Jason and $\mathcal{M}$OISE$^+$

Organisational Programming in the Agent Contest 2008

Jomi F. Hübner[†], Rafael H. Bordini[⋆]
Gauthier Picard[†]

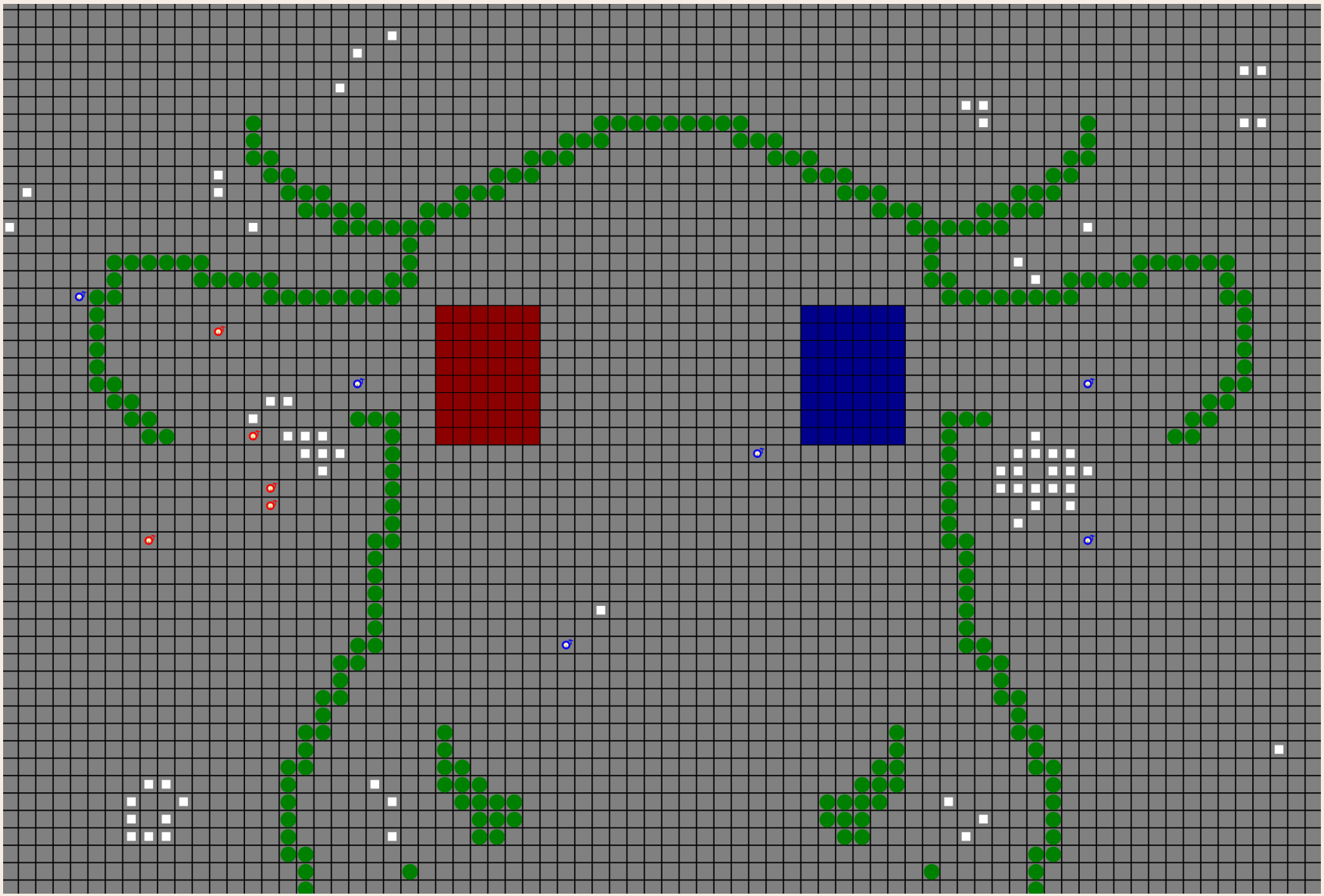
[†] ENS Mines Saint Etienne, France
{hubner,picard}@emse.fr

[⋆] University of Durham, UK
r.bordini@durham.ac.uk

Dagstuhl Seminar on Programming Multi-Agent Systems

Outline

Agent Contest 2008
the Cows and Herders scenario



Objectives of our participation

- In 2006: program our agents using plans
  - $\therefore$ reactive agents
- In 2007: program our agents using goals
  - $\therefore$ goal directed agents
- In 2008: program our agents using organisation
  - $\therefore$ join agent and system levels
  - $\rightsquigarrow$ use **Jason** for the agents
  - $\rightsquigarrow$ use $\mathcal{M}$OISE$^+$ for the organisation
- Test and improve **Jason** and $\mathcal{M}$OISE$^+$ software
- Evaluate the use of organisational constructors in the development of the team

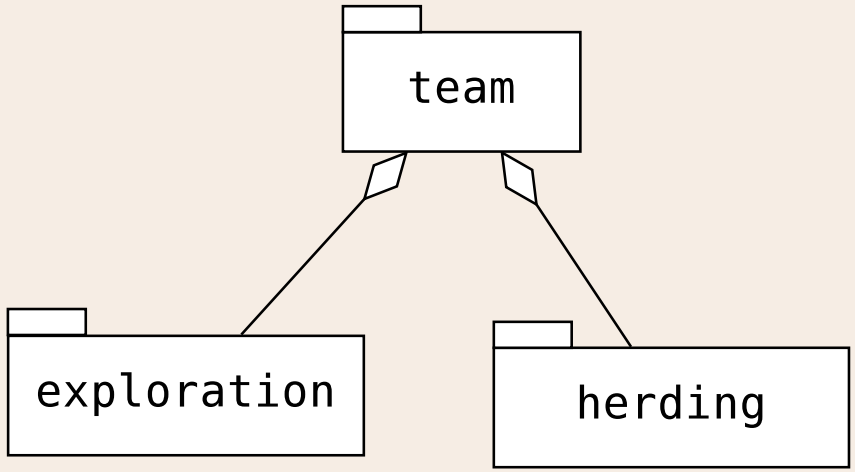
Objectives of our participation

- In 2006: program our agents using plans
  - $\therefore$ reactive agents
- In 2007: program our agents using goals
  - $\therefore$ goal directed agents
- In 2008: program our agents using organisation
  - $\therefore$ join agent and system levels
  - $\rightsquigarrow$ use **Jason** for the agents
  - $\rightsquigarrow$ use $\mathcal{M}$OISE$^+$ for the organisation
- Test and improve **Jason** and $\mathcal{M}$OISE$^+$ software
- Evaluate the use of organisational constructors in the development of the team

