\) the initial state of the environment, and \((\bar{obs}_1, \bar{obs}_2, \bar{obs}_3, \bar{obs}_4)\) the final state of the environment after \( n \) time steps. The first time we fix a set of initial states with \( obs_1 \_p \_pos, x = [-0.95, -0.75], obs_1 \_p \_pos, y = [0.7, 0.95], obs_2 \_p \_pos, x = [-0.95, -0.75], obs_2 \_p \_pos, y = [-0.95, -0.75], obs_3 \_p \_pos, x = [0.7, 0.95], obs_3 \_p \_pos, y = [-0.98, -0.7] \). This corresponds to states where the predators are at the edge of the land, and the prey is in the middle. We checked whether that the prey can be caught by the predator within \( k \) steps, where \( k \) is equal to 5, 10, 15 individually.

The second column vars number includes continuous(c) and binary(b) vars. The solver reported the solution found for the problem if the result is “Y” in the table. This shows that the specification can be satisfied.

In fact there are many strategies for predators to catch the prey. Here we give one strategy that the prey is caught from initial state within 5 steps. The initial state is \( S_0 = (s_0, s_1, s_2, s_3) \), where

\[
\begin{align*}
s_0 &= ([-1.0, 0.2927395690553022], [-0.3, 0.5], [-1.0, -0.6833445104188195], [-1.0, 1.0], [-1.0, 0.5025505270383848], [-1.0, 0.3090956358820337], [-0.0224715328496156, -1.0], [1.0, 1.0], [0.2, -0.2], [1.0, -1.0], [1.0, 0.44776210432684316], [-1.0, 1.0], [-1.0, -1.0], [1.0, 1.0], [1.0, 1.0], [1.0, 0.0293622027069687]),
\end{align*}
\]

After 5 steps, the position between the predator 1 and prey is \([0.0, 0.007961222072069687] \), which means the predator 1 catches the prey.

\( S^2 \): Is it ever the case that the preys hasn’t been caught from the initial states within \( k \) steps.

The formal formula for \( S^2 \) specification is as follows:

\[
\forall x (\text{prey}, x) \forall y (\text{pred}, y) F [\leq k] d (\text{prey}, \text{pred}) \geq 0.125
\]

First we use algorithm 1 to transform the msSL formula to constraints, and it will return the conjunctive form result.

\[
\begin{align*}
\text{Distance} (A^{0}_{\text{pred}}, A^{0}_{\text{prey}}) & \geq 0.125 \\
\lor \text{Distance} (A^{1}_{\text{pred}}, A^{0}_{\text{prey}}) & \geq 0.125 \\
\lor \text{Distance} (A^{2}_{\text{pred}}, A^{0}_{\text{prey}}) & \geq 0.125
\end{align*}
\]

Then the linear constraints are added to the model as following:

\[
\begin{align*}
gmodel.addConstr(delta03 \_x \geq 0.125) \\
gmodel.addConstr(delta03 \_x \leq 0.2) \\
gmodel.addConstr(delta03 \_y \geq 0.125) \\
gmodel.addConstr(delta03 \_y \leq 0.2) \\
gmodel.addConstr(delta13 \_x \geq 0.125) \\
gmodel.addConstr(delta13 \_x \leq 0.2) \\
gmodel.addConstr(delta13 \_y \geq 0.125)
\end{align*}
\]
whether that the prey can be caught by the predator within
k steps where the predators are around the prey. We checked
reachability analysis for S1 specification in scenario 1.

Reachability analysis for S2 specification in scenario 1.

We fix a set of initial states with \( obs_1.p\_pos.x = [-0.5, -0.3] \), \( obs_1.p\_pos.y = [0.3, 0.5] \), \( obs_2.p\_pos.x = [-0.5, -0.3] \), \( obs_2.p\_pos.y = [0.2, 0.4] \), \( obs_3.p\_pos.x = [-0.4, -0.2] \). This corresponds to states where the predators are around the prey. We checked whether the prey can be caught by the predator within k steps, where k is equal to 5, 10, 15 individually.

From the result in table 2, we can see that there is no strategy for prey that it can escape from the predators. This shows the neural networks work well after training.

