Multiagent Systems for Service-Oriented Computing

- Challenge: Organizing a decentralized computation
  - What services constitute a service engagement
  - Who provides what services to whom
  - Without the benefit of a central designer for all services
- Solution: Interacting and communicating
  - Trade off prior agreement with formal reasoning about specifications
  - Specify interaction protocols that describe desired interoperation
  - Design agents to participate in specified protocol
  - Potentially enable agents to negotiate agreements dynamically
- Specialized protocols
  - Negotiation
  - In cooperative, homogeneous setting: maintaining consistency

Agents in Service-Oriented Computing

Breakdown of functionality

- User assistance
- Application adapters
- Directory and ontology
- Brokerage
- Resources: Web, databases, ...
- Process planning and execution

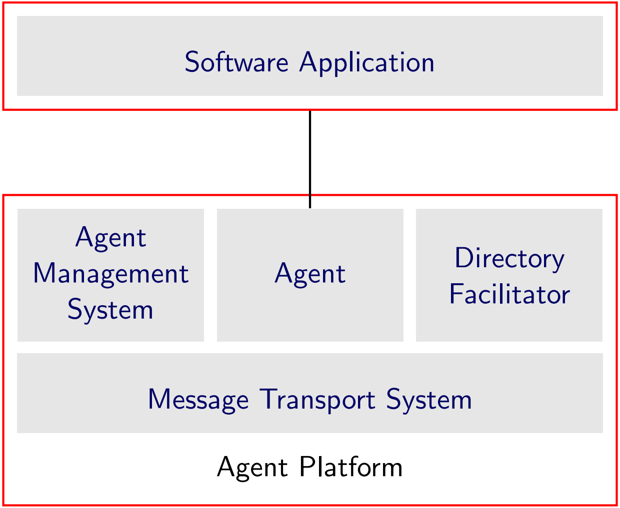
Brokerage

- Cooperates with a Directory Service
- Accepts requests from agents to recruit one or more agents who can provide a service
- Uses knowledge about the requirements and capabilities of registered agents to
  - Identify appropriate agents for an interaction
  - Negotiate with selected agents
  - Potentially learn models of the responses
  - Example: Brokerage determines that advertised results from agent X are incomplete and seeks a substitute for X

FIPA Agent Management System

Foundation for Intelligent and Physical Agents (now in IEEE)

- Good: architecture
  - Highlights agents and interaction
- Wrong: mentalist focus
- Wrong: Over-constrained protocols
- Wrong: Already obsolete low-level details



Agent Management System Functions

Analogous to a Java Enterprise Edition Container

Handles the creation, registration, location, communication, migration, and retirement of agents

- White pages, e.g., agent location and naming
  - Agent identifiers support social names, transport addresses, name resolution services
- Yellow pages, e.g., service location and registration services, from Directory Facilitator
- Agent message transport services

Multiagent Frameworks

- JADE, a popular FIPA-compliant agent framework for multiagent systems:
  - http://jade.tilab.com/
- Jadex: JADE plus BDI constructs
- JaCaMo: Combines three programming approaches
  - Jason: BDI constructs
  - Cartago: Environment artifacts
  - Moise: Organizations (later Moise+)
- Janus http://www.janusproject.io/
  - Comes with the SARL agent-oriented programming language
- Inactive projects: FIPA-OS, Jack, Zeus

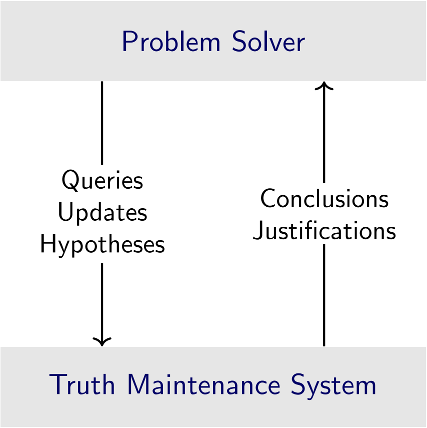
Consistency Maintenance across Services

- A truth maintenance system (TMS) maintains a knowledge base
  - Performs a form of propositional deduction
  - Maintains justifications and explains the results of its deductions
  - Updates beliefs incrementally when premises change
- Therefore, a TMS
  - Ensures the knowledge base remains consistent
  - Ensures all updates propagate before any queries are evaluated

TMS Architecturally

Provides an abstraction analogous to, but more sophisticated than, a database

- Problem solver: decides on actions
- TMS: maintains a network of beliefs
  - Justifications of a belief based on inference rules and other beliefs
  - Propagates updates due to revisions in rules and beliefs (premises)



