ADT2AMAS: Managing Agents in Attack-Defence Scenarios

Demonstration Track

Jaime Arias
LIPN, CNRS UMR 7030, Université Sorbonne Paris Nord
Villetaneuse, France
arias@lipn.univ-paris13.fr

Wojciech Penczek
Institute of Computer Science, Polish Academy of Sciences
Warsaw, Poland
penczek@ipipan.waw.pl

Laure Petrucci
LIPN, CNRS UMR 7030, Université Sorbonne Paris Nord
Villetaneuse, France
petrucci@lipn.univ-paris13.fr

Teofil Sidoruk
[1]Institute of Computer Science, PAS
[2]Warsaw University of Technology
t.sidoruk@ipipan.waw.pl

ABSTRACT

Expressing attack-defence trees (ADTrees) in a multi-agent setting allows for studying a new aspect of security scenarios, namely how the number of agents and their task assignment impact the performance of attacking and defending strategies executed by agent coalitions. Our tool ADT2AMAS allows for transforming ADTrees into extended asynchronous multi-agent systems and computing an optimal schedule with the minimal number of agents. ADT2AMAS is integrated within the graphical verification platform CosyVerif, but can also be run standalone.

KEYWORDS

asynchronous multi-agent systems; scheduling; attack-defence trees

ACM Reference Format:
Jaime Arias, Wojciech Penczek, Laure Petrucci, and Teofil Sidoruk. 2021. ADT2AMAS: Managing Agents in Attack-Defence Scenarios: Demonstration Track. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 3 pages.

1 INTRODUCTION AND MOTIVATIONS

Security of safety-critical multi-agent systems [24] is one of the major challenges. Attack-defence trees (ADTrees) [8, 15] have been developed to evaluate the safety of systems and to study interactions between attacker and defender parties. They provide a simple graphical formalism, where nodes represent possible attacks against a system and defences employed to protect it. However, the defenders might forgo a costly defence, or a defence could take too much time and thus fail. The attack to be achieved is the root one, which requires all (for the AND nodes) or part (for the OR nodes) of those in its subtrees to be successful.

Recently, it has been proposed to model ADTrees in the formalism of asynchronous multi-agent systems (AMAS) extended with certain ADTree characteristics [4, 21]. In this setting, one can reason about attack/defence scenarios considering agent distributions over the tree nodes and their impact on the feasibility and performance (quantified by metrics such as time and cost) of attacking and defending strategies executed by specific agent coalitions. Both the number of agents available and their distribution over the ADTree nodes affect these performance metrics. Hence, there arises the problem of an optimal scheduling, that is, obtaining an assignment that guarantees the lowest possible execution time while using the minimum number of agents required for an attack to be feasible. The tool ADT2AMAS implements the algorithm of [7] that finds such an assignment for a given ADTree. As the ADTree formalism introduces unique caveats, our approach differs from that of classical process scheduling [18, 19, 22]; to the best of our knowledge, this is the first work dealing with agents in this context. Notably, the scheduling algorithm optimises both the number of agents and the attack time, and runs in quadratic time wrt. the number of nodes. However, it is applied to a number of models that is exponential in the number of OR and defence nodes.

This work is supported by the IEA project PARTIES and FNR/NCBiR project STV.

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online.

2 FORMAL BACKGROUND

A well-known formalism [10, 11, 14, 16, 17, 20, 23], ADTrees model security scenarios as an interplay between two opposing parties. Figure 1 depicts basic constructs, [4] gives a comprehensive overview.

