ADT2AMAS: Managing Agents in Attack-Defence Scenarios
Jaime Arias, Laure Petrucci, Wojciech Penczek, Teofil Sidoruk

LIPN, CNRS UMR 7030, Université Sorbonne Paris Nord
{arias, petrucci}@lipn.univ-paris13.fr

Institute of Computer Science, Polish Academy of Sciences
{penczek, t.sidoruk}@ipipan.waw.pl

Attack-Defence Scenarios in a Multi-agent Setting

- **Idea**: translation from attack-defence trees (ADTrees) into asynchronous multi-agent systems (AMAS)
- New aspect of security scenarios: agent coalitions of various size and action assignment can be considered
- Qualitative and quantitative analysis using existing methods and tools developed for multi-agent systems

ADTree to AMAS Translation

- **EAMAS**: AMAS semantics extended with attributes and functions to model ADTrees
- Each ADTree node corresponds to an automaton in the resulting multi-agent system
- Specific patterns for each ADTree construct, embedding reductions to prevent state explosion
- Further reduction on the level of entire EAMAS: exploiting the topology to avoid some interleavings

The Scheduling Algorithm

- Optimal scheduling of agents’ actions is crucial to the performance (e.g. attack time) in ADTree scenarios
- A relevant and non-trivial scheduling problem: optimizing both attack time and the number of agents
- Time normalisation and preprocessing: input ADTree becomes a DAG, sequences replace SAND gates
- Handling choices: OR and defence nodes induce multiple variants for which to compute the schedule
- Schedule length kept at minimum, extra agents added only if execution impossible without increasing time
- Quadratic complexity in the number of nodes, but an exponential number of OR/defence variants

Assignment is Equally Important to Coalition Size!

<table>
<thead>
<tr>
<th>number of agents</th>
<th>slowest</th>
<th>fastest</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>14</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>16</td>
</tr>
</tbody>
</table>

Our Tool: ADT2AMAS

- Open-source tool written in C++
- Input ADTree: simple-syntax text or a user-generated model in the intuitive web interface CosyVerif
- Intermediary steps of the scheduling algorithm visualized with generated \( \text{LTEx} \) files
- Output: minimal schedule using the fewest agents
- Also generates models for verification with IMITATOR

Web App Architecture

ADT2AMAS Architecture

Summary: Our Contribution

- Unified/extended scheme for ADTree representation
- Formal semantics of EAMAS to model ADTrees
- ADTree to EAMAS pattern transformation rules
- Translation and optimal scheduling with ADT2AMAS
- Agent coalitions: study of performance metrics impact
- Parametric synthesis of ADTree attributes in IMITATOR

References

- ADT2AMAS. https://lipn.univ-paris13.fr/adt2amas/