Team specification — Groups

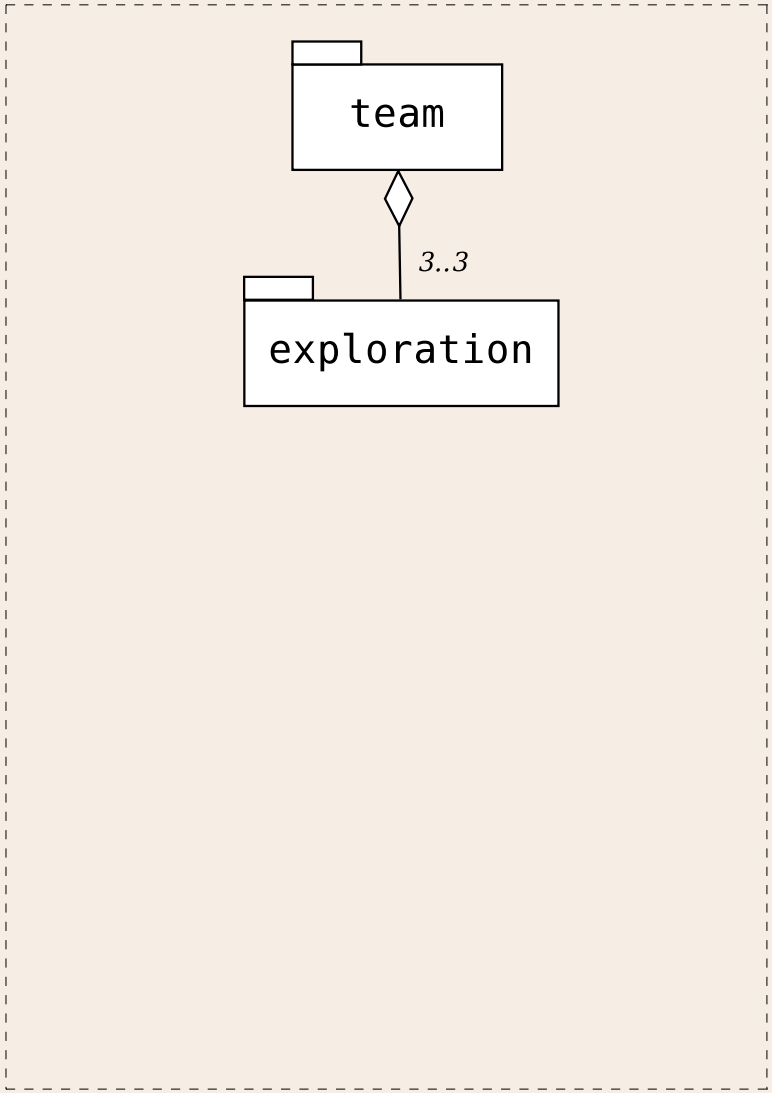
Our agents are organised in two types of groups:

- Exploration group: find cows
- Herding group: push cows into the corral



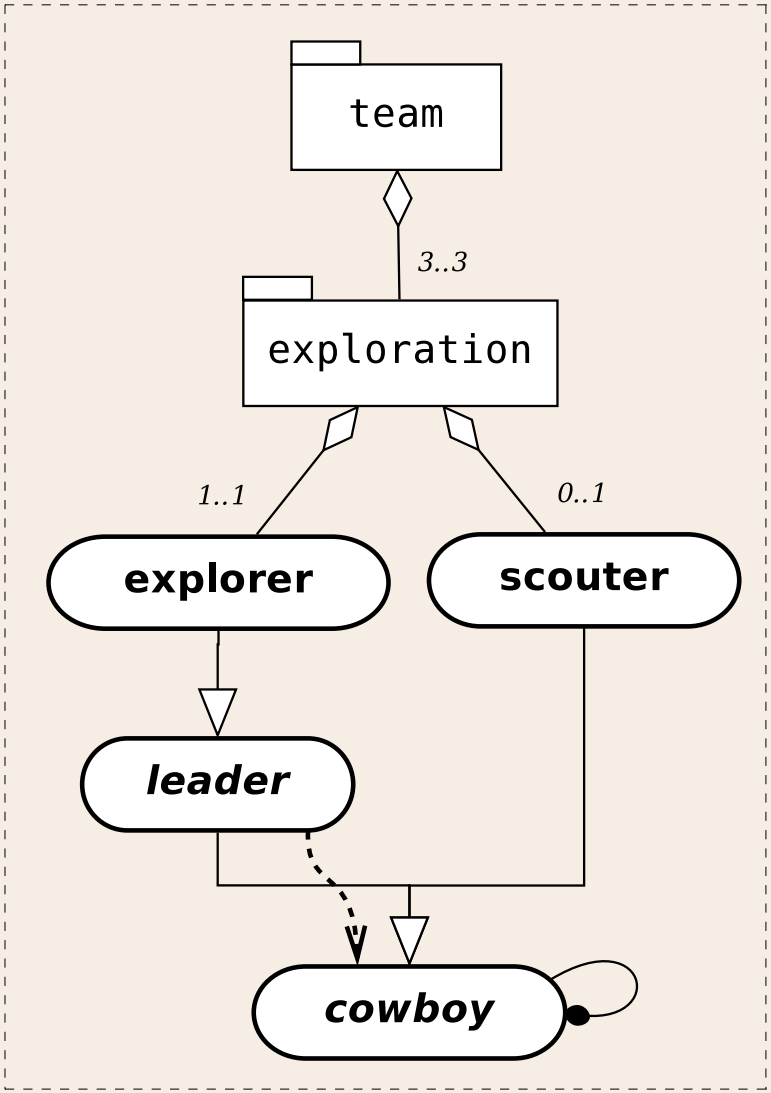
Exploration group

- Three instances of exploration group
- Each group is allocated to an area of the scenario
- One explorer: decides where to go
- One scouter: follows explorer



