Most treatments of communication in (multi)agent systems borrow their inspiration from speech act theory.

Speech act theories are pragmatic theories of language, i.e., theories of language use: they attempt to account for how language is used by people every day to achieve their goals and intentions.

The origin of speech act theories are usually traced to Austin’s 1962 book, *How to do things with Words*.

Austin noticed that some utterances are like ‘physical action’ that appear to change the state of the world.

But more generally, everything we utter is uttered with the intention of satisfying some goal or intention.

A theory of how utterances are used to achieve intentions is speech act theory.

**Locutionary act** – the act of making an utterance. (e.g., saying “please make some tea”)

**Illocutionary act** – intention of utterance. (e.g., “he requested me to make some tea”).

**Perlocutionary act** – resulting action. (e.g., “He got me to make some tea”)
Five different types of speech act

- **representatives** - such as informing e.g. “it’s raining”
- **directives** - attempts to get the hearer to do something (commands, requests)
- **commisives** - which commit the speaker to doing something e.g. “I promise to...”
- **declaratives** - such as declaring war etc..
- **expressives** - whereby the speaker expresses a mental state (feelings and attitudes) e.g. “thank you!”

In general, a speech act can be seen to have two components:
- A **performative verb**: e.g. request, inform,..
  - Used to indicate intention
- A **propositional content**: e.g., “the door is closed”

Consider:

> `performative=request`  
> `content=“the door is closed”`  
> `speech act=“please close the door”`

**KQML (Knowledge Query Manipulation Language)**

- US-based DARPA funded Knowledge Sharing Effort (KSE) formed in early 90’s.
- It is an outer language for Agent Communication.
- Not concerned with the content of the message.
- Only representative and directive speech acts
- Emphasis on info. exchange rather than task allocation
Virtual Knowledge Base

<table>
<thead>
<tr>
<th>Parameters for KQML messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>:content</td>
</tr>
<tr>
<td>:in-reply-to</td>
</tr>
<tr>
<td>:language</td>
</tr>
<tr>
<td>:ontology</td>
</tr>
<tr>
<td>:receiver</td>
</tr>
<tr>
<td>:reply-with</td>
</tr>
<tr>
<td>:sender</td>
</tr>
<tr>
<td>:force</td>
</tr>
</tbody>
</table>

Structure of KQML Messages

```
(performative
  attr_name1  attr_value1
  attr_name2  attr_value2
  ......
  attrk_namek attr_valuek)
```

Attribute/Value pairs can be in any order.

Example KQML dialogue - ask/tell

```
(ask-one
  :sender  joe
  :content  "price(ibm, Price)"
  :receiver  stock_server
  :reply_with  ibm_stock_query
  :language  Prolog
  :ontology  NYSE-TICKS)
```

```
(tell
  :sender  stock_server
  :content  "price(ibm, 123)"
  :receiver  joe
  :in_reply_to  ibm_stock_query
  :language  Prolog
  :ontology  NYSE-TICKS)
```

Plus a list of attribute/ value pairs
Example KQML dialogue - evaluate/reply

(\textbf{evaluate}    \\
:sender engineerAg1    \\
:content (val (torque m1))    \\
:receiver engineeringKB    \\
:reply \_with torqueVal    \\
:language KIF    \\
:ontology Engineering1)

(\textbf{evaluate}    \\
:sender engineerKB    \\
:content (= (torque m1) (scalar 12 kgf))    \\
:receiver engineeringAg1    \\
:in \_reply \_to torqueVal    \\
:language KIF    \\
:ontology Engineering1)

\textbf{ask-all}: \textit{All answers in one reply}

(\textbf{ask-all}    \\
:sender joe    \\
:content "price(ibm,(Price,Time))"    \\
:receiver stock_server    \\
:reply \_with ibm\_stock\_query    \\
:language Prolog    \\
:ontology NYSE-TICKS)

(\textbf{tell}    \\
:sender stock_server    \\
:content [list of stock prices]    \\
:receiver joe    \\
:in \_reply \_to ibm\_stock\_query    \\
:language Prolog    \\
:ontology NYSE-TICKS)

\textbf{stream-all}: \textit{Getting a stream of answers}

(\textbf{stream-all}    \\
:sender Agent A    \\
:receiver Agent B)

\begin{table}[h]
\centering
\begin{tabular}{|l|l|}
\hline
\textbf{Category} & \textbf{Name} \\
\hline
Basic query & evaluate, ask-if, ask-one, ask-all, ask-about \\
\hline
Multi-response (query) & stream-about, stream-all, eos, subscribe \\
\hline
Response & reply, sorry \\
\hline
Generic Informational & tell, achieve, cancel, untell, unachieve \\
\hline
Generator & standby, ready, next, rest, discard, generator \\
\hline
Capability-definition & advertise, recommend, recruit, broker monitor, import, export \\
\hline
Networking & register, unregister, forward, broadcast, route \\
\hline
\end{tabular}
\end{table}
**standby:** getting answers one at a time, on demand

Message sequence:

```plaintext
=> (standby :language KQML :reply-with sq1
 :content
 (stream-all :language KIF
 :content ... :reply-with q1)
 :sender Agent A :receiver Agent B)
<= (ready reply-with sq1 in-reply-to sq1....)
=> (next :in-reply-to sq1...)
<= (tell :in-reply-to q1...)
=> (next .....) or (discard :in-reply-to sq1...)
<= (eos :in-reply-to q1) when no more answers
```

**subscribe:** asking to be told about new answers

```plaintext
(subscribe :sender S :receiver R ...
 :content (ask :content "price(IBM,P)"
 :reply-with s1 ...))
```

Now, if in the future R could give a different answer to
(ask :content "price(IBM,P)"

than could be provided now, an appropriate:

```plaintext
(tell :receiver R :sender S in-reply-to s1 ...
 :content "price(IBM,..))
```

is sent

---

**Mediation Services**

Mediation performatives

- **advertise** — agent says it can handle KQML messages of certain format
- **recommend-one, recommend-all** — requests to mediator for identities of agents that can handle a given message form
- **recruit-one, recruit-all** — forward the embedded KQML message to one (or all) agents who have advertised an ability to handle it. Replies will come from these agents.
- **broker-one, broker-all** — forward the embedded KQML message to one (or all) agents who have advertised an ability to handle it. Replies will come from the mediator/broker.
**advertise**: announce an ability to respond to some messages


S announces to DA that it can handle *ask* queries using a particular ontology and content syntax.


