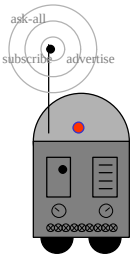
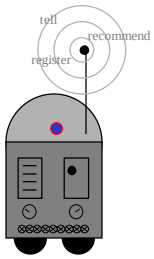
# Agent Communication Languages

**Tim Finin and Yannis Labrou**
**University of Maryland Baltimore County**

**ASA/MA**

**Oct. 3, 1999**




1

## Tutorial Overview

- Introduction to agent communication and ACLs
- KQML
- Content languages
- Ontologies

break

- The FIPA ACL
- Semantics of ACLs
- Systems and applications
- Challenges for the future
- Conclusions

2

# On the agent paradigm

- No consensus on what's an agent, but several key concepts are important to this emerging paradigm. A software agent:
  - is an autonomous, goal-directed process
  - is situated in, is aware of, and reacts to its environment
  - cooperates with other agents (software or human) to accomplish tasks
- Software agents offer a new paradigm for very large scale distributed heterogeneous applications.
- The paradigm focuses on the **interactions** of autonomous, cooperating processes which can adapt to human & other agents.
- Mobility is an orthogonal characteristic which many, but not all, consider important.
- Intelligence is always a desirable characteristic but is not strictly required by the paradigm.
- The paradigm is still forming.

3

# Why is communication important?

- Most, but not all, would agree that communication is a requirement for cooperation.
- Societies can do things that no individual (agent) can.
- Diversity introduces heterogeneity.
- Autonomy encourages disregard for other agents' internal structure.
- Communicating agents need only care about understanding a "common language".

4

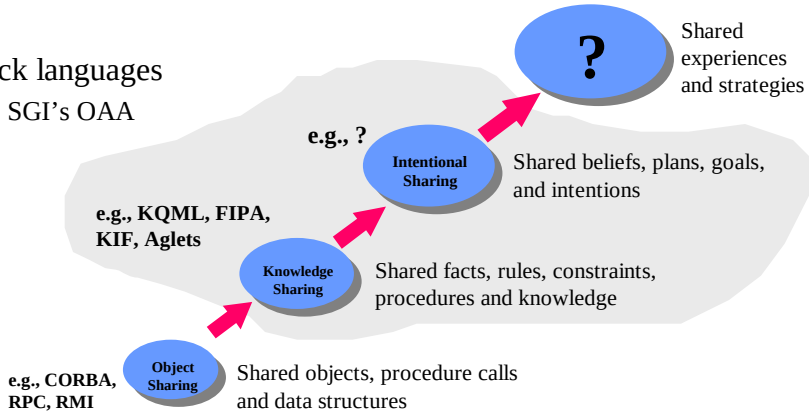
# Agent Communication

- Agent-to-agent communication is key to realizing the potential of the agent paradigm, just as the development of human language was key to the development of human intelligence and societies.
- Agents use an ***Agent Communication Language*** or ACL to communication information and knowledge.
- Genesereth (CACM, 1992) defined a software agent as any system which uses an ACL to exchange information.

5

# Some ACLs

- Is CORBA an ACL?
- Knowledge sharing approach
  - KQML, KIF, Ontologies
- FIPA
- Ad hock languages
  - e.g., SGI's OAA




6

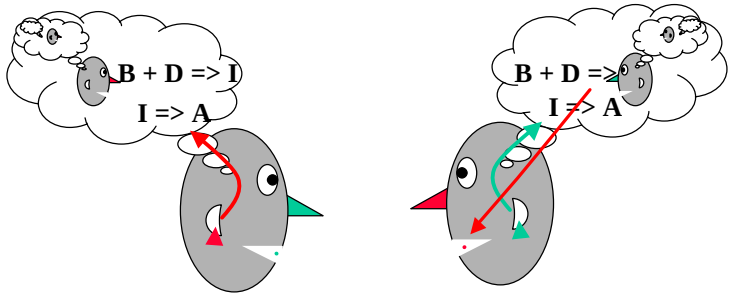
## The intentional level, BDI theories, speech acts and ACLs: How do they all fit together?

- ACL have message types that are usually modeled after speech acts
- Speech acts may be understood in terms of an intentional-level description of an agent
- An intentional description refers to beliefs, desires, intentions and other modalities
- BDI frameworks have the power to describe an agents' behavior, including communicative behavior

- Agents have "propositional attitudes" which are three part relationship between
  - an agent,
  - a content-bearing proposition (e.g., "*it is raining*"), and
  - a finite set of propositional attitudes (e.g., believing, asserting, fearing, wondering, hoping, etc.)
- <*a*, fear, raining($t_{now}$) >

7

## BDI Model and Communication




- Communication is a means to (1) reveal to others what our BDI state is and (2) attempt to effect the BDI state of others.
- Note the recursion: an agent has beliefs about the world, beliefs about other agents, beliefs about the beliefs of other agents, beliefs about the beliefs another agent has about it, ...