**TABLE 1.** Reachability analysis for S1 specification in scenario 1.

| step | Vars number (c,b) | Time(s) | feasible |
|------|------------------|---------|-----------|
| 5    | 8076,2699        | 0.24    | Y         |
| 10   | 16050,5329       | 2.41    | Y         |
| 15   | 24030,7983       | 3.39    | Y         |

**TABLE 2.** Reachability analysis for S2 specification in scenario 1.

| step | Vars number (c,b) | Time(s) | result |
|------|------------------|---------|--------|
| 5    | 16252,5356       | 30.09   | N      |
| 10   | 16254,5356       | 20.02   | N      |
| 15   | 16400,5404       | 73.42   | N      |

S3: Is it ever the case that there exists one prey has been caught from the initial states within k steps.

\[ \exists x (prey, x) \forall y (pred, y) F^{\leq k} d (prey, pred) \leq 0.125 \]

algorithm 3 to transform the mSR formula to constraints, and the disjunctive form result is returned.

\[
\text{Distance}(Agt^0_{pred}, Agt^0_{prey}) \leq 0.125
\]

\[
\bigvee \text{Distance}(Agt^1_{pred}, Agt^0_{prey}) \leq 0.125
\]

\[
\bigvee \text{Distance}(Agt^2_{pred}, Agt^0_{prey}) \leq 0.125
\]

Because the number of prey is 1, we get the same formula with the specification S1.

**B. SCENARIO 2 WITH 4 PREDATORS AND 2 PREYS**

In this scenario, the global states is \( S = (obs_1, obs_2, obs_3, obs_4, obs_5, obs_6) \), where \( obs_1, obs_2, obs_3 \) and \( obs_4 \) are for four predators, \( obs_5 \) and \( obs_6 \) are for two preys. The length of \( obs_1, obs_2, obs_3 \) and \( obs_4 \) is 22, but for \( obs_5 \) and \( obs_6 \) is 20.

First we check the reachability for S1 specification, which can be transformed into the following disjunctive set:

\[
\text{Distance}(Agt^0_{pred}, Agt^0_{prey}) \leq 0.125
\]

We denote by \( (obs_1, obs_2, obs_3, obs_4, obs_5, obs_6) \) the initial state of the environment, and \( (obs_1, obs_2, obs_3, obs_4, obs_5, obs_6) \) the final state of the environment after n time steps. We also set up a similar environment with scenario 1 for S1 specification, where the four predators are at the edge of the land, and the two preys are in the middle. The initial states are \( obs_1.p\_pos.x = [-0.95, -0.75], obs_1.p\_pos.y = [0.7, 0.95], obs_2.p\_pos.x = [-0.95, -0.75], obs_2.p\_pos.y = [-0.95, -0.75], obs_3.p\_pos.x = [0.7, 0.95], obs_3.p\_pos.y = [-0.98, -0.7], obs_4.p\_pos.x = [0.7, 0.95], obs_4.p\_pos.y = [0.7, 0.95], obs_5.p\_pos.x = [-0.01, 0.1], obs_5.p\_pos.y = [0.7, 0.95], obs_6.p\_pos.x = [-0.01, 0.1], obs_6.p\_pos.y = [-0.95, -0.7] \). The results in table 4 show that there always exists a strategy that the two preys are caught by the predators within k steps, where k is equal to 5, 10, 15 individually. For example, when k is equal 5, the relative position of X axis between predator 2 and prey 1 is 0.06328125, and the relative position of X axis between predator 4 and prey 2 is 0.06328125, which are less than 0.125, indicating that S1 specification is satisfied.

We get the same results for S2 specification in scenario 2 as scenario 1. There is no strategy for two preys that they can escape from the predators. We believe this is right because the aim of this task is to catch the preys for predators. It shows the network works well after trained 10000 iterations.

We can’t provide comparisons with other tools because we are not aware of other tools supporting systems of multiple neural agents and role-based strategic properties as we do here.