Attacking and defending actions are depicted in red and green,

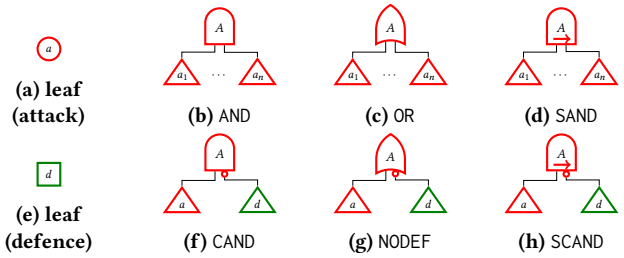
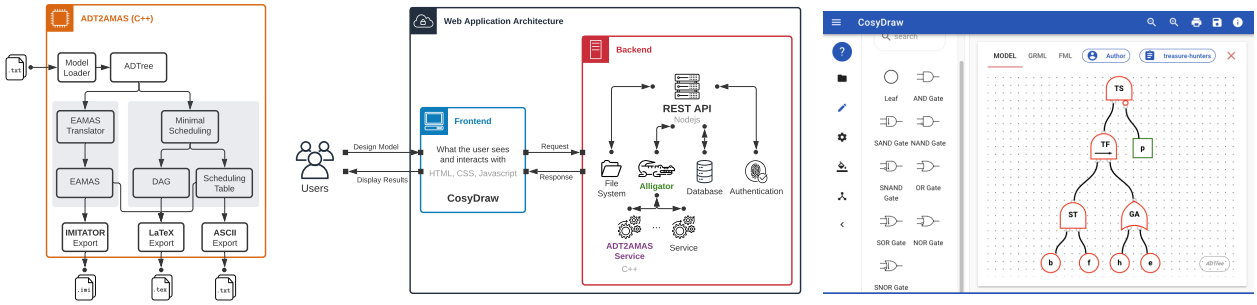


Figure 1: Basic ADTree gates (or nodes)

respectively. The leaves represent individual actions at the highest level of granularity. Different types of gates (available in both attack or defence types) allow for modeling increasingly broad intermediary goals, all the way up to the root, which corresponds to the overall objective. OR and AND gates are defined analogously to their logical counterparts. SAND is a sequential variant of the latter, *i.e.* the entire subtree a_i needs to be completed before handling a_{i+1}. Countering actions can be expressed using gates CAND (counter defence; successful iff a succeeds and d fails), NODEF (no defence; successful iff either a succeeds or d fails), and SCAND (failed reactive defence; sequential variant of CAND, where a occurs first). ADTree nodes may have numerical *attributes*, *e.g.* the time needed to perform an attack, or its financial cost. *Conditions* are boolean functions over the attributes, used as constraints to counter-defence nodes.



(a) Architecture of ADT2AMAS **(b) Architecture of the web-based application** **(c)** treasure hunters **ADTree in CosyDraw**

Figure 2: Graphical Interface for ADT2AMAS

The translation of ADTrees to extended asynchronous multi-agent systems (EAMAS, [4, 12, 13]), called also *models*, was proposed in [4, 21]. To that end, the semantics of an asynchronous MAS of [12, 13] was extended to account for the node attributes and conditional constraints. Each ADTree node corresponds to a separate automaton, with specific patterns used for different types of nodes. Transitions are decorated with actions and the following elements: (i) a *message* $f_M(t) \in (\{!, ?\} \times M) \cup \{\bot\}$ indicating whether there is a synchronisation on a message or the action is local; (ii) a *guard* $f_G(t) \in Guards$ constraining the transition; (iii) an *update function* $f_t : AT_t \to EXP(AT, FUN)$ expressing how the transition modifies the attributes. In this setting, groups of agents working for the attacking and defending parties can be considered. Note that the *feasibility* of an attack is not affected by the number or distribution of agents over ADTree nodes, while some *performance* metrics, such as time, clearly depend on both factors (*e.g.* a lone agent can sequentially handle all the actions, albeit usually much slower). Furthermore, the success or failure of particular defence nodes results in a number of unique configurations, leading to multiple model variants to be analysed. Consequently, the optimal distribution of agent coalitions is of vital importance for both parties, allowing them to prepare for multiple scenarios, depending on how many agents they can afford to recruit. Thus, synthesising an assignment that achieves a minimal execution time using the least possible number of agents is a problem of high interest and importance.

3 ARCHITECTURE AND TECHNOLOGY

ADT2AMAS [5] is an open source tool written in C++17. It allows for: (1) translating an ADTree into an EAMAS, and (2) computing a minimal scheduling with a minimal number of agents to make the root node's task successful. The ADTree can be specified using the provided API, or loaded from a simple-syntax text file (Figure 2a).

For (1), the tool generates a LaTeX file of the EAMAS, and an `.imi` file containing its specification, which can be read by the model-checker IMITATOR [3] in order to run automatic verifications. For (2), ADT2AMAS generates a LaTeX file of each tree transformation needed to compute the minimal assignment, and an ASCII table of the minimal scheduling. Compiling LaTeX to PDF provides a visual feedback of the EAMAS and of the scheduling algorithm's steps.