8

## Criticism of BDI theories

- The necessity of having all three modalities is questioned from both ends:
  - too few
  - too many
- System builders question their relevance in practice:
  - multi-modal BDI logics do not have complete axiomatizations
  - they are not efficiently computable

⇨ There is a gap between theory and practice

9

## Speech Act Theory

**A high level framework to account for human communication.** *Language as Action* (Austin)

- Speakers do not just utter true or false sentences.
- Speakers perform speech acts: assertions, commands, requests, suggestions, promises, threats, etc.
- Every utterance is some speech act
- Identifying the intended speech act is critical

Example: *"Shut the door!"*

- **locution** -- physical utterance with context and reference, *i.e.*, who is the speaker and the hearer, which door etc.
- **illocution** -- the act of conveying intentions, *i.e.*, speaker wants the hearer to close the door
- **perlocutions** -- actions that occur as a result of the illocution, *i.e.*, hearer closes the door

10

# Agent Communication Components

11

## Historical Note: Knowledge Sharing Effort

- Initiated by DARPA circa 1990
- Sponsored by DARPA, NSF, AFOSR, etc.
- Participation by dozens of researchers in academia and industry.
- Developing techniques, methodologies and software tools for *knowledge sharing* and *knowledge reuse*.
- Sharing and reuse can occur at *design*, *implementation* or *execution* time.

12

# Knowledge Sharing Effort

- Knowledge sharing requires a communication which requires a common language
- We can divide a language into syntax, semantics, and pragmatics
- Some existing components that can be used independently or together:
  - **KIF** - knowledge interchange format *(syntax)*
  - **Ontolingua** - a language for defining sharable ontologies *(semantics)*
  - **KQML** - a high-level interaction language *(pragmatics)*

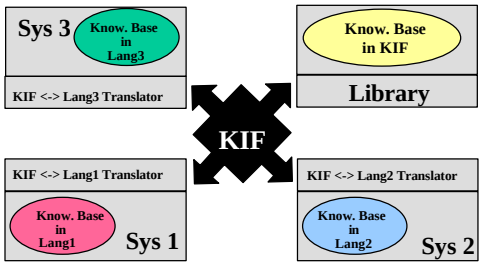
Propositional

Propositional attitudes

13

# Knowledge Interchange Format

- KIF ~ First order logic set theory
- An **interlingua** for encoded declarative knowledge
  - Takes translation among n systems from $O(n^2)$ to $O(n)$




- Common language for reusable knowledge
  - Implementation independent semantics
  - Highly expressive - can represent knowledge in typical application KBs.
  - Translatable - into and out of typical application languages
  - Human readable - good for publishing reference models and ontologies.
- Current specification at http://logic.stanford.edu/

14

## Common Semantics
## Shared Ontologies and Ontolingua

- ***Ontology***: A common vocabulary and agreed upon meanings to describe a subject domain.
- Ontolingua is a language for building, publishing, and sharing ontologies.
  - A web-based interface to a browser/editor server.
  - Ontologies can be automatically translated into other content languages, including KIF, LOOM, Prolog, etc.
  - The language includes primitives for combining ontologies.

15

## Common Pragmatics
### Knowledge Query and Manipulation Language

- KQML is a high-level, message-oriented, communication language and protocol for information exchange independent of content syntax and ontology.
- KQML is also independent of
  - transport mechanism, e.g., tcp/ip, email, corba, IIOP, ...
  - High level protocols, e.g., Contract Net, Auctions, …
- Each KQML message represents a single *speech act* (e.g., ask, tell, achieve, …) with an associated semantics and protocol.
- KQML includes primitive message types of particular interest to building interesting agent architectures (e.g., for mediators, sharing intentions, etc.)

16

## For completeness...

There are some other aspects to effective agent communication that we must address

- Common transport protocols (e.g., tcp/ip, udp, smtp, http, …)
- Common high-level protocols (e.g., contract-net, auctions, name registration, ...)
- Common service infrastructure (e.g., ans, broker, facilitator, logger …)

Our ACLs have to fit these in somewhere

17

# The KQML Agent Communication Language

18

# KQML

## Knowledge Query and Manipulation Language

- KQML is a high-level, message-oriented, communication language and protocol for information exchange independent of content syntax and ontology.
- KQML is independent of
  - the transport mechanism (e.g., tcp/ip, email, corba objects, IIOP, etc.)
  - Independent of content language (e.g., KIF, SQL, STEP, Prolog, etc.)
  - Independent of the ontology assumed by the content.
- KQML includes primitive message types of particular interest to building interesting agent architectures (e.g., for mediators, sharing intentions, etc.)

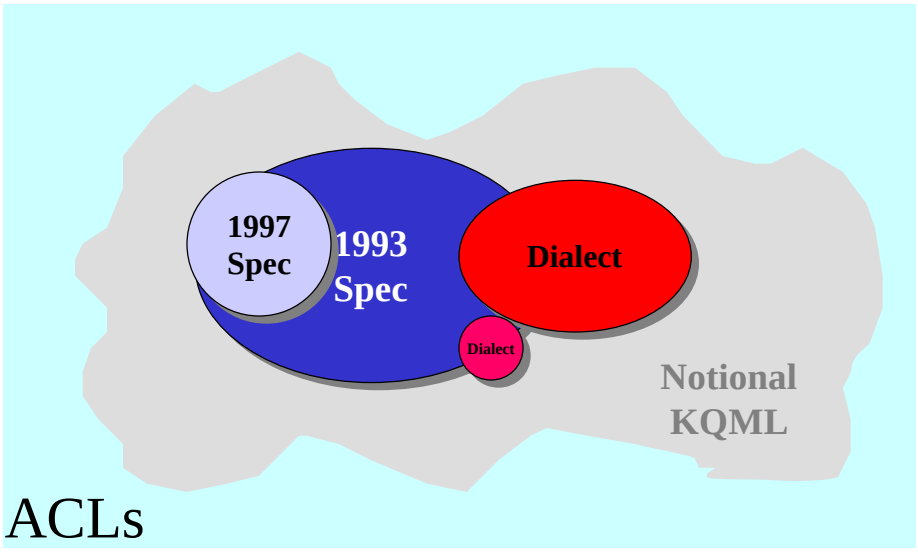
19

# KQML Specifications

- There are two KQML specification documents:
  - *Specification of the KQML Agent-Communication Language plus example agent policies and architectures*, The DARPA Knowledge Sharing Initiative, External Interfaces Working Group, 1993. http://www.cs.umbc.edu/papers/kqml93.pdf
  - *A Proposal for a new KQML Specification*, Yannis Labrou and Tim Finin, TR CS-97-03, Feb.1997, Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, Baltimore, MD 21250. http://www.cs.umbc.edu/kqml/papers/kqml97.pdf
- There are also many dialects and “extended” versions of KQML plus lots of important concepts not addressed in either specification document (e.g., security).