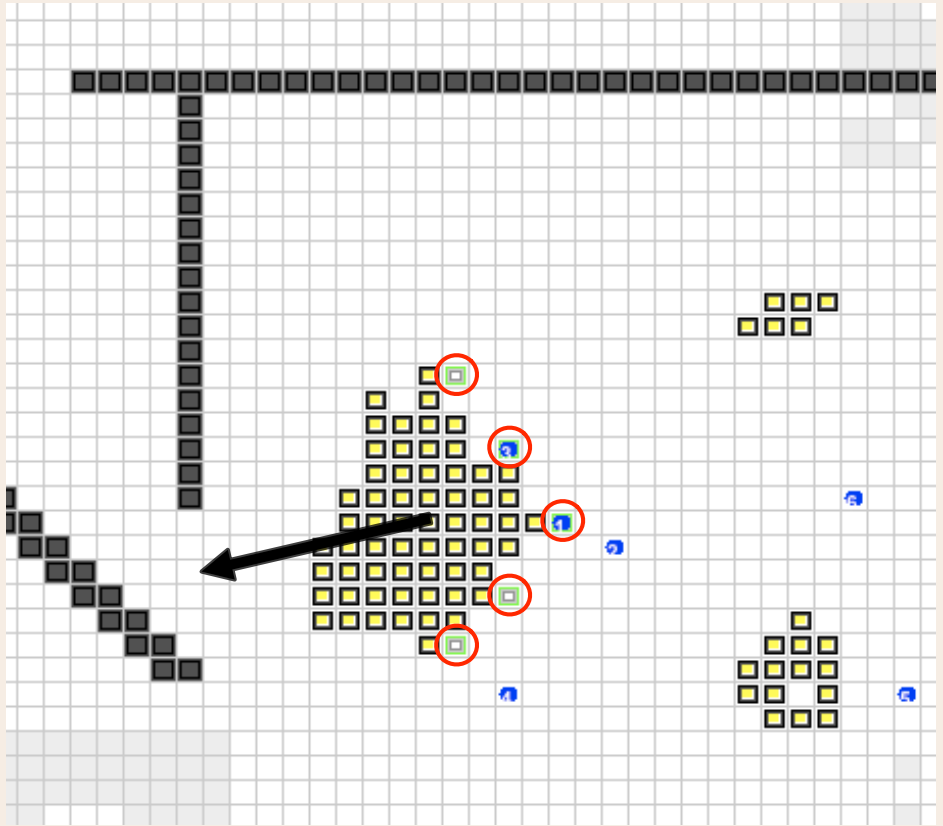
Exploration group

- Three instances of exploration group
- Each group is allocated to an area of the scenario
- One explorer: decides where to go
- One scouter: follows explorer



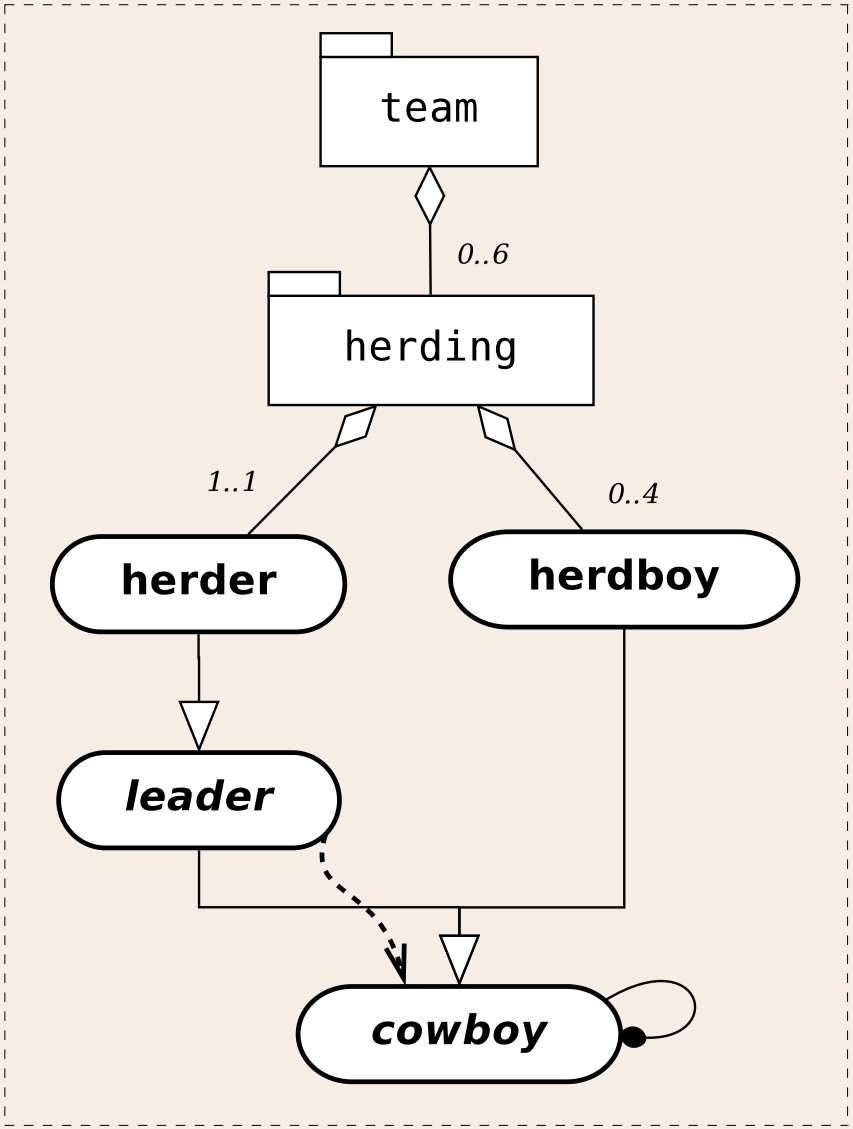
Herding group

- Created when a member of the team sees some cow
- As many instances as the number of clusters of cows
- One herder:
  difines the team formation for the cluster
- scouters:
  be in formation
- spatial coordination