**Other performatives**

- **achieve**: S wants R to make something true in R’s env.
- **deny**: S claims it *cannot* handle embedded form of message (opp. of advertise)
- **forward**: S wants R to send embedded messages to another agent
- **error**: S considers R’s previous message to be ill formed
- **untell**: S claims content message is not in its KB

**Drawbacks of KQML**

- Initially no formal semantics
- Focus is on knowledge transfer
- No commissives - promises or commitments
- Only achieve action request
- Not suitable for distributed problem solving/ task sharing

**FIPA ACL**

- FIPA (Foundations of Intelligent Physical Agents) - develop standards for agent systems.
- FIPA ACL message structure similar to KQML.
  - Additional attributes:
    - :conversation-id, :protocol
- They have different performatives for info exchange and extra performatives for task exchange
**FIPA ACL: Performatives**

<table>
<thead>
<tr>
<th>Performative</th>
<th>Passing Information</th>
<th>Requesting Information</th>
<th>Negotiation</th>
<th>Performing Actions</th>
<th>Error Handling</th>
</tr>
</thead>
<tbody>
<tr>
<td>accept-proposal</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>agree</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cancel</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cfp</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>confirm</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>disconfirm</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>failure</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>inform</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inform-if</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inform-ref</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>not-understood</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

**FIPA Performatives ....cont.**

<table>
<thead>
<tr>
<th>Performative</th>
<th>Passing Information</th>
<th>Requesting Information</th>
<th>Negotiation</th>
<th>Performing Actions</th>
<th>Error Handling</th>
</tr>
</thead>
<tbody>
<tr>
<td>propagate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>propose</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>proxy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>query-if</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>query-ref</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>refuse</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>reject-proposal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>request</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>request-when</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>request-whenever</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>subscribe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

**Sample brief descriptions-1**

- **cfp**: Call for proposals. Content is an action and a condition - e.g. sell sender a car, particular make and price
- **propose**: Response to a cfp
- **reject-proposal**: Response to a propose
- **accept-proposal**: Response to a propose
- **request**: Of an action - performative
- **refuse**: Will not perform req. action
- **agree**: Will perform requested action

**Sample brief descriptions-2**

- **query-if**: Is some statement believed
- **query-ref**: Request for value for an expression (cf evaluate of KQML) - content could be a set expression to achieve effect of a KQML ask-all
- **inform**: Like KQML tell used in response to queries
- **request-when**: Do action when a cond. is believed
- **request-whenever**: Do action whenever a cond. is believed
Knowledge Interchange Format (KIF)

- Geneserth and Fikes (1992)
  - Developed for content of KQML messages
  - Essentially a version of the first order predicate calculus in Lisp notation e.g.
  
  \[(\text{employee} \ 015468 \ 1000 \ \text{warehouse})\]

  \[(\text{and} \ (> \ \text{width} \ \text{chip1}) \ (> \ \text{length} \ \text{chip1}) \ (> \ \text{processor} \ \text{chip1}))\]

- It is possible to express:
  - properties of things in a domain
  - relationships between things in a domain
  - general properties of a domain

More KIF examples

```lisp
(forall (?P person)
  (<= (likes ?P cabbage)
       (likes ?P sprouts)))
```

```lisp
(not (exists ?C (child ?C) (taller_than ?C (metres 2.5))))
```

```lisp
(defrelation person (?P) := (and (animal ?P) (human)))
```

```lisp
(deffunction father_of (?P person) :=
  (cond ( (and (parent ?P ?F) (male ?F)) ?F)
        (true undef))
```

FIPA SL (Foundation for Intelligent Physical Agents)

- Main content language for FIPA ACL
- LISP style predicate logic like KIF but augmented with:
  - Modal operators
  - Description operators
  - Descriptive terms

SL modal operators

- (B A S) - A believes S
- (U A S) - A is uncertain about S
- (I A S) - A intends S becomes true
- (PG A S) - A has S as persistent goal
- (feasible Act S) - Act is feasible when S holds
- (done Act) - Act has been achieved
**SL description operators**

(iota T F)  - the T such that F
(any T F)   - any T such that F
(all T F)   - all T such that F

**SL descriptive terms**

Individual can be denoted by a functional expression, as in KIF, eg
(\text{height-of paul})

Or by an ‘object’ attribute expression, eg
(\text{vehicle :colour red :maxspeed 100 :owner (person :name paul)})

**Example messages using SL**

(query-ref

... :content (all ?f (child-of ?f paul)))

(inform

..... :content (= (all ?f (child-of ?f paul)) [bill,mary])))

**Ontologies for Agent Communication**

- An *ontology* is a common vocabulary and agreed upon meanings to describe a subject domain.

- A shared representation is essential for successful communication and coordination.
Standards and tools for Ontologies

- **DAML** ([http://www.daml.org](http://www.daml.org)): DARPA Agent markup language, DAML+OIL is a standard for defining ontologies.
- **ONTOLINGUA** ([http://ontolingua.stanford.edu](http://ontolingua.stanford.edu)) tools for writing, publishing and sharing ontologies.