Extending 3APL with Communication

Jeroen van der Ham
vdham@phil.uu.nl

17th June 2002

Structure of this talk

- Introduction to 3APL
- Introduction to Communication
  - 'Communicative Acts'
  - Message Structure
  - Meaning of Messages
  - Interaction Protocols
- Examples of Communication
- Extending 3APL with Communication
  - Extending the Agents
  - Defining Semantics of Communicative Acts
- Conclusion
- Open Problems and Future Work

Introduction to 3APL

- 3APL = AAAPL = **A**n **A**bstract **A**gent **P**rogramming **L**anguage
- Mental Attitudes: Beliefs & Goals
- Basic Actions
- Dynamic Goal Revision through Practical Reasoning Rules

Definition of an Agent

An agent is defined as the tuple

$$< a, \Pi, \sigma, BAct, \Gamma >$$

With:

- a the name of the agent
- Π the set goals
- σ the set of beliefs
- $BAct$ the set of basic actions
- Γ the set of PR-Rules

Basic Actions and PR Rules

- Definition of Basic Actions

$$\{\ \textit{Precondition}\ \}$$
$$\textit{Action}$$
$$\{\ \textit{Postcondition}\ \}$$

- Definition of Practical Reasoning Rules

$$\textit{Goal-variables} \longleftarrow \quad \textit{Precondition} \ \mid \ \textit{New Goals}$$

Examples

- Examples of a Basic Actions

$$\{\neg on(light)\} \quad \textit{LightOn()} \quad \{on(light)\}$$
$$\{on(light)\} \quad \textit{LightOff()} \quad \{\neg on(light)\}$$

- Example of a Practical Reasoning Rule

$$LightOn(); X \longleftarrow on(light) \mid X; LightOn()$$

- Example of an Agent

GoalBase	BeliefBase
LightOn();LightOff()	$on(light)$
LightOff();LightOn()	$on(light)$
LightOn()	$\neg on(light)$
	$on(light)$

Introduction to Communication

Communicative Act

- Proposition
- Propositional Attitudes
  (Agent, Proposition, Attitude)
  Example: (A, money, want)
- Ontology

Example of a Communicative Act:

```
(inform  :sender    PC-shop
         :receiver  client3507
         :content   (price PC12 2500)
         :ontology  PC-sales
         :language  FIPA-SL)
```

Meaning of an `inform` message

Communicative Act:	inform
Summary:	The sender informs the receiver that a given proposition is true.
Meaning (Sender):	The content of this message is a proposition that the sender believes is true. The sender can only send this message when it believes the proposition, wants the receiving agent to also believe this proposition and does not believe the receiving agent already believes the proposition.
Meaning (Receiver):	The receiver can now believe that the sender holds the proposition for true and also that the sender wishes that the receiver believes the proposition as well. The receiver can decide for itself whether to believe the proposition or not.
Formal model: Preconditions: Effects:	$< i, \text{inform}(j, \varphi) >$ $B_i\varphi \wedge \neg B_i(Bif_j\varphi \vee Uif_j\varphi)$ $B_j\varphi$