20

# Multiple KQML dialects




21

# A KQML Message

performative
parameter
value

```
(tell   :sender       bhkAgent
        :receiver     fininBot
        :in-reply-to  id7.24.97.45391
        :ontology     ecbk12
        :language     Prolog
        :content      "price(ISBN3429459,24.95)")
```

Represents a single *speech act* or *performative*

ask, tell, reply, subscribe, achieve, monitor, ...

with an associated *semantics and protocol*

$tell(i,j,B_i\phi) = fp[B_i B_i\phi \wedge \neg B_i(Bif_j B_i\phi \vee Uif_j B_i\phi)] \wedge re[B_j B_i\phi] \ldots$

and a list of *attribute/value pairs*

:content, :language, :from, :in-reply-to

22

# KQML Syntax

- KQML was originally defined as a language with a particular linear syntax which is based on Lisp.
- Alternate syntaxes have been used, e.g., based on SMTP, MIME, HTTP, etc.)
  - There are proposals for a meta-syntax that can support different syntactic dialects.
- KQML has also been mapped onto objects and passed from agent to agent as objects (e.g., if in the same memory space) or serialized objects.
- **KQML is not about syntax.**

23

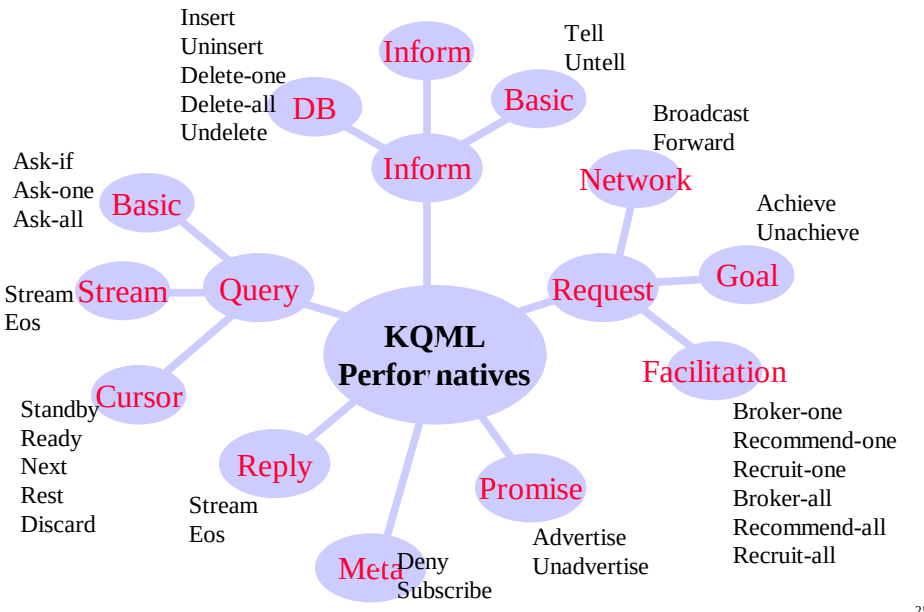
# KQML Reserved Parameter Keywords

**1997**

| | |
|---|---|
| **:sender** | the actual sender of the performative |
| **:receiver** | the actual receiver of the performative |
| **:from** | the origin of the performative in *:content* when forward is used |
| **:to** | the final destination of the performative in *:content* when forward is used |
| **:in-reply-to** | the expected label in a response to a previous message (same as the *:reply-with* value of the previous message) |
| **:reply-with** | the expected label in a response to the current message |
| **:language** | the name of the representation language of the :content |
| **:ontology** | the name of the ontology (e.g., set of term definitions) assumed in the :content parameter |
| **:content** | the information about which the performative expresses an attitude |

24

# Performatives (1997)




25

# The Virtual Knowledge Base

- Both specifications describe the performatives using the notion of a *virtual knowledge base* (VKB)
- It is virtual in two senses
  - The KQML- speaking agent may not have an explicit knowledge-base at all.
  - The "facts" in the VKB cam be intentionally or extensionally defined.
- Modeling agents as having a KB is a useful abstraction.

