Agent-Based Systems Tutorial 4

Suggested solutions

Michael Rovatsos

Suggestions for solutions and hints are printed in italics below each question

Q1 Design a solution for the vacuum-world example following Brooks' subsumption architecture. Identify how the design needs to change to allow an additional action to charge the robot at the home location whenever it runs low on power. Discuss how this design compares with the design based on deductive reasoning.
Solution suggestions: *Overview of action selection in the subsumption architecture:*

- *Set of all behaviours* $Beh = \{(c,a)|c \subseteq Per$ *and* $a \in Ac\}$
- *Behaviour will fire in state* s *iff* $see(s) \in c$
- *Agent's set of behaviours* $R \subseteq Beh$, **inhibition relation** $\prec \subseteq R \times R$
- $\prec$ *is a strict total ordering (transitive, irreflexive, antisymmetric)*
- *If* $b_1 \prec b_2$, b_1 *will get priority over* b_2
- *Action selection in the subsumption architecture:*

  Function: Action Selection in the Subsumption Architecture
  1. function $action(p : Per) : Ac$
  2. var $fired : \wp(R)$, $selected : A$
  3. begin
  4. $fired \leftarrow \{(c,a)|(c,a) \in R$ *and* $p \in c\}$
  5. for each $(c,a) \in fired$ *do*
  6. if $\neg(\exists(c',a') \in fired$ *such that* $(c',a') \prec (c,a))$ *then*
  7. return a
  8. return $null$
  9. end

The problem can be made a bit more interesting if we allow an additional action "move in random direction" to avoid having to specify conditions for each and every situation and to analyse whether the robot can get stuck.

We consider the following percepts: "There is dirt in the robot current location", and "The robot is facing a wall". The robot's behaviours are the following:
- "*If* detect dirt *then* suck up dirt" $(b1)$
- "*If* detect wall *then* turn" $(b2)$
- "*If* true *then* move in random direction" $(b3)$

We employ the following inhibition relation: $b1 \prec b2 \prec b3$.

To introduce information about the battery state, we introduce the percepts "The robot is low on power" and "The robot is at the home location", and the behaviour "*If* low on power and at the home location *then* charge" $(b4)$. The design of the agent does not enable storing knowledge about where the home position is located (or which actions are necessary in order to reach it). In order to avoid having to randomly search for this position, we introduce an additional signal sent by the home position that weakens as distance from this position increases. We also introduce the behaviour "*If* low on power *then* travel down gradient" $(b5)$. This mechanism is called *gradient field*.

The inhibition relation is modified accordingly: $b4 \prec b5 \prec b1 \prec b2 \prec b3$.

There are several points that can be made comparing the reactive and the deductive reasoning designs. Some key points are the following:

- The reactive design is suitable for this problem, and provides a cheap and robust solution with near optimal performance
- As the complexity of the problem increases it becomes harder to develop and evaluate a reactive solution. This is mainly due to the limitations of the approach in terms of lacking non-local (or long-term) information and explicit communication
- There are mechanisms that can be exploited to overcome such issues as for instance gradient field and ant trail following
- Lack of clear design methodology
- Loss of transparency. Not always straightforward to debug or identify the reasons behind good performance

Q2 Consider an InteRRaP-based design for a multiagent system composed for building a house, consisting of a builder, an architect, and a house owner agent. Describe informally how you would design the layers of each agent and discuss the issues that arise in the process with particular emphasis on the issue of balancing deliberative and reactive behaviour.

Solution suggestions: *(Before going into the details of this exercise it may be useful to quickly go over the main building blocks of the InteRRaP architecture (three layers, upward activation & downward commitment, separation of knowledge bases).) A possible design might look as follows (the builder agent description is the most comprehensive here, you can discuss details in a similar way for the owner and the architect):*

- *Builder agent*
  1. *Behaviour-based layer (BBL): contains low-level behaviours such as stacking bricks on each other, moving building materials, applying simple procedures for putting individual parts together, applying health and safety procedures, identifying material flaws, getting instructions from architect, reporting (e.g. problems) to architect*
  2. *Local planning layer (LPL): plan library (or planning algorithm) for more complex building tasks such as raise wall, building the roof, laying foundations, fitting windows; hierarchical decomposition is necessary to be able to look at different sub-goals independently from each so as to reduce planning complexity*
  3. *Social planning layer (SPL): contains reporting procedures, procedures for getting instructions and advice from architect*