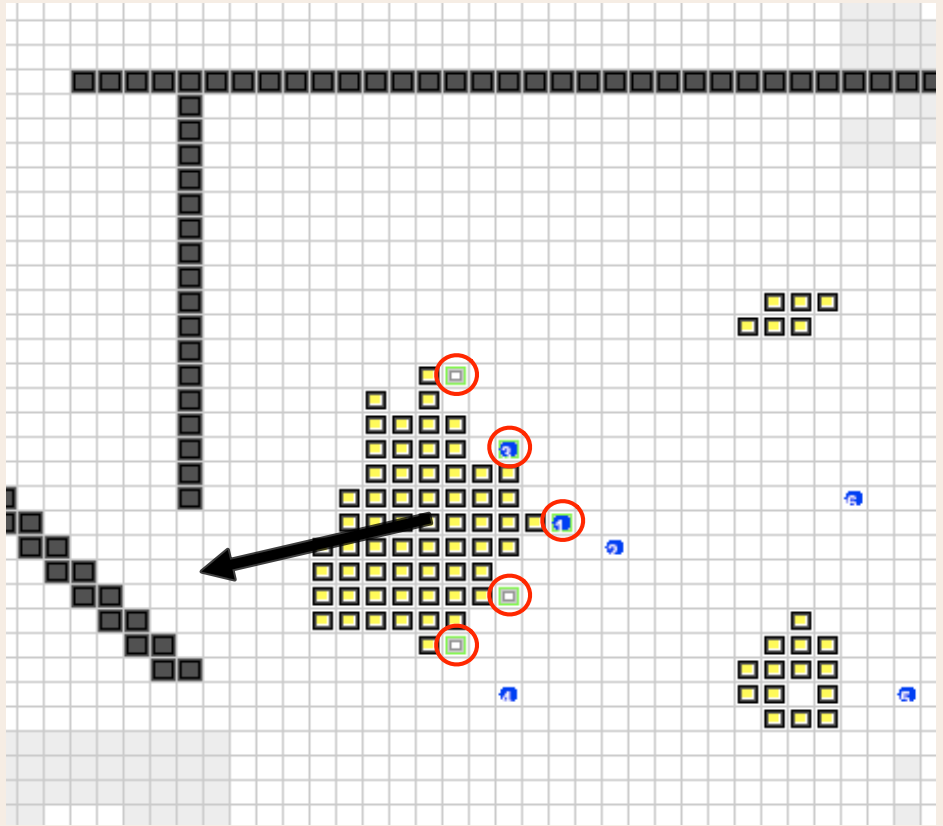
Herding group

- Created when a member of the team sees some cow
- As many instances as the number of clusters of cows
- One herder:
  difines the team formation for the cluster
- scouters:
  be in formation
- spatial coordination



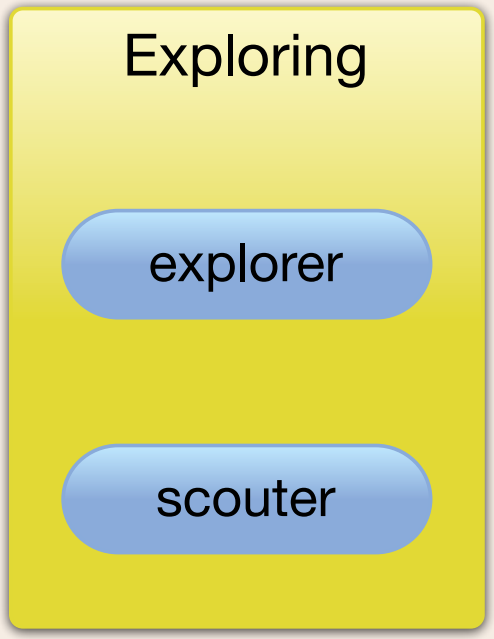
Herding group

- Created when a member of the team sees some cow
- As many instances as the number of clusters of cows
- One herder:
  difines the team formation for the cluster
- scouters:
  be in formation
- spatial coordination