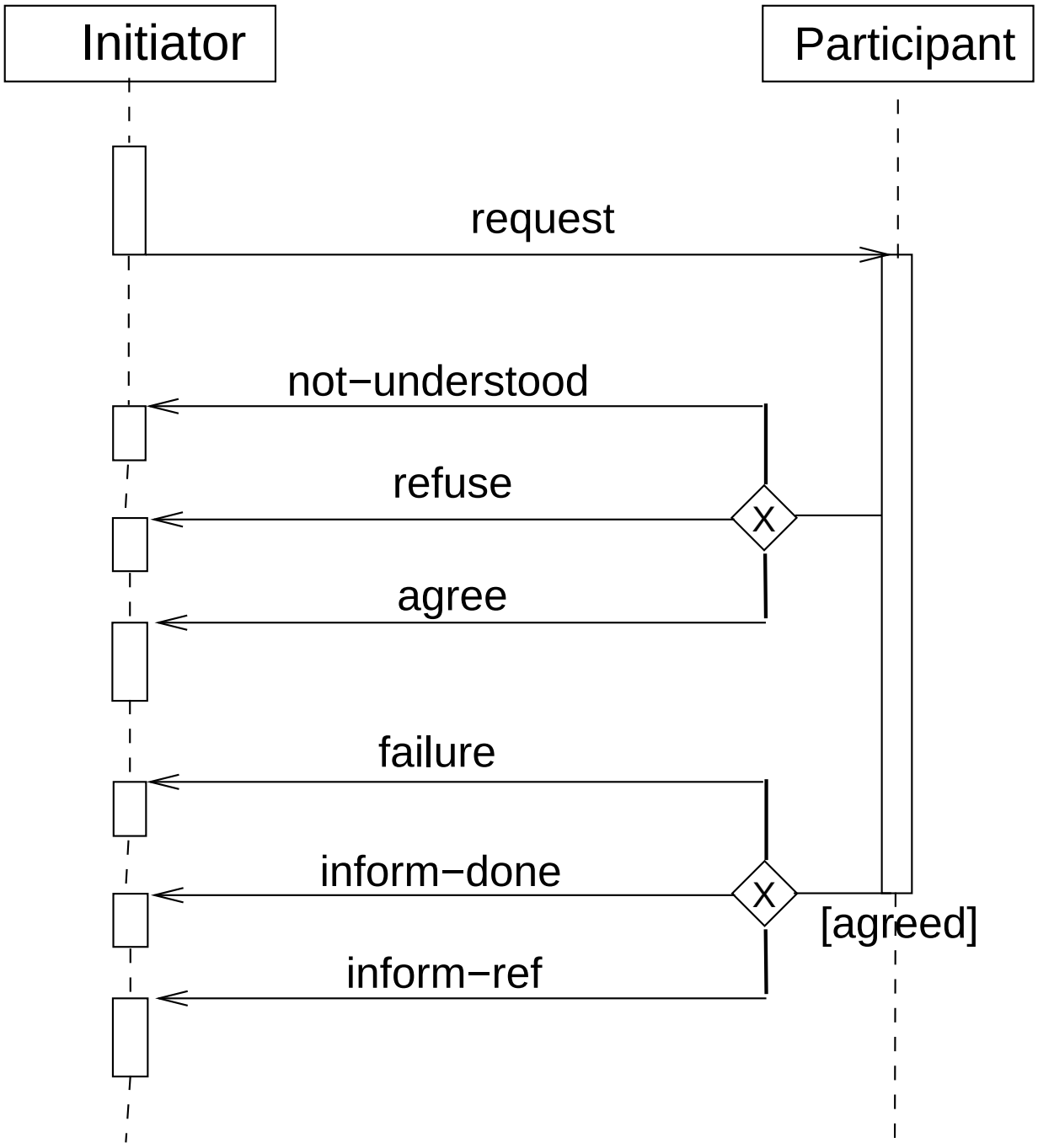
Meaning of a `request` message

Communicative Act:	request
Summary:	The sender requests the receiver to perform some action.
Meaning:	The content of this message is a description of the action to be performed. The action can be anything that the receiving agent is capable of doing. The receiver can reply with a *refuse* message or with a *agree* message. After which it has to perform the act and (optionally) inform the requester that the action has been completed.
Formal model:	$< i, \text{request}(j, a) >$
Preconditions:	$FP(a)$ $[i \backslash j]$ $\wedge$ B_i Agent(j, a) $\wedge$ $\neg B_i$ I_j Done(a)
Effect:	$Done(a)$