ADT2AMAS is also integrated within the CosyVerif formal specification and verification environment [2], thus providing a multi-platform, user-friendly, zero-configuration tool (Figure 2b). Its JavaScript web interface, CosyDraw, allows for designing formal models extensible via FML (Formalism Markup Language) and GrML (Graph Markup Language) specifications. Furthermore, it supports any tool providing a SOAP web service. In our case, we extended the graphical interface with the ADTree elements (Figure 2c) and implemented a web service offering the algorithms supported by ADT2AMAS [6]. The tool can be accessed at https://cosyverif.lipn.univ-paris13.fr, and its video demonstration at https://youtu.be/DGLtUSP-ao8.

4 EXPERIMENTS

Selected experimental results are provided in Table 1, with additional ones available on the tool web page [1]. Subsequent columns denote: (1) the case study, (2) the number of possible defence configurations, which influence the number of models to consider, (3) the number of a model variant; note that some combinations of defences result in the same model, or do not allow for a successful attack, (4) the minimal number of agents obtained by our scheduling algorithm, (5) the minimal attack time. The tool also provides a corresponding schedule of agents' actions (not shown here), and generates the resulting EAMAS (currently, for 1 agent only), which can then be passed to the IMITATOR model checker. This allows for the verification of other desired properties (*e.g.* a model with a higher attack time might be preferable due to its far lower cost).

Table 1: Experimental results on case studies

case study	# def.	no. of model	# agents	time
treasure	2	1	2	125 min
forestall	4	1	1	43 days
		2	1	54 days
		3	1	53 days
iot-dev	4	1	2	694 min
gain-admin	16	1	1	2942 min
		2	1	4320 min
		3	1	5762 min
scaling	1	1	6	5 min

5 FUTURE WORK

Currently, model specifications (`.imi` files) including more than one agent are prepared by hand. We plan to improve the integration of ADT2AMAS with IMITATOR by a fully automatic generation of EAMAS for an arbitrary number of agents. Furthermore, ADT2AMAS can be integrated with other model checkers such as UPPAAL [9].

REFERENCES

[1] [n.d.]. ADT2AMAS Minimal Scheduling. https://depot.lipn.univ-paris13.fr/parties/publications/minimal-scheduling.

[2] [n.d.]. CosyVerif. https://www.cosyverif.org.

[3] [n.d.]. IMITATOR. https://www.imitator.fr.

[4] Jaime Arias, Carlos E. Budde, Wojciech Penczek, Laure Petrucci, Teofil Sidoruk, and Mariëlle Stoelinga. 2020. Hackers vs. Security: Attack-Defence Trees as Asynchronous Multi-agent Systems. In *22nd International Conference on Formal Engineering Methods, ICFEM 2020, Singapore, Singapore, March 1-3, 2021 (LNCS)*, Vol. 12531. Springer, 3–19. https://doi.org/10.1007/978-3-030-63406-3_1

[5] Jaime Arias, Wojciech Penczek, Laure Petrucci, and Teofil Sidoruk. [n.d.]. ADT2AMAS. https://depot.lipn.univ-paris13.fr/parties/tools/adt2amas.

[6] Jaime Arias, Wojciech Penczek, Laure Petrucci, and Teofil Sidoruk. [n.d.]. ADT2AMAS Alligator Service. https://depot.lipn.univ-paris13.fr/cosyverif/services/service-adt2amas.

[7] Jaime Arias, Laure Petrucci, Wojciech Penczek, and Teofil Sidoruk. 2021. Minimal Schedule with Minimal Number of Agents in Attack-Defence Trees. https://arxiv.org/abs/2101.06838

[8] Zaruhi Aslanyan and Flemming Nielson. 2015. Pareto Efficient Solutions of Attack-Defence Trees. In *Proceedings of the 4th Conference on Principles of Security and Trust, POST 2015, London, UK, April 11-18, 2015*. Springer, 95–114.