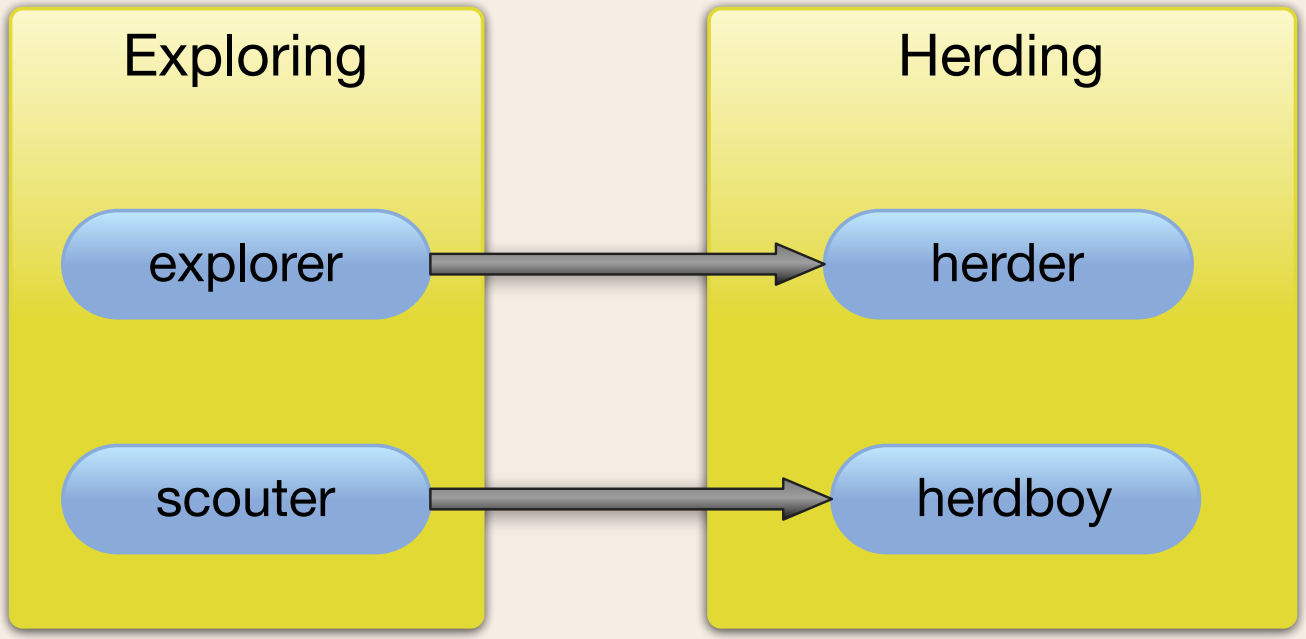
Structural Dynamics

Start exploring



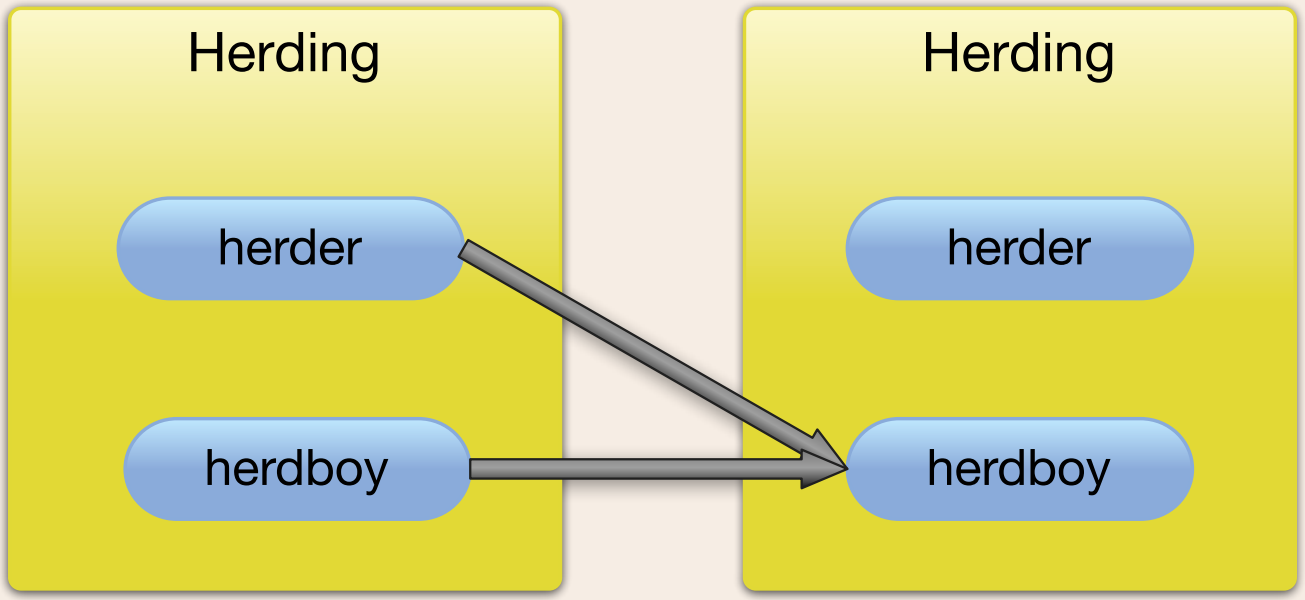
Structural Dynamics

Creation of herding group



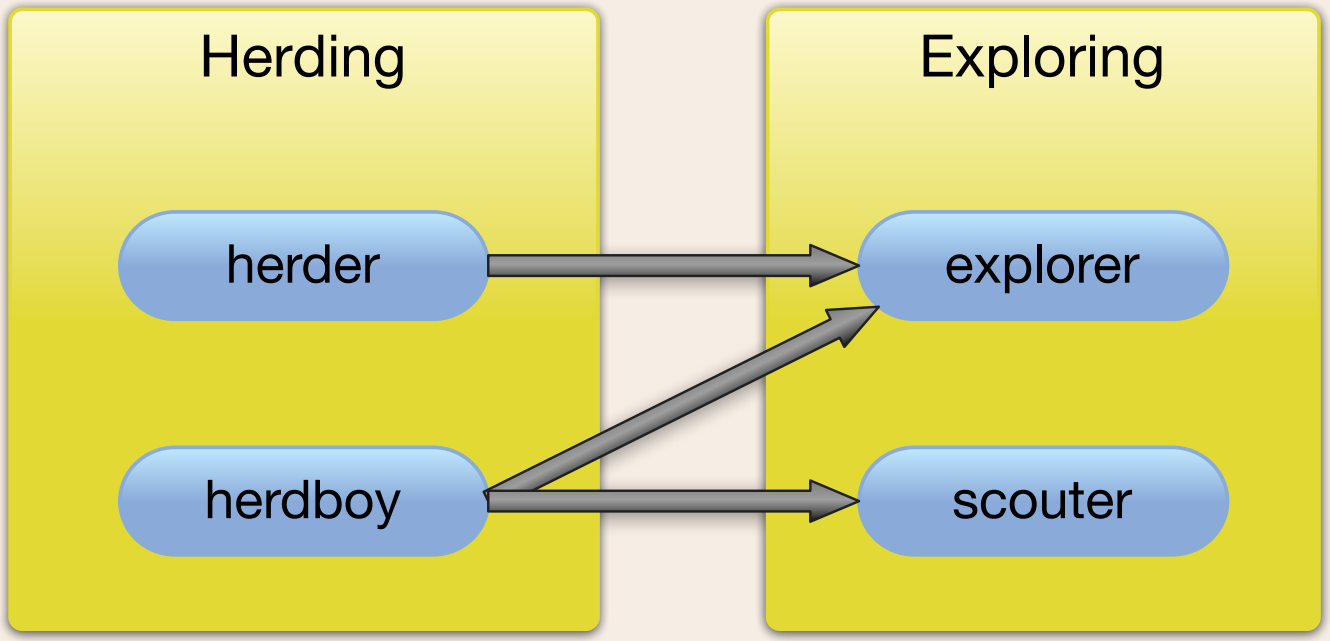
