Final Report
Multi-Agent Programming Contest

Rahul Arora, Artur Daudrich, Sergey Dedukh, Michael Ruster, Michael Sewell, and Yuan Sun

University Koblenz-Landau

1 Motivation

2 Scientific Background and Fundamentals

2.1 MAPC: Contest and Scenario

2.2 Agent Programming Concepts

BDI.

Formal Methods.

Negotiation and Argumentation.

Agent Societies.

2.3 Agent Programming Languages

GOLOG and FLUX.

Jadex.

AgentSpeak (L) and Jason. Why are AS(L) and Jason so awesome that we chose them over the other options? Why didn't we invent our own language or at least our own system/architecture/infrastructure?

3 Team Organisation

3.1 Structure and Meetings

Dynamic working groups that were built weekly to tackle the newly crafted tasks per week.

3.2 Git, Hangouts and Skype

Revisionsing system, Wiki for minutes and issues for problems. VoIP-solutions for collaborative programming.

4 Architectural (?) Structure

4.1 Agents

Talk a bit about generalisation e.g. a saboteur is a specialisation of an agent. I.e. both share exploring but the saboteur also knows how and when to attack. Explain what tasks our agents have and where our priorities are.

4.2 Simulation Phases

Explain the general split up into an exploration and a zoning phase.

5 Algorithms and Strategies

5.1 General Strategy Overview

This could also be an introductionary text which motivates the following subsections.

5.2 DSDV

What is it? How is it used in our context? What are advantages we gain from it? What is problematic (speed loss)?

5.3 Exploration

How do agents move around during the exploration phase?

5.4 Zone Calculation

This can also be dealt with in DSDV already but then with subsections.

5.5 Zone Forming

5.6 Zone Extensions and Breakups

6 Implementation Details

6.1 BDI in AS(L) and Jason

Or in general: how did we implement what we had learnt from our scientific background?

6.2 Information Flow

Who gets what information how and when? How do we communicate with the server?

6.3 Lifecycle of one Step

Maybe illustrate what happens within one step and how we prevent multiple actions to be executed in one step.

6.4 Lessons Learned

Here we could explain what was working well and what was troublesome. Java: fast. AS(L): slow and hard for us to program. Communication: extreme bottleneck.

7 Conclusion