Knowledge Integrity

Nontrivial when knowledge is distributed

Property	**Meaning**
Stability	Believe everything justified validly Disbelieve everything justified invalidly
Well-Foundedness	Beliefs are not circular, meaning the justifications bottom out
Consistency	No logical contradictions
Completeness	Find a consistent state, if any

Distributed TMS

- Each agent has a justification-based TMS
- Each datum can have status
  - OUT (not believed)
  - IN: valid local justification (believed)
  - EXTERNAL: must be IN for some agent
- When a problem solver adds or removes a justification, the DTMS determines whether any datum is affected
- In case of updates,
  - Unlabels data based on the changed datum
  - Relabels all unlabeled shared data (in one or more iterations)
  - Notifies agents with whom the datum is shared

Degrees of Logical Consistency

- Inconsistency: an agent is internally inconsistent
  - All bets are off with such an agent
- Local Consistency: all agents are individually consistent
  - Totally disconnected agents can't interact effectively
- Global Consistency: union of KBs is consistent
  - Total integration is not viable in open settings
- Local-and-Shared Consistency (for the DTMS): agents are locally consistent and agree about any data they might share
  - Captures essential interdependence

Knowledge Inconsistency Examples

Form of Inconsistency	Example
Both a fact and its negation are believed	Believe the goods have been delivered and believe the goods have not been delivered
A fact is both believed and disbelieved	Believe the goods have been delivered and not believe the goods have been delivered
An object is believed to be of two incompatible types	Believe PO-99 is a purchase order and believe PO-99 is a request for quotes
Distinct objects are believed to be identical	Believe PO-99 and PO-98 are the same resource when they are not
Cardinality constraints of relationships are violated	Believe C's shipping address is A_1 and believe C's shipping address is A_2 and believe that $A_1 \neq A_2$ and believe that shipping addresses are unique

Initial States of Knowledge Bases of Interacting Agents

Patient

f3: need(dentist Yes)

r3: Infer schedule(?X ?Y) from query(Friend recommend(?X ?Y)) and need(?X Yes)

? recommend(dentist ?Y)

Friend

f1: need(dentist No)

f2: reputed(dentist Dennis)

r1: Infer recommend(?X ?Y) from qualified(?X ?Y)

r2: Infer qualified(?X ?Y) from reputed(?X ?Y)

Response to Patient's Query

Patient

f3: need(dentist Yes)

r3: Infer schedule(?X ?Y) from query(Friend recommend(?X ?Y)) and need(?X Yes)

f4: recommend(dentist Dennis)

Status: EXTERNAL; Justification: (); Shared with: Friend

f5: schedule(dentist Dennis)

Status: IN; Justification: (f3 f4 r3)

↑ recommend(dentist Dennis)

Friend

f1: need(dentist No)

f2: reputed(dentist Dennis)

r1: Infer recommend(?X ?Y) from qualified(?X ?Y)

r2: Infer qualified(?X ?Y) from reputed(?X ?Y)

f3: recommend(dentist Dennis)

Status: IN; Justification: (f2 r1 r2); Shared with: Patient

Withdraw Recommendation

Patient

f3: need(dentist Yes)

r3: Infer schedule(?X ?Y) from query(Friend recommend(?X ?Y)) and need(?X Yes)

f4: recommend(dentist Dennis)
    Status: OUT; Justification: (); Shared with: Friend

f5: schedule(dentist Dennis)
    Status: OUT; Justification: (f3 f4 r3)

↑ Relabel recommend(dentist Dennis)

Friend

f1: need(dentist No)

f2: reputed(dentist Dennis) $\longrightarrow$ OUT

r1: Infer recommend(?X ?Y) from qualified(?X ?Y)

r2: Infer qualified(?X ?Y) from reputed(?X ?Y)

f3: recommend(dentist Dennis)
    Status: OUT; Justification: (f2 r1 r2); Shared with: Patient

Distributed TMS Applicability

- Presumes the agents are cooperative and adopt the same representation
- Ensures consistency with respect to shared data
  - Considers **one** state of the world
  - The agents may learn or unlearn data about the same state
- Not suitable for dealing with a changing world
  - Cannot deal with real-world actions
  - Can undo reasoning steps but not actions

Summary: Multiagent Systems

Interactions among agents enable interoperation necessary in service engagements

- Communication among agents is key
- Programming environments can support agent interactions
- In cooperative settings, consistency maintenance is a useful utility
- To intelligently cooperate or compete, agents must model each other
  - Such modeling requires complex representations and reasoning
- The guarantees we achieve without relying upon agent internals are the most robust
  - Correspond to interaction protocols for interoperation
  - Yield loose coupling
  - ... The next topic