Structural Dynamics

Merge two herding groups



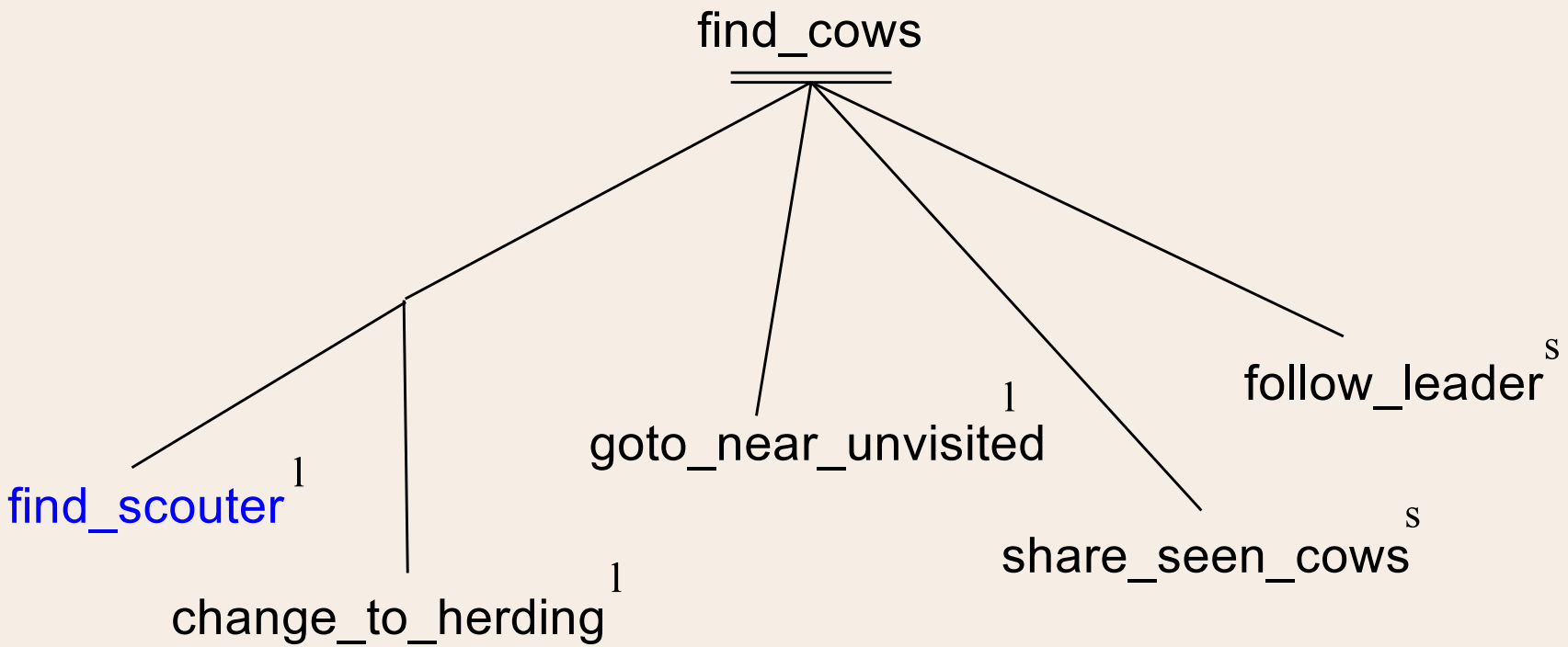
Structural Dynamics

Dissolve herding group



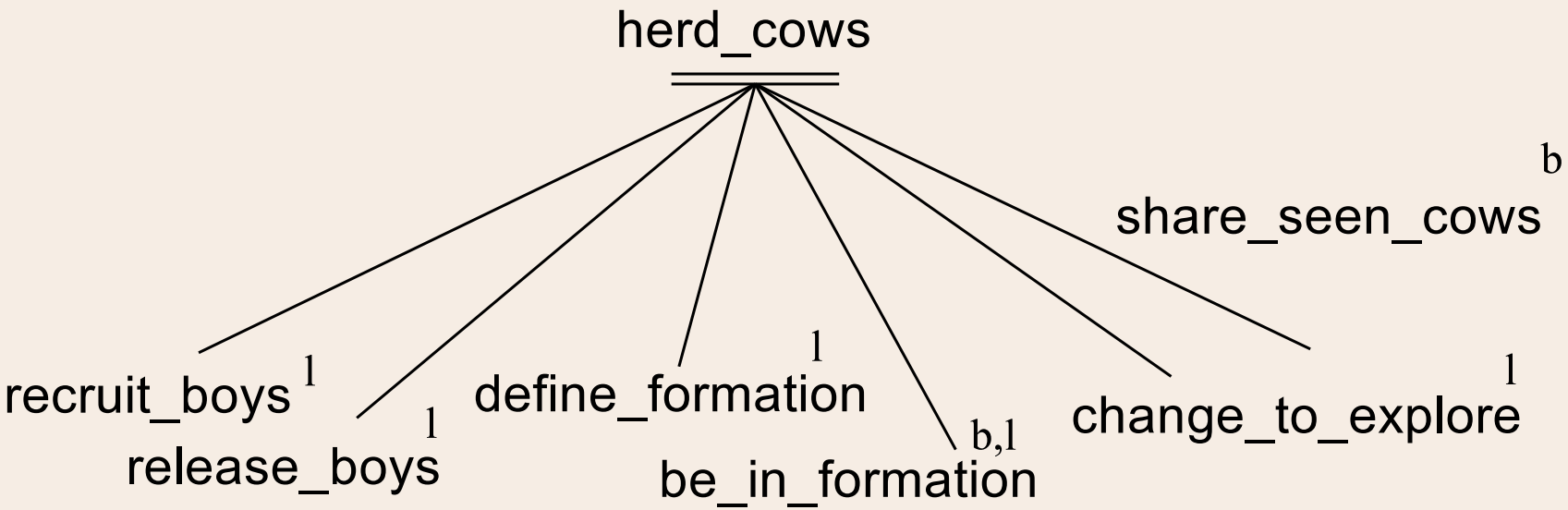
...

