Programming e-commerce applications in CLAIM

Amal El Fallah-Segrouchni
Amal.Elfallah@lip6.fr

Alexandru Suna
Alexandru.Suna@poleia.lip6.fr

University of Paris 6

Talk Outline

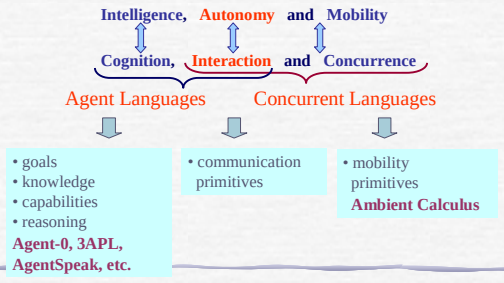
- Motivations
- CLAIM
  - Characteristics
  - Agent's Components
- SyMPA
- E-commerce application
- Conclusion and future work

Motivations

- Think and implement in the same paradigm: agent
- Meet the requirements of:
  - distribution
  - migration

Our CLAIM

An agent oriented programming language that homogeneously combines:



CLAIM: Computational Language for Autonomous, Intelligent and Mobile agents

- Declarative language
  - Specify the interactions and the mobility
    - Communication primitives
    - Mobility primitives — *Ambient Calculus*
  - Cognitive elements
    - Goals, knowledge, capabilities
    - Forward reasoning : reactivity
    - Backward reasoning : planning — *Agent Oriented Languages*

Operational semantics
- it is based on the *ambient calculus* and the *safe ambients*
- it manages the mobility and interaction
- it takes into account the specificity of cognitive agents

Agent Description

Reactive behavior

Goal-driven behavior

```
defineAgent  agentName  {
        authority = agentName ;
        parent = null | agentName ;
        knowledges = null | { (knowledge;)*} ;
        goals = null | { (goal ;)* } ;
        messages = null | { (queueMessage ;)* } ;
        capabilities = null | { (capability ;)* } ;
        processes = null | { (process | )* } ;
        agents = null | { (agentName ;)* } ;
}
defineAgentClass  className (args)  { ... }
newAgent  agentName:className(args)
        variable : ?x
```


Agents' Communication

send (receiver, message)

queueMessage = agentName > message

⇨ ***receiver:***
- *agentName*
- ***all*** (broadcast)
- ***this***
- *?Ag:className* (multicast)

⇨ ***message:***
- *proposition*
- **messages concerning knowledge**
  tell(*knowledge* ***), askAllCapabilities (), askIfCapabiliy (****capSign* ***),***
  achieveCapability(*capSign****), removeCapability(****agName,capSign****)***
- **mobility messages**
  openBy(), openOK(), openNotOK(), wantIn(*mobArg****), inOK(****mobArg****),***
  inNotOK(*mobArg****),wantOut(****mobArg****), outOK(****mobArg****), outNotOK(****mobArg****)***

Capabilities

⇨ ***capabilities :***

```
capability = capabilitySignature  {
                message = null | message ;
                conditions = null | condition ;
                do { process };
                effects = null | { (effect ;)*}
             }
condition =  Java(objectName.function(args))
           | agentName.effect
           | hasKnowledge (knowledge)
           | hasAgent (agentName)
           | not (condition)
           | and (condition,(condition)+)
           | or (condition,(condition)+)
```

Processes

⇨ ***processes :***

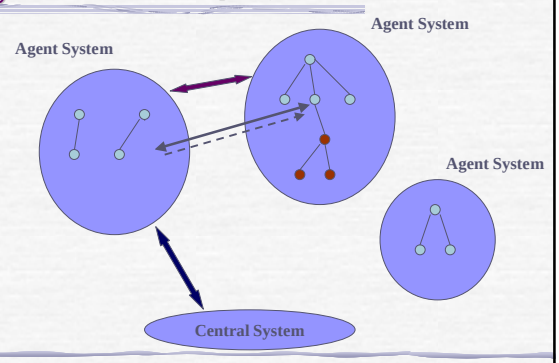
P ::=

Q.R
instruction
*?x=(value| **Java**(objectName.function(args)))*
***Java**(objectName.function(args))*
***newAgent** agentName**:**agentClass (args)*
in (** mobilityArg,agentName **)
out (** mobilityArg,agentName **)
moveTo(** mobilityArg,agentName **)
open (** agentName **)
acid
send (** receiver, message**)

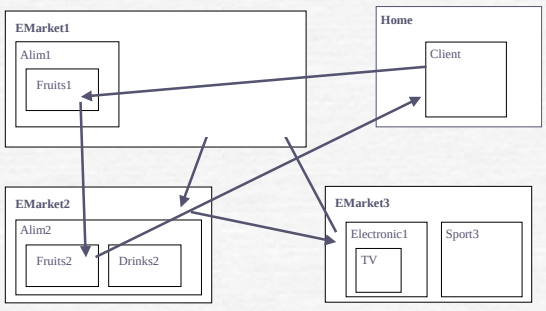
this
| clone
| process

***forAllKnowledge** (knowledge){ P }*
***forAllAgents** (agentName){ P }*

SyMPA (MASIF compliant)



E-commerce application



Implementation in CLAIM (1)

```
defineAgentClass Domain(?prod) {
  authority=null; parent=null; knowledge={hasProduct(?prod);}
  goals=null; messages=null;
  capabilities={
    ...
    createMarketSearcher(?prod) {
      message=null;
      condition=not(this.hasClients(?prod));
      do{newAgent M:MarketSearcher().send(M,search(?prod))}
      effects=null;
    }
    goToNewMarket() {
      message=bestM(?p,?bestEM);
      condition=not(this.hasClients(?p));
      do{moveTo(this,?bestEM).send(?bestEM,newDomain(?p))}
      effects=hasClients(?p);
    }
  }
  processes=null;agents=null;
}
```

Implementation in CLAIM (2)

```
defineAgentClass MarketSearcher(?prod) {
  authority=null; parent=null; knowledge=null;
  goals=null; messages=null;
  capabilities={
    ...
    migrate(?prod) {
      message=null;
      condition=and(not(authority.hasClients(?p)),
                    Java(ObjT.wait(30));
      do{?bestEM=parent.?c=0.forAllKnowledge(?EM:Emarket()){
          moveTo(this,?EM).
          ?c=Java(MS.countClients(?p,?c,?bestEM)}
         moveTo(this,authority).
         send(authority,bestM(?bestEM)).acid
      }
      effects=null;
    }
  }
  processes=null;agents=null;
}
```


Conclusion: Main Contribution

CLAIM language

- Goals, Knowledge, Capabilities
- Forward reasoning: reactive behavior
- Backward reasoning : goal driven behavior
- Communication primitives
- Mobility primitives

Expressiveness: e-commerce, load balancing, information research on web

SyMPA system

- Easy design of distributed MAS
  - editing agents, interpret, agent interface
- Suitable platform for implementation
  - management, creation, execution, authentication, migration of agents
  - protocols for communication and mobility

Future work

- Operational semantics of CLAIM and planning
- Enrich the language
  - add security primitives and data types
  - call functions/subroutines defined in other languages
- Extend de language possibilities
  - develop libraries of agents
  - endow agents with learning capabilities
- Improve SYMPA
  - improve the security
  - add fault tolerance mechanisms
  - offer several mechanisms for the management of agents and agent systems