Meaning of a `refuse` message

Communicative Act:	refuse
Summary:	The sender refuses to perform the requested action.
Meaning (Sender):	The sender responds to an earlier *request* and informs the receiver that he refuses to perform the action. A reason for refusing can be sent along with the the *refuse*.
Meaning (Receiver):	The receiver can conclude that the sender does not commit himself to the action.
Formal model:	$< i, refuse(j, < i, act >, \varphi) >= < i, disconfirm(j, Feasible(< i, act >)); < i, inform(j, \varphi \wedge \neg Done(< i, act >) \wedge \neg I_i Done(< i, act >)) >$
Preconditions:	$B_i \neg Feasible(< i, act >) \wedge B_i(B_j Feasible(< i, act >)) > \vee U_j Feasible(< i, act >)) \wedge B_i \alpha \wedge \neg B_i(Bif_j \alpha \vee Uif_j \alpha)$
Effect:	$B_j \neg Feasible(< i, act >) \wedge B_j \alpha$ Where $\alpha = \varphi \wedge \neg Done(< i, act >) \wedge \neg I_i Done(< i, act >)$. Agent i informs j that action act is not feasible, and further that, because of proposition φ, act has not been done and i has no intention to do act.

Interaction Protocol



	Agent A	**Agent B**
GB BB MB	Send(inform(s(A), r(B), content(φ))) φ, $\neg(K_B(\varphi))$	
GB BB MB	 φ, $\neg(K_B(\varphi))$	 inform(s(A), r(B), content(φ))
GB BB MB	Remove($\neg(K_B(\varphi))$; Believe($K_B(\varphi)$) φ, $\neg(K_B(\varphi)$, $K_B(K_A(\varphi))$	Believe($K_A(\varphi)$)
GB BB MB	Believe($K_B(\varphi)$) φ, $K_B(K_A(\varphi))$	 $K_A(\varphi)$
GB BB MB	 φ, $K_B(K_A(\varphi))$, $K_B(\varphi)$	Believe(φ) $K_A(\varphi)$
GB BB MB	 φ, $K_B(K_A(\varphi))$ $K_B(\varphi)$	 $K_A(\varphi)$, φ

Extending 3APL's Semantics

The definition of a 3APL agent was:

$$< a, \Pi, \sigma, BAct, \Gamma >$$

1. Add a messagebuffer for (sending and) receiving of messages

$$< a, \Pi, \sigma, BAct, \Gamma, \Omega >$$

2. Define basic communication actions

3. Define Message Practical Reasoning Rules

$$\pi_h \stackrel{MB}{\longleftarrow} \varphi \mid \pi_b$$

4. Define semantics for communicative acts with

   - Message Practical Reasoning Rules
   - Transition Rules

Send, Receive and Synchronize

$$\frac{\varphi =< \iota, a, \beta, \rho, \gamma >}{< a, \Pi \cup \{Send(\varphi); \pi\}, \sigma, \theta, \Omega > \xrightarrow{\varphi!} < a, \Pi \cup \{\pi\}, \sigma, \theta, \Omega \cup \{s^{ent(\varphi)}\} >}$$

$$\frac{\psi =< \iota, \alpha, b, \rho, \gamma >}{< b, \Pi \cup \{Re^{c}(\psi)\}, \sigma, \theta, \Omega > \xrightarrow{\psi\tau?} < b, \Pi \cup \{Rec(\psi)\}, \sigma, \theta, \Omega \cup \{received(\psi\tau)\} >}$$

$$\frac{A \xrightarrow{\psi\tau?} A' \ , \ B \xrightarrow{\varphi!} B' \ , \ \psi\tau = \varphi}{M \cup \{A, B\} \rightarrow M \cup \{A', B'\}}$$

Semantics of `inform` with Transition-Rules

- The sending agent (before sending):

$$\frac{\varphi =< \iota, a, \beta, inform, \psi > \wedge\ \sigma \models \psi}{< a, \Pi \cup \{Send(\varphi); \pi\}, \sigma, \theta, \Omega > \xrightarrow{\varphi!} < a, \Pi \cup \{\pi\}, \sigma, \theta, \Omega \cup \{s^{ent(\varphi)}\} >}$$