26

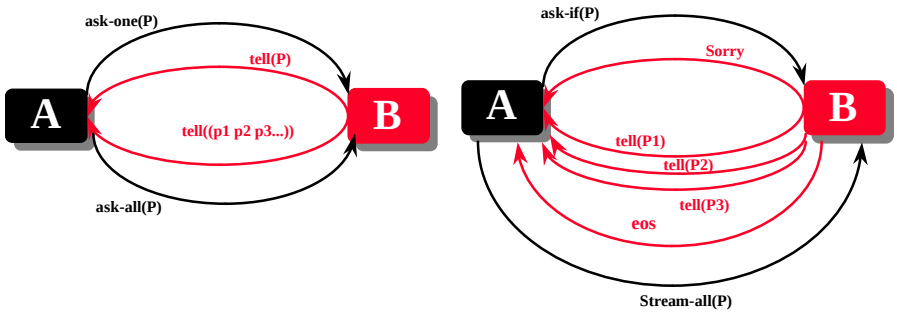
# Facilitation Services

Facilitators are a class of agents who

- traffic in meta-knowledge about other agents.
- provide communication services such as:
  - message forwarding and broadcasting
  - resource discovery
  - matchmaking
  - content-based routing
  - meta-knowledge queries
- Performatives of special interest to facilitators are
  - advertise, broker, recruit, recommend, forward, broadcast, etc.
- Brokers are generally considered to focus on matchmaking
- Facilitators can be intelligent or not
  - Intelligent facilitators use domain knowledge in matching services needs and offers.

27

# Simple Query Performatives



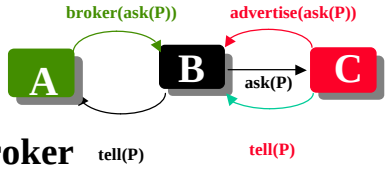

- The **ask-one**, **ask-all**, **ask-if**, and **stream-all** performatives provide a basic query mechanism.
- The 1993 spec also included **ask-about** which most
- systems do not implement

28

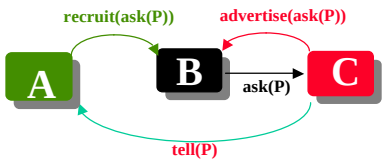
# Facilitation Performatives

The three facilitation performatives come in a X-one and X-all versions:

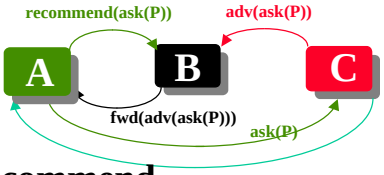
- Broker-one and broker-all
- Recruit-one and recruit-all
- recommend-one and recommend-all




**Broker**




**Recruit**




**Recommend** tell(P)

29

# What's Missing from KQML?

- Your favorite performative
  - e.g., OFFER, ACCEPT, …
- Useful new parameters
  - e.g., :PROTOCOL, :VERSION, :REPLY-BY, :SIGNATURE, …
- Conventions for "conversational context"
  - e.g., a context-dependent way of determining default values for parameters

30

## Extensibility

- We can extend KQML by either:
  - Adding new performatives
  - adding new parameters
  - Creating new ontologies
- There are no language impediments to any of these types of extension
- A mechanism for creating machine-readable performative definitions is desirable
  - Having a "compositional semantics" would be helpful here
- Some features have to be added in a more systematic way (e.g., security)

31

# Content Languages for Agent Communication

32

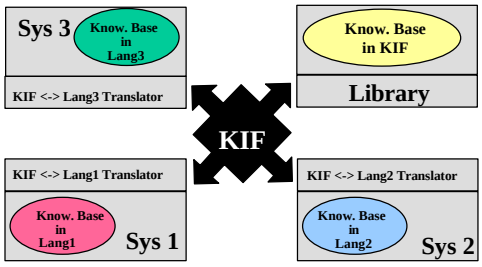
# Representation and Reasoning

- Intelligent agents need to be able to represent and reason about many things, including:
  - models of other agents (human or artificial) beliefs, desires, intentions, perceptions, plans, etc.
  - task, task structures, plans, etc.
  - meta-data about documents and collections of documents
- In general, they will need to **communicate** the same range of knowledge.
- A variety of content languages have been used with ACLs, including KIF, SL, Loom, Prolog, CLIPS, SQL, …
- There is a special interest in content languages that can serve as a neutral, but expressive, **interlingua** for a wide range of systems.
- We'll look at KIF in a bit more detail.

33

# Knowledge Interchange Format

- KIF ~ First order logic set theory
- An **interlingua** for encoded declarative knowledge
  - Takes translation among n systems from $O(n^2)$ to $O(n)$




- Common language for reusable knowledge
  - Implementation independent semantics
  - Highly expressive - can represent knowledge in typical application KBs.
  - Translatable - into and out of typical application languages
  - Human readable - good for publishing reference models and ontologies.
- Current specification at http://logic.stanford.edu/

34

## KIF Syntax and Semantics

- Extended version of first order predicate logic
- Simple list-based linear ASCII syntax, e.g.,

  (forall ?x (=> (P ?x) (Q ?x)))

  (exisits ?person (mother mary ?person))

  (=> (apple ?x) (red ?x))

  (<<= (father ?x ?y) (and (child ?x ?y) (male ?x))
- Model-theoretic semantics
- KIF includes an axiomatic specification of large function and relation vocabulary and a vocabulary for numbers, sets, and lists

35

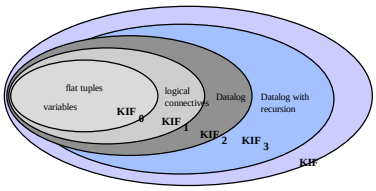
## Big KIF and Little KIF

- That KIF is highly expressive language is a desirable feature; but there are disadvantages.
  - complicates job of building fully conforming systems.
  - resulting systems tend to be "heavyweight"
- KIF has "*conformance categories*" representing dimensions of conformance and specifying alternatives within that dimension.
- A "*conformance profile*" is a selection of alternatives from each conformance category.
- System builders decide upon and adhere to a conformance profile sensible for their applications.