4. *Interaction between layers: agent remains idle unless instructions received from architect are passed from BBL to LPL and then planned for, which causes builder to forward commitments for physical activity to BBL; if problems with LPL execution arise, they are forwarded to SPL and plans for problem reporting/advice-seeking are activated by SPL which provides LPL with a plan for communication (e.g. a specific communication protocol) and LPL then spawns individual messages via BBL; while such communication is underway, other BBL/LPL activity is interrupted, BBL is used to listen to architect's responses, LPL plans next communication steps, may require SPL to generate further plans for social interaction*

– *House owner agent*

1. *Behaviour-based layer: behaviours for making calculations, talking to bank, planning council and architect, going to building site, responding to queries from and giving instructions to architect, making payments to architect and authorities, etc.*
2. *Local planning layer: methods for calculating budget, managing the project (high-level plan for the entire problem)*
3. *Social planning layer: procedures for (re-)negotiating building plans and conditions with architect, reaching agreement with bank, getting planning permission, getting reports from architect, etc.*
4. *Interaction between layers: the whole multiagent system only takes action at the initiative of the house owner, LPL describes things the owner has to do herself, SPL all those procedures which involve communication (control flow similar to builder)*

– *Architect agent*

1. *Behaviour-based layer: basic behaviours for drawing plans, making calculations, buying materials and making payments, receiving payments from owner, talking to owner and builder, visiting the building site, making measurements on site etc.*
2. *Local planning layer: designing plan for house, computing physical properties of building, devising project schedule, generating workplan for builder, preparing proposal for planning council, evaluating builder reports and owner proposals (both technically and financially) etc.*
3. *Social planning layer: giving advice to builder, communicating intermediate and problem reports to owner, spawning re-negotiations if circumstances change, etc.*
4. *Interaction between layers: similar to owner and builder*

Q3 Give an informal definition of the semantics of the following speech acts in terms of the capabilities and mental states of the participating agents:

– $reject(A, B, \varphi)$: A indicates to B that it does not accept that φ is a valid statement

– $refuse(A, B, \alpha)$: A indicates to B that it is not going to perform action α

Distinguish between two different cases: (i) if the two messages are supposed to be used as responses to some request, and (ii) if they can be "standalone" statements.

Solution suggestions: *Although an informal specification is fine as a response to this question, we use modalities such as "can", "believes", "knows" (knows-whether in the sense of*

"has information about") and "intends" for brevity. The informal interpretation of these should be obvious, but the definitions themselves are of course debatable and should generate some discussion.
We first discuss the case $reject(A, B, \varphi)$*:*

- *Standalone:* A *is only going to utter this if it intends* B *to know whether it believes* φ *or* $\neg\varphi$*, and if* A *it actually does not believe* φ*.*
  - *preconditions:* $(\mathsf{Bel}\, A \neg\varphi)$*,* $(\mathsf{Int}\, A (\mathsf{Bel}\, B (\mathsf{Bel}\, A \neg\varphi)))$*,* $(\mathsf{Bel}\, A \neg(\mathsf{Knows}\, B (\mathsf{Bel}\, A \varphi)))$
  - *postconditions:* $(\mathsf{Bel}\, B (\mathsf{Bel}\, A \neg\varphi))$*,* $(\mathsf{Bel}\, A (\mathsf{Bel}\, B (\mathsf{Bel}\, A \neg\varphi)))$*,* $\neg(\mathsf{Int}\, A (\mathsf{Bel}\, B (\mathsf{Bel}\, A \neg\varphi)))$