[9] Gerd Behrmann, Alexandre David, Kim Guldstrand Larsen, John Håkansson, Paul Pettersson, Wang Yi, and Martijn Hendriks. 2006. UPPAAL 4.0. In *Third International Conference on the Quantitative Evaluation of Systems (QEST 2006), 11-14 September 2006, Riverside, California, USA*. IEEE Computer Society, 125–126.

[10] Ahto Buldas, Peeter Laud, Jaan Priisalu, Märt Saarepera, and Jan Willemson. 2006. Rational Choice of Security Measures Via Multi-parameter Attack Trees. In *Critical Information Infrastructures Security*. Springer, 235–248.

[11] Barbara Fila and Wojciech Widel. 2020. Exploiting Attack-Defense Trees to Find an Optimal Set of Countermeasures. In *Proceedings of the 33rd IEEE Computer Security Foundations Symposium, CSF 2020, Boston, MA, USA, June 22-26, 2020*. IEEE, 395–410.

[12] Wojciech Jamroga, Wojciech Penczek, Piotr Dembinski, and Antoni Mazurkiewicz. 2018. Towards Partial Order Reductions for Strategic Ability. In *Proceedings of the 17th International Conference on Autonomous Agents and Multi-agent Systems, AAMAS '18, Stockholm, Sweden, July 10-15, 2018*. ACM, 156–165.

[13] Wojciech Jamroga, Wojciech Penczek, Teofil Sidoruk, Piotr Dembinski, and Antoni W. Mazurkiewicz. 2020. Towards Partial Order Reductions for Strategic Ability. *J. Artif. Intell. Res.* 68 (2020), 817–850. https://doi.org/10.1613/jair.1.11936

[14] Ravi Jhawar, Barbara Kordy, Sjouke Mauw, Saša Radomirović, and Rolando Trujillo-Rasua. 2015. Attack Trees with Sequential Conjunction. In *ICT Systems Security and Privacy Protection*. Springer, 339–353.

[15] Barbara Kordy, Sjouke Mauw, Saša Radomirović, and Patric Schweitzer. 2011. Foundations of Attack-Defense Trees. In *FAST 2010 (LNCS)*, Vol. 6561. Springer, 80–95.

[16] Barbara Kordy, Ludovic Piètre-Cambacédès, and Patrick Schweitzer. 2014. DAG-based Attack and Defense Modeling: Don't Miss the Forest for the Attack Trees. *Computer Science Review* 13-14 (2014), 1–38.

[17] Rajesh Kumar, Stefano Schivo, Enno Ruijters, Buğra Mehmet Yildiz, David Huistra, Jacco Brandt, Arend Rensink, and Mariëlle Stoelinga. 2018. Effective Analysis of Attack Trees: A Model-Driven Approach. In *Fundamental Approaches to Software Engineering*. Springer, 56–73.

[18] Yu-Kwong Kwok and Ishfaq Ahmad. 1999. Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors. *ACM Computing Surveys* 31, 4 (1999), 406–471.

[19] Michele Lombardi, Michela Milano, Martino Ruggiero, and Luca Benini. 2010. Stochastic allocation and scheduling for conditional task graphs in multi-processor systems-on-chip. *J. Sched.* 13, 4 (2010), 315–345.

[20] Sjouke Mauw and Martijn Oostdijk. 2006. Foundations of Attack Trees. In *ICISC 2005*. Springer, 186–198.

[21] Laure Petrucci, Michal Knapik, Wojciech Penczek, and Teofil Sidoruk. 2019. Squeezing State Spaces of (Attack-Defence) Trees. In *24th International Conference on Engineering of Complex Computer Systems, ICECCS 2019, Guangzhou, China, November 10-13, 2019*. IEEE, 71–80.

[22] Khushboo Singh, Mahfooz Alam, and Sushil Kumar. 2015. A Survey of Static Scheduling Algorithm for Distributed Computing System. *International Journal of Computer Applications* 129 (11 2015), 25–30.

[23] Chris Slater, O. Sami Saydjari, Bruce Schneier, and Jim Wallner. 1998. Toward a Secure System Engineering Methodolgy. In *NSPW'98*. ACM, 2–10. https://doi.org/10.1145/310889.310900

[24] Michael J. Wooldridge. 2002. *An Introduction to Multiagent Systems*. John Wiley & Sons.