# DiC_chpt7

Intelligent agents have to be able to communicate with one another. Since they are independent agents, they respect each other's integrity, meaning that one agent cannot directly access the functionality of another agent, unlike member-functions in object oriented programming. The exception being police-agents stopping and arresting citizen-agents.

It is by sending messages that agents inform each other over events or ask other agents to interact. Examples of the use of messages between agents in the project are for example:

- citizen agents asking other citizen agents if they want to meet or do something together
- citizen agents communicating over social media with other agents
- citizen agents informing other citizen agents about ongoing protests or violations
- police-agents asking other police-agents for help

## A. Questions regarding the theory in the textbook

Referring to Chapter 7 in the textbook according to the reading instructions, see course activities week 16.

### A.1 Studying the list in table 7.3 of performatives provided by the FIPA communication language: - which performatives could be useful in our porject and in which ways?

Accept proposal could be used in a plethora of ways in our project, we want the citizens to be able to organise riots and other activities so for stuff like that the accept proposal would be a given that it has use in our project.

The cancel performative could also be put into the project so that a citizen who agrees to rioting can back out if something beneficial happens to it, however this is not something that has been discussed during classes but it could be used in the project if the project flow is good.

The inform performative also has an obvious use case in the form of talking to colleagues at work about pay wages since we want them to talk about salary and it to affect the workers happiness.

The inform-if version of this could also be used in the same fashion but add compellability to the program so that a happy worker doesn’t perhaps give out information about its salary if it is happy and that only the upset workers go around Charing this information.

Inform ref also has the same use case but needs to be implemented a bit differently by perhaps asking citizens with only higher patches and depending on how many respond they get affected more or less by anger.

Not-understood Is a good precaution to use in the project to make sure things work as they should, according to the text it is a central error handling mechanism in FIPA ACL so should definitely be implemented in the project.

Propagate sounds like a good option for he social behaviours we intend to implement in the project so that someone broadcasting to their friends send a propagating message to all their friends.

Proxy is also a good performative for our project because it will allow us to spread propaganda and such

Query if could also be used to spread information about riots and such

Refuse is a given for protest and converting into a anarchist and such

Reject proposal could also be used if we want the agents to know why the other agent refused.

Request is a big part of FIPA ACL but I don't really see any use for it since our agents wont be asking favours and for others to perform actions for them.

CFP can also be used for communicating to others to create some sort of riot

# B. Questions regarding research articles/the other research material

Referring to the articles that describe the NetLogo extension for communication.

### B.1 What is the message format proposed by the articles? In which ways does it fulfill the FIPA-ACL or not?

The article proposes that an agent appears then listens if any incoming cfp have came I, if they have gotten one they check their stock and send back a propose with quantity and price. It not enough items in the stock they send back a refuse.

They also use accept proposal so agree to the offer which leads to stock changing appropriately

In the other article we see a code example for the structure of a message and they are missing parts that FIPA specifies however this is due to the fact that the author deemed them unnecessary and they also have

implemented the custom containers and these can be used to add whatever is needed (ontology and language).

**B.2 Write NetLogo pseudo-code using the functions defined by communication.nls for:**

A. **a single agent writing a inform-message to another single agent**

let msg create-message “inform”

set msg sender self msg

set msg add-reciever agent1 msg

set msg add-content “blab la bla “ msg

send msg

.....

Send add-content “meddelande” add-reciever agent1

Create-message “inform”

Both create and send a inform message type with message bla blab la or meddlande to agent1.

B. **a single agent writing a request-message to a list of other agents**

let msg create-message “request”

set msg sender self msg

set msg add-multiple-reciever agents_list msg

set msg add-content “meddelande” msg

send msg

C. **a list of agents answering to a request-message**

D. **a single agent sending an inform message to all the agents**

```
let msg create-message “inform”

set msg add-sender self msg

broadcast-to agent_breed msg self
```

# C. Questions regarding the implementation in the project

Look at the implementation of sending and receiving messages in the NetLogo extension 'communication.nls'.

Within our hybrid agent architecture, messages should not just be sent, but be put as intentions on the intention stack. Even the task to read and answer to messages should be put on the intention stack.

### C.1 Explain why it is important that writing and reading messages are placed as intentions on the intention-stack, and what the problem could be if agents directly send and answer messages.

We need to place the messages on the intentionstack so that we don’t interrupt any important processes, the stack is a higherarchy of what needs to be done and it follows the order of the code, having it so that messages get instantaneous responses would result in them acting like uncontrolled interrupts in c code and this can be devastating for a program if left unchecked. Keeping them on the stack means we execute our code in th intender order and things our agent needs to prioritize will be prioritized instead of responding to a message.

### C.2 Write and test the code for putting the sending and reading of messages onto the intention stack.

```
To collect-msg-update-intentions
    Let msg []
    Let performative []
    While[not-empty? Incoming-queue]
        [set msg get-message
        Set performative get-performative msg
```

```
        if performative = “accept-proposal” [reply-to-accept-proposal msg]
        if performative = “cfp” [reply-to-cfp msg]
        if performative = “reject-proposal” [do-nothing]
end
to reply-to-cfp [msg]
    ifelse current-intention = “go-to-town-square”
    [let dist distancexy get-content msg
    Send add-content dist create-reply “propose” msg
    [send add-content “busy” create-reply “refuse” msg
End
To reply-to-accept-proposal [msg]
    Ifelse current-intention = “go-to-town-square”
    [let crds get-content msg
    Add-belief msg
    Add-intention “send-confirmation” “true”
    Add-intention “preaching-done” “preaching-satisfied”
    Add-intention (word “move-towards-town-square “ crds) (word “at-town-square” crds)]
    [send add-content get-content msg create-reply “failure” msg]
End
```

This code is taken from the article and checks all messages in the queue as long as its not empty and looks at the performatives and does appropriate action for the messages using If statements.

The reoly functions check current intentions and depending on if the statement is true the agent either refuses or proposes.

The reply to accept proposal checks current intention again and looks at the message, it creates a belief of the msg and adds intentions to send confirmation and put out the fire in this case, also adds a move towards and at coordinates intention and if this doesn’t go through the function sends out a failure message instead.

In this case we propose going to town square and preaching instead of the original code about putting out a fire.