36

## Conformance Categories and Profiles

- Conformance Categories
  - **logical form:** {atomic, conjunctive, positive, logical, rule-based, quantified}
  - **recursion:** yes/no
  - **terms:** {constants, variables, complex terms}
  - **relational variables:** yes/no
- Common Conformance Profiles might be
  - Databases (ground atomic assertions & conjunctive forms)
  - Datalog
  - Relational logic
  - First order logic
  - Second order logic




37

## KIF Software

- Several KIF-based reasoners in LISP are available from Stanford (e.g., EPILOG).
- IBM's ABE (Agent Building Environment) & RAISE reasoning engine use KIF as their external language.
- Stanford's Ontolingua uses KIF as its internal language.
- Translators (partial) exist for a number of other KR languages, including LOOM, Classic, CLIPS, Prolog,...
- Parsers for KIF exist which take KIF strings into C++ or Java objects.

38

## Other alternatives

- OKBC (see ontologies)
- Java objects (see AgentBuilder)
- SL (see FIPA)
- Constraints
- Database tuples
- RDF
- *..your favorite representation language here..*

39

## Content Languages Summary

- KIF is the only widely used interlingua for KB systems
  - KIF is the focus of an ANSI standardization effort
  - See KIF spec at <http://logic.stanford.edu/> and also <http://www.cs.umbc.edu/kif> for more information.
- Its future outside the AI-related community is unclear
  - It may not be acceptable to a wider community because its too logic-oriented or not object-oriented or …
  - Then again, it's expressive power may win the day!
- Defining an mapping of KIF to XML might make it more acceptable.

40

# Ontologies

41

## Overview

- What is an ontology?
- Tools for building, using and maintaining ontologies
- Existing ontologies of general interest
- FIPA's view on agents and ontologies

42

## Common Semantics
## Shared Ontologies and Ontolingua

*Ontology* : A common vocabulary and agreed upon meanings to describe a subject domain.

> On*tol"o*gy (?), n. [Gr. the things which exist (pl.neut. of , , being, p.pr. of to be) + -logy: cf.F. ontologie.]
>
> That department of the science of metaphysics which investigates and explains the nature and essential properties and relations of all beings, as such, or the principles and causes of being.
>
> *Webster's Revised Unabridged Dictionary (G & C. Merriam Co., 1913, edited by Noah Porter)*

This is not a profoundly new idea …
- Vocabulary specification
- Domain theory
- Conceptual schema (for a data base)
- Class-subclass taxonomy
- Object schema

43

## Conceptual Schemas

A conceptual schema specifies the intended meaning of concepts used in a data base

**Data Base:**

| | |
|---|---|
| 139 | 74.50 |
| 140 | 77.60 |
| … | … |

**Data Base Schema:**

**Table: price**
***stockNo: integer; cost: float**

**Conceptual Schema:**

**price(x, y) =>**
**$ (x', y') [auto_part(x')**
**& part_no(x') = x**
**& retail_price(x', y', Value-Inc)**
**& magnitude(y', US_dollars) = y]**

Auto Product Ontology

Product Ontology

Units & Measures Ontology

44

## Implicit vs. Explicit Ontologies

- Systems which communicate and work together must share an ontology.
- The shared ontology can be **implicit** or **explicit**.
- Implicit ontology are typically represented only by procedures
- Explicit ontologies are (ideally) given a declarative representation in a well defined knowledge representation language.

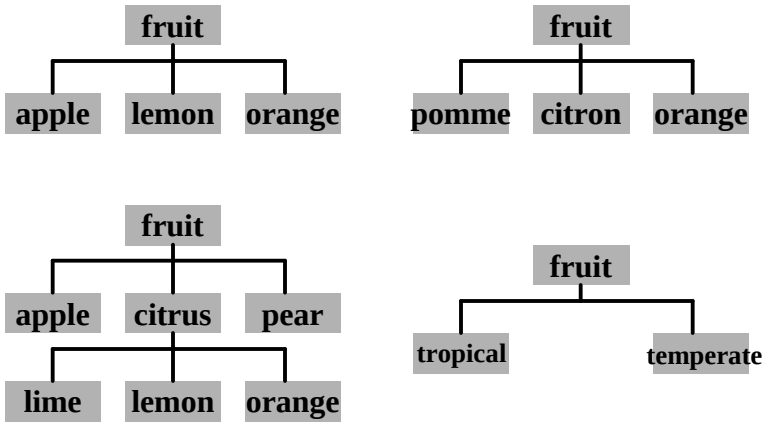
45

## Conceptualizations, Vocabularies and Axiomitization

- Three important aspects to explicit ontologies
  - **Conceptualization** involves the underlying model of the domain in terms of objects, attributes and relations.
  - **Vocabulary** involves assigning symbols or terms to refer to those objects, attributes and relations.
  - **Axiomitization** involves encoding rules and constraints which capture significant aspects of the domain model.
- Two ontologies may
  - be based on different conceptualizations
  - be based on the same conceptualization but use different vocabularies
  - differ in how much they attempt to axiomitize the ontologies

46

## Simple examples




47

## KR Language := Logic + Ontology

Knowledge representation language provides:

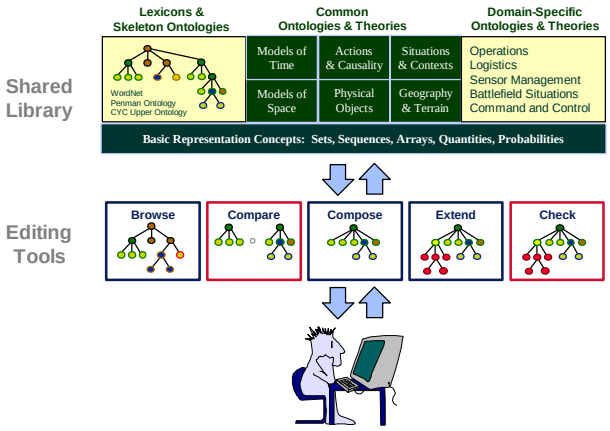
- **A logical formalism**
  - Syntax for well formed formulae (wffs)
  - Vocabulary of logical symbols (e.g., and, or, not)
  - Interpretation semantics for the logical symbols
- **An ontology**
  - Vocabulary of non-logical symbols
  - Definitions of symbols
  - Axioms constraining interpretation of primitive symbols

48

# Ontology Library and Editing Tools

**Ontolingua** is a language for building, publishing, and sharing ontologies.

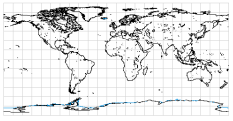
- A web-based interface to a browser/editor server at *http://ontolingua.stanford.edu/* and mirror sites.
- Ontologies can be translated into a number of content languages, including KIF, LOOM, Prolog, CLIPS, etc.




49

# Big Ontologies

- There are several large, general ontologies that are freely available, for example:
  - **Cyc -** Original general purpose ontology
  - **WordNet** - a large, on-line lexical reference system
  - **World Fact Book --** 5Meg of KIF sentences!
  - **UMLS** - NLM's Unified Medical Language System
- See http://www.cs.utexas.edu/users/mfkb/related.html



50

## Ontology Conclusions

- Shared ontologies are essential for agent communication and knowledge sharing
- Ontology tools and standards are important
  - Ontolingua and OKBC are good examples
  - XML and RDF may be a next step
- Some large general ontologies are available
  - Cyc, WFB, WordNet, …
- For more information…
  - http://www.kr.org/top describes projects addressing major ontology construction issues
  - Ontology mailing list: send mail to majordomo@cs.umbc.edu with "*info ontology*" in message body for information.
  - ANSI Ad Hoc Group on Ontology Standards: http://WWW-KSL.Stanford.EDU/onto-std/

51

# FIPA

52

# What is FIPA


- The Foundation for Intelligent Physical Agents (FIPA) is a non-profit association.
- FIPA's purpose is to promote the success of emerging agent-based applications, services and equipment.
- FIPA's goal is pursued by making available in a timely manner, internationally agreed specifications that maximise interoperability across agent-based applications, services and equipment.
- http://drogo.cselt.stet.it/fipa

53

# Who is FIPA

- FIPA operates through the open international collaboration of member organisations, which are companies and universities active in the agent field.
- Companies: Alcatel, British Telecom, Deutsche Telekom, France Telecom, Hitatchi, Hewlett Packard, IBM, Intel, Lucent, NEC, NHK, NTT, Nortel, Siemens, Telia, etc.
- Universities and Research Institutes: GMD, EPFL, Imperial, IRST, etc.

54

## FIPA's Work Model

- FIPA's work is built around annual rounds of FIPA specification deliverables.
- Current specification is FIPA97, which can be found at http://drogo.cselt.stet.it/fipa
- FIPA97 has 7 Technical Committees (TCs)
  - TC1: Agent Management
  - TC2: Agent Communication Language
  - TC3: Agent/Software Interaction
  - TC4-TC7: Specification of Applications

55

## TC2: Agent Communication Language

- Called FIPA ACL
- Based on speech acts
- Messages are actions (communicative actions or CAs)
- Communicative acts are described in both a narrative form and a formal semantics based on modal logic
- Syntax is similar to KQML
- Specification provides a *normative* description of high-level interaction protocols (aka conversations)

56

## Major Features of FIPA ACL

- Management and facilitation primitives (register, broker, recruit, etc.) are not part of the ACL
- Primitives can be defined compositionally from "core" primitives
- Use of a powerful language to define agents' states (Semantic Language, or SL)
- Semantics based on mental attitudes (belief, intention, etc.)
- The meaning of primitives is given in terms of Feasibility Preconditions (FPs) and Rational Effect (RE)

57

## Comparison of KQML tell and FIPA ACL inform

- The difference is only observable in the semantics
- Syntactically the two messages are almost identical
- Both languages make the same basic assumption of non-commitment to a content language (in this performative)
- Semantically they differ at two levels:
  - different ways to describe the primitive, i.e., pre-, post-, completion conditions for KQML, FPs and REs for FIPA ACL
  - different language to describe the propositional (mental) attitudes, e.g., KQML's bel is not the same as FIPA ACL B operator

58

## How do KQML and FIPA ACL differ?

- Different semantics; mapping of KQML performatives to FIPA primitives and vice versa is a futile exercise.
- Different treatment of the "administration primitives"; in FIPA ACL register, unregister, etc., are treated as requests for action with reserved (natural language) meaning
- No "facilitation primitives", e.g., broker, recommend, recruit, etc., in FIPA ACL
- Reserved content language: a very murky issue ...

59

## Which ACL should I use?

- The "sad truth" is that programmers do not care about semantics and their details.
- As long as the agent does not *implement* modalities (belief, intention, etc.) the semantic differences are irrelevant to the developer.
- The similar syntax guarantees that a developer will not have to alter the code that receives, parses and sends messages.
- The code that processes the primitives should change depending on whether the code observes the proper semantics.