Functional specification — Scheme to explore



Explorer (the leader) is obligated to mission *l*
Scouter is obligated to *s*

Functional specification — Scheme to herd



Herder (the leader) is obligated to mission *l*
Herdboy is obligated to *b*

Goals

Role	Goal	Goal Description
explorer	find_scouter	find agent nearby to play scouter
	change_to_herding	change to a herding group
	goto_near_unvisited	go to the nearest unvisited location
scouter	share_seen_cows	share information about cows
	follow_leader	follow the leader of the group
herder	recruit	recruit more herdboys
	release_boys	release some herdboys
	define_formation	compute the formation of the group
	be_in_formation	go to the place allocated to the agent
	merge	merge two herding groups
	change_to_exploring	change to an exploring group
herdboy	share_seen_cows	
	be_in_formation	

Organisation Oriented Programming I

```
...
<group-specification id="team">
  <sub-groups>
    <group-specification id="exploration_grp"
                         min="0" max="3" >
       <roles>
          <role id="explorer" min="1" max="1"/>
          <role id="scouter"  min="0" max="1"/>
       </roles>
    </group-specification>
...
```

Organisation Oriented Programming II

Tools to run the organisation:

- $\mathcal{S}$-$\mathcal{M}$OISE$^+$: organisational infrastructure
  - manage the state of the organisation
- $\mathcal{J}$-$\mathcal{M}$OISE$^+$: integration with **Jason**
  - library of organisational actions
  - organisational architecture

Organisation Oriented Programming III

- Agents are informed about their obligations
  $\therefore$ new goal event

Plan to handle a new goal — maintenance goal pattern

- the goal is annotated with the group and role that generated the obligation

```
+!define_formation[group(G),role(R)]
  <- ... <the code> ...

     // wait for the next cycle
     .wait("+pos(X,Y,Cycle)");

     // achieve that goal again
     !define_formation[group(G),role(R)].
```

Organisation Oriented Programming IV

- Agents are also informed by changes in the organisation
  $\therefore$ change the belief base
  $\therefore$ produce events

Examples

```
+play(Me,herder,G)
   : .my_name(Me)
  <- +group_leader(G,Me);
     .broadcast(tell, group_leader(G,Me)).

-group(Type,GroupId)
  <- .drop_intention(_[group(GroupId)]).
```

Agent Oriented Programming I

The achievement of organisational goals is implemented in **Jason**

Goal: merge herding group

plan merge

let g_i be my herding group
forall herding group g_j such that $g_i > g_j$ **do**
    let S_i be the set of cows of g_i's cluster
    let S_j be the set of cows of g_j's cluster
    if $S_i \cap S_j \neq \emptyset$ **then**
        remove group g_j from the organisation
        ask all agents of g_j to adopt the role *herdboy* in g_i
                                    `// new role → new goals`

Agent Oriented Programming II

Code in **Jason**

```
+!merge
  : .my_name(Me) &
    play(Me, herder, Gi) & // I play role herder
    current_cluster(MyC)   // MyC is the list with my cows

<- // for all other groups
    for (group_leader(Gj, L) & Me < L) {
      .send(L,askOne,current_cluster(_),current_cluster(TC));
      .intersection(MyC,TC,I);
      if (I \== []) {
         .send(L, achieve, change_role(herdboy,Gi))
      }
    }.
```

When the leader of other group change the role, he will ask his herdboys to also change the group

Object Oriented Programming

The following components were implemented in Java

- Integration with the contest simulator
  Agent perception and action
- Find paths: A*
- Compute the formation (a lot of vector calculations)
- ...

130 lines of code in $\mathcal{M}$OISE$^+$

696 lines of code in **Jason**

4218 lines of code in Java

Object Oriented Programming

The following components were implemented in Java

- Integration with the contest simulator
  Agent perception and action
- Find paths: A*
- Compute the formation (a lot of vector calculations)
- ...

130 lines of code in $\mathcal{M}$OISE$^+$

696 lines of code in **Jason**

4218 lines of code in Java

Useful tools — Mind inspector

play(gaucho1,herder,gr_herding_grp_13)[source(orgManager)].
play(gaucho4,herdboy,gr_herding_grp_13)[source(orgManager)].
play(gaucho5,herdboy,gr_herding_grp_13)[source(orgManager)].
pos(45,44,128)[source(percept)].
scheme(herd_sch,sch_herd_sch_18)[owner(gaucho3),source(orgManager)].
scheme(herd_sch,sch_herd_sch_12)[owner(gaucho1),source(orgManager)].
scheme_group(sch_herd_sch_12,gr_herding_grp_13)[source(orgManager)].
steps(700)[source(self)].
target(6,44)[source(gaucho1)].

