Recommended reading


Communicating agents in AgentSpeak

We have learned how to use the following performatives:

- Information exchange:
  - tell
  - untell
- Goal delegation:
  - achieve
  - unachieve
- Information seeking:
  - askOne
  - askAll
- Exchanging means (plans):
  - tellHow
  - untellHow
  - askHow

How can we implement the Contract Net Protocol?
### 7b.1 Contract Net Protocol

A UML diagram of the Contract Net Protocol:

![UML diagram](Figure 6.4 Contract net protocol)

(JasonBook page 131)

### CNP in Jason

The Jason implementation can be found in `<Jason_path>/examples/contract-net-protocol`. It has the following MAS file `ContractNetProtocol.mas2j` (shortened a bit):

```mas
MAS cnp {
agents:
  c;  // the CNP initiator
  p #3;  // the participants (3)
  pr;  // a participant that always refuses
  pn;  // a participant that does not answer
}
```

### The simplest participant

The agent `pn` is the simplest participant:

// the name of the agent playing initiator in the CNP
plays(initiator,c).

// send a message to the initiator introducing myself
// as a participant
+plays(initiator,In) : .my_name(Me)
  <- .send(In,tell,introduction(participant,Me)).

Although it registers with initiator, it never send proposals or refusals.
The "always reject" participant

The agent pr:

\[ \text{plays(initiator, c).} \]
\[ +\text{plays(initiator, In)} \]
\[ : \text{.my_name(Me)} \]
\[ <- \text{.send(In, tell, introduction(participant, Me))}. \]

// plan to answer a CFP
\[ +\text{cfp(CNPId, _Service)[source(A)]} \]
\[ : \text{plays(initiator, A)} \]
\[ <- \text{.send(A, tell, refuse(CNPId))}. \]

It registers with initiator and always rejects/refuses proposals.

Answering a CFP

The agents answer a cfp by proposing an offer:

\[ @c1 +\text{cfp(CNPId, Task)[source(A)]} \]
\[ : \text{plays(initiator, A) & price(Task, Offer)} \]
\[ <- // remember my proposal \]
\[ +\text{proposal(CNPId, Task, Offer)}; \]
\[ .\text{send(A, tell, propose(CNPId, Offer))}. \]

and they react, when agent c sends an accept message:

\[ @r1 +\text{accept_proposal(CNPId)} \]
\[ : \text{proposal(CNPId, Task, Offer)} \]
\[ <- \text{.print("My proposal ", Offer," won CNP ", CNPID, " for ", Task,"!").} \]
\[ // do the task and report to initiator \]

Starting a CFP

The agent c uses the following plan to start a cfp:

\[ +!\text{startCNP(Id, Object)} \]
\[ <- \text{.wait(2000); // wait participants introduction} \]
\[ +\text{cnp_state(Id, propose); // remember state of CNP} \]
\[ .\text{findall(Name, introduction(participant, Name), LP);} \]
\[ .\text{print("Sending CFP to ", LP);} \]
\[ .\text{send(LP, tell, cfp(Id, Object));} \]
\[ .\text{concat("+!contract("Id,"),Event);} \]
\[ // the deadline of the CNP is now + 4 seconds, \]
\[ // so the event +!contract(Id) is generated \]
\[ // at that time \]
\[ .\text{at("now + 4 seconds", Event).} \]

The .at(...) is important as a timeout to handle the non-responding agent(s)!

CFP procedure

A Call For Proposals (CFP) by agent c is processed by participants as follows:

1. Agent c uses tell performative to send a message cfp that has two arguments:
   - A "unique identification" so that participants can participate in more than one proposal at a time
   - A "task description" telling the agents what needs to be achieved
2. When received by participants, a belief (such as cfp(1, fix(computer))[source(c)] is added to belief base and a (similar) event to the event queue
3. Participants with a plan to react to this event will refuse to participate in the call or propose an offer
4. Agents not responding at all, need to be ignored by agent c ⇒ timeout necessary!