60

## Really ... which one is better?

- FIPA ACL is more powerful with composing new primitives.
- The power stems from the power of the SL language as a content language to describe agents' states.
- KQML's weakness is its religious non-commitment to a content language.
- Both have shortcomings; there are features that developers would like to see in an ACL.

61

# ACL Semantics

62

# Cohen & Levesque

63

## The Cohen & Levesque Approach

- Most attempts for semantics for ACL descend from the work of Cohen & Levesque (C&L)
- Intention = Choice + Commitment
- Integration of Agent Theory and Semantics of Communication Primitives
- A (partial) theory of rational agency
- Possible-worlds semantics

64

## Semantics for INFORM

- {INFORM speaker listener e p} =

  {ATTEMPT speaker listener e
    (know listener p)
    [BMB listener speaker
      (P-GOAL speaker (KNOW listener (KNOW speaker P)))]}

Not present in ATTEMPT def'n

The "honest effort"

- An INFORM is defined as an attempt in which to make an "honest effort", the speaker is committed to making public that he is committed to the listener's knowing that he (the speaker) knows p.

65

# KQML Semantics

66

## Semantics for TELL

TELL(A,B,X)

- A states to B that A believes X to be true (for A).
- bel(A,X)
- Pre(A): bel(A,X) ∧ know(A,want(B,know(B,S)))
  where S may be bel(B,X) or NOT(bel(B,X))
  Pre(B): intend(B,know(B,S))
- Post(A): know(A,know(B,bel(A,X)))
  Post(B): know(B,bel(A,X))
- Completion: know(B,bel(A,X))
- The completion condition and postconditions hold unless a SORRY or ERROR suggests B's inability to properly acknowledge the TELL.

67

## Semantics for the proactive-TELL

proactive-TELL(A,B,X)

- A states to B that A believes the content to be true.
- bel(A,X)
- Pre(A): bel(A,X)
  Pre(B): NONE
- Post(A): know(A,know(B,bel(A,X)))
  Post(B): know(B,bel(A,X))
- Completion: know(B,bel(A,X))
- The postconditions and completion condition hold unless a SORRY or ERROR suggests B's inability to properly acknowledge the TELL.

68

# FIPA ACL Semantics

69

## An example of FIPA ACL semantics (inform)

<*i*, inform( *j*, ϕ )>

FP: $B_i\phi \wedge \neg B_i( Bif_j\phi \vee Uif_j\phi)$

RE: $B_j\phi$

**Agent i informs agent j that (it is true that) it is raining today:**

```
(inform
    :sender i
    :receiver j
    :content "weather(today,raining)"
    :language Prolog
    :ontology weather42)
```

70

## Shortcomings of Current ACLs

- Intentional level description: which mental attitudes, what definitions?
- Problems with mental attitudes: from theory to practice
- Can all desirable communication primitives be modeled after speech acts? Should they?
- Flexible description of agents' capabilities and advertising of such capabilities.
- How can we test an agent's compliance with the ACL?
- Ease of extending an ACL

71

# Challenges

72

# XML and ACLs

73

## Using XML to describe ACL messages

- Both KQML and FIPA ACL are using a LISP-like syntax to describe properly-formed ACL messages
- ACL messages have "deep" semantics (KR-like) than account for the Communicative Act, the Sender and the Receiver
- The deep semantics, in the case of FIPA ACL are described in SL
- A ACL message as a syntactic object has parameters that are not accounted for in the semantics (language, ontology, in-reply-to, etc.)

74

## Using XML to describe ACL messages (continued)

- Syntactically, ACL messages introduce pragmatic elements and a particular syntax useful for parsing and routing.
- The syntactic form (e.g., LISP-like) need not be unique.
- Syntactically, ACL messages can be thought as having an "abstract syntax".
- The abstract syntax "allows" for multiple **syntactic representations** or **encodings**
- Examples of encodings are: Lisp-like balanced parenthesis list, XML or even a Java structure

75

## Comments on the XML-encoding of ACL messages

- The content itself of the ACL message could have been encoded in XML
- The "deep semantics" of the ACL message are taken to be the same as before ("canonical" syntactic encoding)
- The XML-encoding enhances the canonical syntactic encoding:
  - it contains parsing information
  - parameter values are not strings but links
- The XML-encoding is not equivalent to the canonical syntactic encoding

76

## Advantages of XML-encoding ACL messages

- Parsing ACL messages is a big overhead of agent development.
- The XML encoding is easier to develop parsers for:
  - One can use off-the-shelf XML parsers
  - a modified DTD does not mean re-writing the parser
- ACL messages are more WWW-friendly
  - easier integration with web-based technologies
  - potential for taking advantages of WWW-solutions to outstanding ACL issues (e.g., security)

77

## Advantages of XML-encoding ACL messages (continued)

- ACL messages introduce a pragmatics layer that is unaccounted at the semantic level
- Using XML, helps better address these pragmatic aspects through the use of links. Links point to additional information.
  - Links can assist with the ontological problem (defining and sharing ontologies)
  - Links can point to agent capability and identity information, protocols, even semantics.

