# RDF graph synchronization for collaborative robotics – Progress report

ADAM FREJ

PATRYK TOMASZEWSKI

# Goal of the system

Synchronise knowledge graphs between multiple agents in real-time, while each agent continuously appends new data, and while also handling issues typically present in wireless communication.

By:

Adding knowledge graph versioning

Implementing merging mechanism

Designing and implementing communication protocols for graph synchronisation

# Main research question

Is the system proposed in the paper "RGS⊕: RDF graph synchronization for collaborative robotics" feasible to implement and functional?

# Our application

One role: RDFAgent

One subrole: MergeMasterAgent

Three interactions:

- Status
- Revision
- Revision-request

# Graph storage

A modified version of a knowledge graph that allows for git-like versioning.

Consists of:

- Graph author UUID
- Current state
- Tree of revisions
- Current revision id

# Revision

A single batch (commit) of changes.

Consists of:

- Revision author UUID
- Parents – None (if root), single id or two ids
- Deltas
  - List of added triples
  - List of removed triples
- Locally-timed creation timestamp
- Revision id – hash generated with sha512 on author UUID and timestamp

# Environment

There exists a full knowledge graph of "truth at current time". This graph may change (addition/removal of triples) through time.

Each agent may, at some irregular interval, retrieves a triple (or information about removal of a triple) which is true at a given point of time. The same triple may be retrieved multiple times by different agents.

# Environment – implementation

- Generate *n* triples as ground truth
- On "information uncovering":
  - If random() < mutation chance:
    - Remove a random triple and add a random triple to ground truth
  - If agent contains outdated triples and random < uncover outdated chance:
    - Return "triple removed" for random outdated triple
  - Else:
    - Return "triple added" for random triple from ground truth

# Implementation – RDFAgent

Connecting to other agents – status interaction

Choosing merge master – based on arbitrary rule

Generating local revisions – graph generator

Sending revisions to known agents

Messaging:

- JSON format of the body
- Metadata to distinguish protocols

# Implementation – server

Server – an entity to which all online agents register

Simulation – an entity which runs a given scenario, starts the agents and stops the experiment

# Further work

Merging and rebasing mechanic – main quality of the system

Frontend – visualize agent interactions and graph versioning

Perform failure simulations and stress tests

# Thank you for attention

Feel free to ask questions