  Here, you can discuss how $\neg(\mathsf{Int}\ A\ X)$ *is different from* $(\mathsf{Int}\ A\ \neg X)$ *and also under which conditions the "intention removal" postcondition is necessary.*
- *Response (additional conditions to the "standalone" case): The difference here is that there is a (mutually believed) intention of* B *to find out whether* A *believes* φ *or not.*
  - *preconditions:* $(\mathsf{Int}\, B\, (\mathsf{Knows}\, B\, (\mathsf{Bel}\, A\, \varphi)))$*,* $(\mathsf{Bel}\, A\, (\mathsf{Int}\, B\, (\mathsf{Knows}\, B\, (\mathsf{Bel}\, A\, \varphi))))$*,* $(\mathsf{Int}\, A\, (\mathsf{Knows}\, B\, (\mathsf{Bel}\, A\, \varphi)))$
  - *postconditions:* $\neg(\mathsf{Int}\, B\, (\mathsf{Knows}\, B\, (\mathsf{Bel}\, A\, \varphi)))$*,* $(\mathsf{Bel}\, A \neg(\mathsf{Int}\, B\, (\mathsf{Knows}\, B\, (\mathsf{Bel}\, A\, \varphi))))$*,* $(\mathsf{Int}\, A\, (\mathsf{Knows}\, B\, (\mathsf{Bel}\, A\, \varphi)))$

  If the "reject" message is the response to a previous assertion of B *rather than a simple question (this is not clear from the context), an additional condition would be that* B *believes* φ *(otherwise stating* $\neg\varphi$ *would be* A*'s affirmation of what* B *believes).*

The analysis for $refuse(A, B, \alpha)$ is very similar in that most of the inner $(\mathsf{Bel}\ A\ \varphi)$ facts can be replaced by $(\mathsf{Int}\, A\, \alpha)$ (but negation obviously does not carry over for intentions). Also, in modelling the semantics for "refuse" one reason for refusal may be that the agent is not capable of performing the action ($\neg(\mathsf{Can}\ A\ \alpha)$ as a precondition).

Q4 Consider the following FIPA protocol diagram for the Brokering Protocol:

1. Explain its purpose and describe what the admissible message sequences look like that it generates.
2. Describe any problems that could arise during execution of this protocol.
3. Discuss the relationship between such a protocol and the process of (planning-based) means-ends reasoning in agent architectures.

Solution suggestions: *The full specification of the protocol is available from*

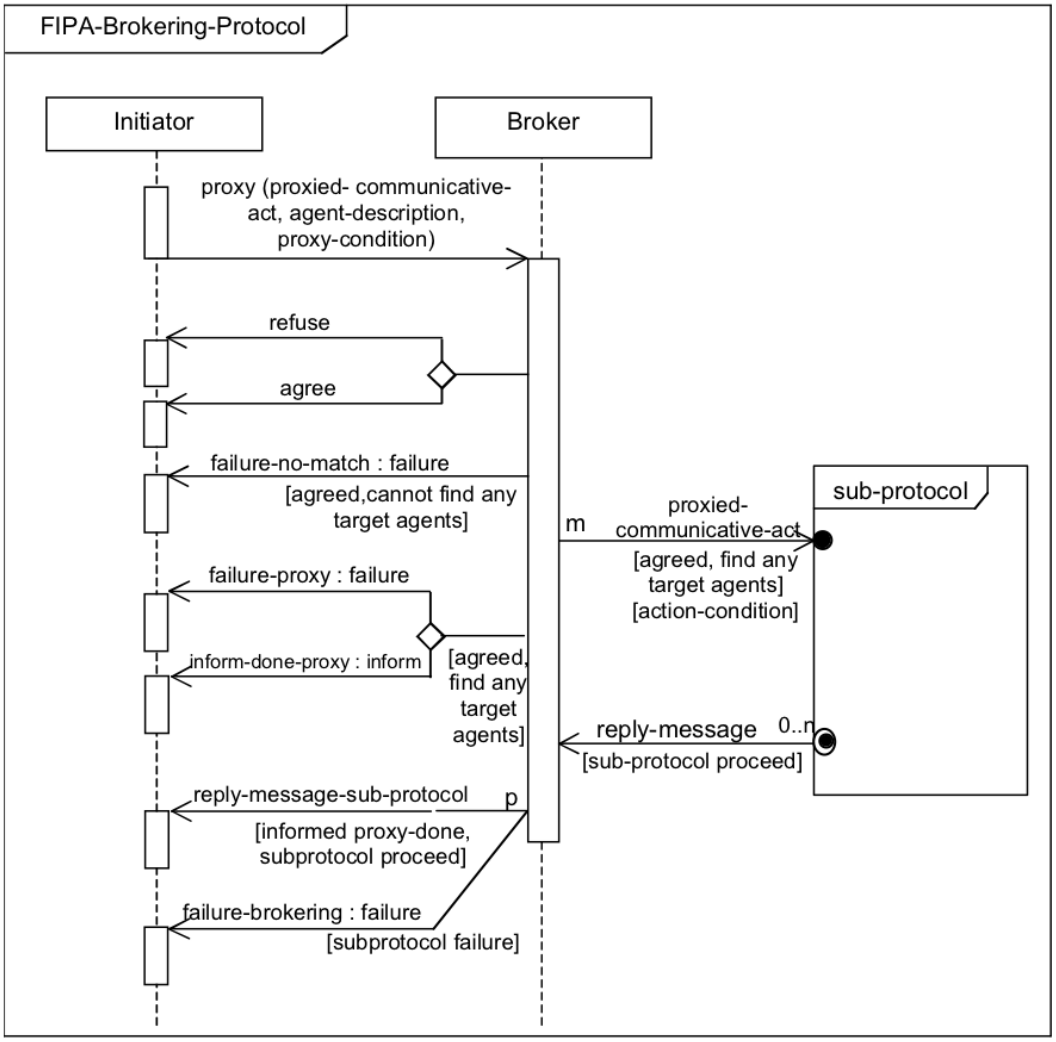
`http://www.fipa.org/specs/fipa00033/SC00033H.html`