78

# Mobile Agents and ACLs

79

## Agents and Mobile Agents: Exploring the differences

- Different intellectual origins
  - Distributed Computing, Process Migration, Distributed Objects for Mobile Agents
  - Artificial Intelligence, Planning, Logic, DAI, MAS
- Different tools and methodologies
  - Object-Oriented Languages
  - Traditional AI languages and languages stemming from BDI (Belief, Desire, Intentions) logics
- Different implementation concerns
  - Execution environment
  - Knowledge sharing, Planning, Coordination

80

## New trends in the Agent Community

- Java becomes the language of choice
  - a shift from AI languages to OO languages
- API's for ACL's focus on "conversations"
  - conversations, like protocols, define sequences of messages that may be exchanged
- These changes signal a departure from traditional AI-minded approaches
  - emphasis is on "behavior" than internal details

81

## How does an agent speak an ACL?

For any agent to "speak" an ACL the following have to be provided:

- API's for the composition, sending and receiving of messages
- infrastructure for naming, registering, finding other agents
- agent-dependant code for processing ACL messages depending on the message type

82

# Conversations

83

## What are the Conversations?

- Desirable sequences of messages for particular tasks.
- They indicate where/how messages "fit" in exchanges.

84

# Advantages of Conversations

- Allow more intuitive and convenient method for handling messages in context.
- Through conversation composition, scale to varying levels of granularity.
- Provide conversation management independent of agent implementation.
- Facilitate communication through conversation sharing.

85

# Addressing the shortcomings of the semantics with conversations

- Both KQML and FIPA ACL include specifications for conversations (or conversation protocols)
- Conversations are not part of the semantic definition of the ACL
- Conversations shift the focus to an agent's observable behavior
- Programmers might find conversations more useful than formal semantics
- The meaning of primitives is often context/situation dependant and conversations can accommodate context

86

# Conclusions

87

## Some key ideas

- Software agents offer a new paradigm for very large scale distributed heterogeneous applications.
- The paradigm focuses on the interactions of autonomous, cooperating processes which can adapt to humans and other agents.
- Agent Communication Languages are a key enabling technology
  - Mobility is an orthogonal characteristic which many, but not all, consider central.
  - Intelligence is always a desirable characteristic but is not strictly required by the paradigm.
- The paradigm is still forming and ACLs will continue to evolve.

88

## Agent Communication

- Agent-agent communication is a key to realizing the potential of the agent paradigm.
- Since interoperability is a defining characteristic of agents, standards are important!
- Candidates for standardization include
  - Agent architecture
  - Agent communication language
  - Agent interaction protocols
  - Agent knowledge
  - Agent programming languages
- Standards will most develop through natural selection, *"nature red in tooth and claw"*

89

## Agent Methodology

The KSE offers a four-part methodology for for developing complex agent-based systems:

- Collect/construct necessary ontologies
  - Use standard, published ontologies if possible
  - Develop (and publish) new components as needed
  - Use common tools, e.g. **Ontolingua**, GFP, ...
- Choose common representation language(s)
  - e.g., SQL or KBMS with **KIF** is a recommended default
- Use an ACL like **KQML** as communication language
  - extend with new performatives and protocols as needed
- Identify and define new higher-level protocols
  - e.g., for negotiation, purchasing, cataloging, etc.

90

# What's Needed Tomorrow

- **Further develop semantics of ACLs**
  Common content languages and ontologies
  A language for describing agent actions, beliefs, intentions, etc.
- **Agent ontologies**
  Sharable ontologies for agent properties, behavior, etc
- **Better handle on metadata**
  Abstractable and applicable to many content languages
- **Declarative and learnable protocols**
  Languages for defining higher-level protocols based on more primitive ones
- **Practical agent knowledge sharing**
  "Social" mechanisms for distributing information and knowledge
  Viewing knowledge sharing as mobile declarative code?
- **Frameworks for controlling collections of agents**
  E.g., artificial markets, natural selection, etc.

91

# For More Information

- **General information on software agents**
  - http://www.cs.umbc.edu/agents
- The FIPA home
  - http://www.fipa.org/
- Information on KQML, KIF, ontologies
  - http://www.cs.umbc.edu/kqml
  - http://www.cs.umbc.edu/kif
  - http://www.cs.umbc.edu/ontology/
- Information in Agent Communication Languages
  - http://www.cs.umbc.edu/acl/

92

# Presenters

93

## Tim Finin

**http://umbc.edu/~finin**
**finin@cs.umbc.edu**

Dr. Timothy Finin is a Professor in the department of Computer Science and Electrical Engineering and director of the Institute for Global Electronic Commerce at the University of Maryland Baltimore County (UMBC). He has over 25 years of experience in the applications of AI to information systems, intelligent interfaces and robotics and is currently working on the theory and applications of intelligent software agents. He holds degrees from MIT and the University of Illinois and has held positions at Unisys, Lockheed-Martin, the University of Pennsylvania, and the MIT AI Laboratory. Finin is the author of over one hundred refereed publications and has received research grants and contracts from a variety of sources. He has been the past program chair or general chair of several major conferences, including the IEEE Conference on Artificial Intelligence for Applications, The ACM Conference on Information and Knowledge Management, and the ACM Autonomous Agents conference. He is a former AAAI councilor and is currently serving as AAAI's representative on the board of directors of the Computing Research Association.

94

# Yannis Labrou

**http://www.cs.umbc.edu/~jklabrou**
**jklabrou@cs.umbc.edu**

Dr. Yannis K. Labrou is a Research Assistant Professor of Computer Science and Electrical Engineering at the University of Maryland Baltimore County. He holds a BSc degree in Physics from the University of Athens, and he received his PhD in Computer science at UMBC in 1996. His dissertation addressed the issue of semantics for an agent communication language and in particular, the specification and semantics of KQML. Dr. Labrou is a founding member of the FIPA academy and has been an active participant in the development of the FIPA specification. Prior to joining UMBC, Dr. Labrou worked as an intern at the Intelligent Network Technology group of the I.B.M. T.J. Watson Research Center.

95