- The sending agent (after sending):

$$\frac{}{< a, \Pi, \sigma, \Omega \cup \{sent(\iota, a, \beta, inf, \varphi)\} > \rightarrow < a, \Pi, \sigma + \{B_\beta B_a \varphi\}, \Omega >}$$

- The receiving agent:

$$\frac{}{< b, \Pi, \sigma, \Omega \cup \{r^{ec(\iota, \alpha, b, inf, \varphi)}\} > \rightarrow < b, \Pi, \sigma + \{B_\alpha \varphi\}, \Omega >}$$

Semantics of `inform` with PR-Rules

- The sending agent:

$$\overset{MB}{\longleftarrow} sent(\iota, a, \beta, inf, \varphi) \mid Update(B_\beta B_a \varphi)$$

- The receiving agent:

$$\overset{MB}{\longleftarrow} rec(\iota, \alpha, b, inf, \varphi) \mid Update(B_\alpha \varphi)$$

Semantics of `request, agree, refuse` with Transition-Rules

- The initiating agent:

$$\frac{}{< a, \Pi, \sigma, \Omega \cup \{s^{ent}(\iota, a, \beta, req, Actio^{n})\} > \rightarrow < a, \Pi, \sigma, \Omega >}$$

- The receiving agent:

$$\frac{\sigma \models pre(Actio^{ns})}{< b, \Pi, \sigma, \Omega \cup \{rec(\iota, \alpha, b, req, Actio^{ns})\} \rightarrow < b, \Pi \cup \pi_{agr}, \sigma, \Omega >}$$

  With:

  - $pre()$ is a function, which maps a sequence of actions to its preconditions.
  - π_{agr} is $\{Send(\iota', b, \alpha, agr, (Actio^{ns})\}$; *Actions*; $assess(done(\varphi), \alpha)\}$
  - $assess(done(\varphi), \alpha)$ sends either a *failure* or *inform-done* message.

- The receiving agent, self-deciding:

$$\overset{MB}{\longleftarrow} \quad rec(\iota, \alpha, \beta, req, \varphi) \mid \\ (know(\alpha)?; Update(agree(\iota, \alpha, \beta, req, \varphi))) \\ + \\ (\neg know(\alpha)?; Update(refuse(\iota, \alpha, \beta, req, \varphi)))$$

$$\frac{}{< b, \Pi, \sigma \cup \{agree(\iota, \alpha, b, req, \varphi)\}, \Omega > \rightarrow < b, \Pi \cup \pi_{agr}, \sigma, \Omega >}$$

With:

$$\pi_{agr} = \{ \quad Send(\iota', b, \alpha, agr, \varphi); \\ \varphi; \\ assess(done(\varphi), \alpha\}$$

Semantics of `request` with PR-Rules

- The receiver of a request:

$$
\begin{aligned}
handle\_request(f) \quad & \overleftarrow{MB}\, received(\iota, \alpha, \beta, request, Actio^n) \mid \\
& (\neg f(Actio^n)?; \\
& \quad Send(reply(\iota', \beta, \alpha, refuse, Actio^n)) \\
& + \\
& (\ f(Actio^n)?; \\
& \quad Send(\iota', \beta, \alpha, agree, Actio^n); \\
& \quad Actio^n; \\
& \quad (\neg post(Actio^n)?; \\
& \quad\ Send(\iota'', \beta, \alpha, failure, (done(Actio^n), \rho))) \\
& \quad + \\
& \quad (\ post(Actio^n)?; \\
& \quad\ Send(\iota'', \beta, \alpha, inform, done(Actio^n)))
\end{aligned}
$$

- Conclusions
  - Transition Rules not very suitable for defining semantics
  - FIPA ACL Semantics are not complete
- Future Work & Open Problems
  - Implementation
  - Defining semantics for more performatives in 3APL
  - Identity-problems