**VI. RELATED WORKS**

The purpose of reachability analysis is to compute the output reachable set to verify the problem through layer-by-layer analysis. We are not aware of any work addressing the reachability analysis/verification for MARL system. However, the reachability analysis for MAS or single RL system has received much attention over the years. These approaches can be divided into 3 main groups: (1) MILP-based techniques that formulate the verification problem as a mixed integer linear program. For example, [19] presents the verification problem of agent-environment systems against bounded CTL and prove that verification is coNExpTime. Reference [20] presents an ideal mixed-integer programming (MIP)
TABLE 3. Reachability analysis for S3 specification in scenario 1.

| step | Vars number (c,b) | Time(s) | result |
|------|------------------|---------|--------|
| 5    | 8076,2699        | 0.24    | Y      |
| 10   | 16050,5329       | 2.41    | Y      |
| 15   | 24030,7983       | 3.59    | Y      |

TABLE 4. Reachability analysis for S1 specification in Scenario 2.

| step | Vars number (c,b) | Time(s) | result |
|------|------------------|---------|--------|
| 5    | 13482,4010       | 9.65    | Y      |
| 10   | 25444,8000       | 36.58   | Y      |
| 15   | 37414,11990      | 80.79   | Y      |

TABLE 5. Reachability analysis for S2 specification in Scenario 2.

| step | Vars number (c,b) | Time(s) | result |
|------|------------------|---------|--------|
| 5    | 13490,4096       | 4.96    | N      |
| 10   | 25460,7996       | 42.41   | N      |
| 15   | 37430,11986      | 91.03   | N      |

formulation for a ReLU appearing in a trained neural network, and this method is much more computationally efficient and scalable than the extended formulation. Reference [21] present an efficient implementation of MILP verifier for properties of piecewise-linear feed-forward neural networks, and this method improves performance by several orders of magnitude when compared to a naïve MILP implementation. Reference [6] is also a MILP-based technique that verify automatically for the single agent RL system. (2) SMT-based techniques that encode the verification problem as the satisfiability modulo theory problem. Reference [22] partitions the input domain into small grid cells, and loosely approximates the reachable set for each grid cell considering the maximum sensitivity of the network at each grid cell. It works for any activation function and its runtime scales well with the number of layers but poorly with the input dimension. Reference [23] present Marabou, a framework for verifying deep neural networks. Marabou is an SMT-based tool that can answer queries about a network’s properties by transforming these queries into constraint satisfaction problems. In [4], an SMT solver named Reluplex is proposed for a special class of neural networks with ReLU activation functions. (3) other techniques get a definite answer. Reference [24] computes the exact reachable set for networks with only ReLU activations. As there is no over-approximation, this method is sound and complete. However, because the number of polytopes grows exponentially with each layer, this method does not scale. Reference [25] investigate model checking algorithms for variants of SL over pushdown multi-agent systems, this algorithms are automata-theoretic. Reference [26] describe various properties of activation functions using Quadratic Constraints (QCs) and develop a novel framework based on semidefinite programming (SDP) for safety verification and robustness analysis of neural networks against norm-bounded perturbations in their input.

VII. CONCLUSION AND FUTURE WORK

With the deployment of systems based on MARL there has been a growing interest in their verification. While the benefits of formal methods have long been recognized, and they have found large adoption in safety-critical systems as well as in industrial scale software, but little research has been done to verify the MARL system.

In this article we put forward a formal methodology to solve the reachability problem for MARL system. First the reachability property is described by a different logic mrSL, then the property is translated into a union of a number of constraints, finally reachable set computation algorithm MMSR is used in Gurobi, and the result feasible or in feasible is returned.

In future work we plan to explore other methodologies, including introducing parameters for mrSL to make verification more flexible, job partition to improve the computing speed. Furthermore, we believe the technique here put forward can be used as an ideal stepping stone to verify MARL system.