Here, we only sketch its central properties (this is a fairly complex example and should illustrate that the meaning of all the elements of such a protocol diagram are not immediately clear).

1. *What the protocol basically does is to cause a broker to search for appropriate proxy agents that can process a certain communicative act. After accepting the request (which it may also reject), the broker may return to the initiator with any of the following responses:*
   - *it found no suitable proxy*

- *the proxy failed to generate an appropriate response in the spawned sub-protocol, or starting the sub-protocol was successful*
- *the forwarded reply of the proxy agent, i.e. the result of the sub-protocol*
- *a failure report, generated if execution of the sub-protocol broke*

2. *One problem is that there are no deadlines for the initiator, i.e. it may wait indefinitely for a response (or, from the broker's perspective, it is unclear for how long it makes sense to look for proxy agents or to execute the sub-protocol since we don't know whether the initiator is still waiting for a response). Also, the diagram is not clear about when a broker may agree or refuse such a request . For example, can it agree even though it doesn't know how to execute the sub-protocol? Another problem is that the proxy agents are never asked for permission to forward their reponses, this may raise security and privacy issues. (Many other similar intricacies can be spotted, if one is sufficiently creative/pedantic . . .)*
3. *Ideally, a protocol captures a set of complex sequences with clear pre- and post-conditions that is internally consistent, so that it can be used by a "communication planning" agent without searching for appropriate actions in each step. Thus, if well-defined, a protocol library is like a plan library for "multiagent plans" facilitated through communication. However, the main problem with this view is that we have to simulate others' mental states, and very often additional communication is necessary to find out what these states are, otherwise we can't plan over several agents' actions – this is also the reason why protocols allow some autonomy and contain different paths of communication. Therefore, a protocol should rather be seen as a disjunction of "possible execution paths" of a plan, where only some of these execution paths actually lead to a goal state for oneself.*


FIPA-Brokering-Protocol
Initiator
Broker
proxy (proxied- communicative-act, agent-description, proxy-condition)
refuse
agree
failure-no-match : failure
[agreed,cannot find any target agents]
failure-proxy : failure
inform-done-proxy : inform
[agreed, find any target agents]
reply-message-sub-protocol
p
[informed proxy-done, subprotocol proceed]
failure-brokering : failure
[subprotocol failure]
m
proxied-communicative-act
[agreed, find any target agents]
[action-condition]
sub-protocol
reply-message
0..n
[sub-protocol proceed]