REFERENCES

[1] M. Kwiatkowska, G. Norman, and D. Parker, “Verification and control of turn-based probabilistic real-time games,” in The Art of Modelling Computational Systems: A Journey From Logic and Concurrency to Security and Privacy (Lecture Notes in Computer Science), vol. 11760. Springer, 2019, pp. 379–396.
[2] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri, “NuSMV: A new symbolic model checker,” in Proc. Int. Conf. Comput. Aided Verification, 2003, pp. 495–499.
[3] K. Djivokham, R. Stanforth, S. Gowal, T. Mann, and P. Kohli, “A dual approach to scalable verification of deep networks,” in Proc. 34th Ann. Conf. Uncertainty Artif. Intell. (UAI). Monterey, CA, USA: AUAI Press, 2018, pp. 162–171.
[4] S. Wang, K. Pei, J. Whitehouse, J. Yang, and S. Jana, “Formal security analysis of neural networks using symbolic invariants,” in Proc. 27th USENIX Secur. Symp. (USENIX), 2018, pp. 1599–1614.
[5] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, “Efficient neural network robustness certification with general activation functions,” in Proc. 31st Ann. Conf. Neural Inf. Process. Syst. (NeurIPS), 2018, pp. 4944–4953.
[6] M. Akintunde, A. Lomuscio, L. Magnani, and E. Pirovano, “Reachability analysis for neural agent-environment systems,” in Proc. 16th Int. Conf. Princ. Knowl. Represent. Reasoning (KR), 2018, pp. 184–193.
[7] P. Kouvaros and A. Lomuscio, “Parameterised verification for multi-agent systems,” Artif. Intell., vol. 234, pp. 152–189, May 2016.
[8] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, “Safety verification of deep neural networks,” in Computer Aided Verification, Berlin, Germany: Springer, 2017, pp. 3–29.
[9] G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer, “Reluplex: An efficient SMT solver for verifying deep neural networks,” 2017, arXiv:1702.01135. [Online]. Available: http://arxiv.org/abs/1702.01135
[10] W. Ruan, M. Wu, Y. Sun, X. Huang, D. Kroening, and M. Kwiatkowska, “Global robustness evaluation of deep neural networks with provable guarantees for the L_0 norm,” 2018, arXiv:1804.05805. [Online]. Available: http://arxiv.org/abs/1804.05805
[11] A. Lomuscio and L. Maganti, “An approach to reachability analysis for feed-forward ReLU neural networks,” CoRR, vol. abs/1709.09130, pp. 1–10, Jun. 2017.
[12] S. Dutta, S. Jha, S. Sanakaranarayanan, and A. Tiwari, “Output range analysis for deep neural networks,” 2017, arXiv:1709.09130. [Online]. Available: http://arxiv.org/abs/1709.09130
XIAOYAN WANG was born in Changchun, Jilin, China, in 1977. She received the B.S. degree in computer science and technology, the M.S. degree in software engineering, and the Ph.D. degree in computer architecture from Jilin University, in 2000, 2003, and 2008, respectively.

From 2010 to 2011, she was a Visiting Scholar with West Virginia University. She is the author of one book, more than 20 articles, and three inventions. Her research interests include model checking, software engineering, big data, and deep machine learning.

JUN PENG was born in Chongqing, China, in 1981. He received the B.S. degree in computer science and technology, the M.S. degree in software engineering, and the Ph.D. degree in computer architecture from Jilin University, in 2004, 2006, and 2010, respectively.

Since 2011, he has been a Lecturer with the College of Computer Science and Technology, Jilin University. He is the author of one book, more than ten articles, and five inventions. His research interests include computer architecture, software engineering, software design and develop patterns, model checking and validation, machine learning, and deep learning.

SHUQU LI was born in Heilongjiang, China, in 1966. He received the B.S. degree in computer science and technology from Zhejiang University, in 1989, and the M.S. degree in optical instrument major from the Changchun Institute of Optics, Fine Mechanics and Physics (CIOMP), in 1994.

Since 2001, he has been an Associate Professor with the College of Computer Science and Technology, Jilin University. He is the author of one book and more than 15 articles. His research interests include software engineering, software design and develop patterns, model checking and validation, machine learning, and deep learning.

BING LI received the B.Sc., M.Sc., and Ph.D. degrees from Jilin University, in 1999, 2005, and 2011, respectively. He is currently a Lecturer in computer science with Jilin University. His main research interests include network security, software engineering, and machine learning.