- Rules

```
random_pos(X,Y) :-
    (pos(AgX,AgY,_418) & (jia.random(RX,40) & ((RX > 5) & ((X = ((RX-20)+AgX)) & ((X >
```

- Intentions

Sel	Id	Pen	Intended Means Stack (hide details)
	16927	suspended-self	+!be_in_formation[scheme(sch_herd_sch_12),mission(help
			+!be_in_formation[scheme(Sch),mission(Mission)]

Useful tools — $\mathcal{M}$OISE$^+$ GUI

jason-cowboys (Organisational Entity)

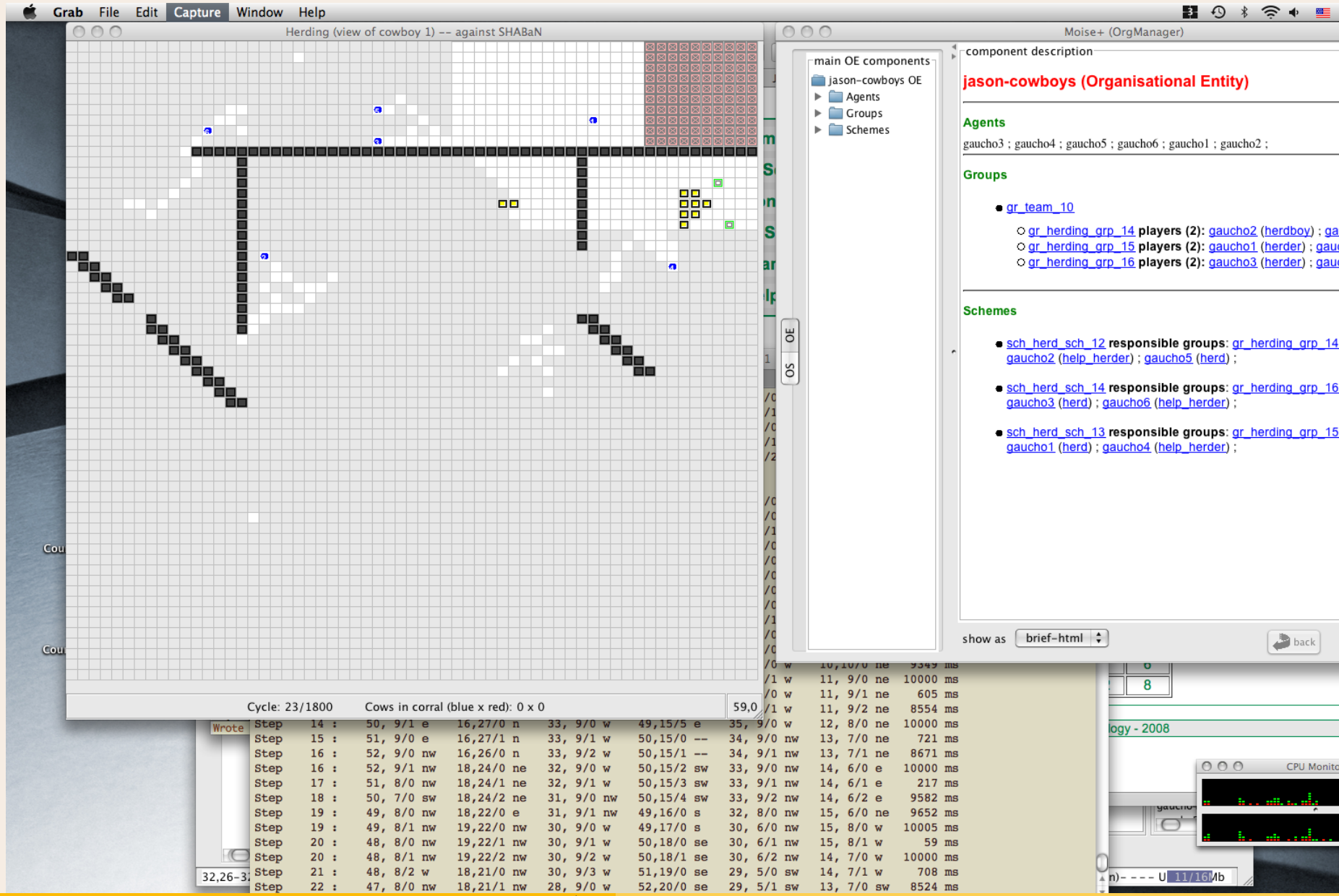
Agents

gaucho3 ; gaucho4 ; gaucho5 ; gaucho6 ; gaucho1 ; gaucho2 ;

Groups

- gr_team_01
  - gr_exploration_grp_02 **players (2):** gaucho3 (explorer) ; gaucho6 (scouter) ;
  - gr_herding_grp_05 **players (4):** gaucho1 (herder) ; gaucho2 (herdboy) ; gaucho4 (herdboy) ; gaucho5 (herdboy) ;

Typical screen



Typical development cycle

- have a brilliant idea
- code it
- basic test (JUnit, ASUnit, ...)
- watch the result in the contest scenarios
- read and analyse long logs, minds dumps, traces, performance...

  note that we need to analyse the execution of 6 concurrent agents
- find bugs (in the team, in $\mathcal{S}$-$\mathcal{M}$OISE$^+$, ...), start again
- tuning of parameters (the cluster size?), start again
- give up the idea, start again

Typical development cycle

- have a brilliant idea
- code it
- basic test (JUnit, ASUnit, ...)
- watch the result in the contest scenarios
- read and analyse long logs, minds dumps, traces, performance...

  note that we need to analyse the execution of 6 concurrent agents
- find bugs (in the team, in $\mathcal{S}$-$\mathcal{M}$OISE$^+$, ...), start again
- tuning of parameters (the cluster size?), start again
- give up the idea, start again

Typical development cycle

- have a brilliant idea
- code it
- basic test (JUnit, ASUnit, ...)
- watch the result in the contest scenarios
- read and analyse long logs, minds dumps, traces, performance...

  note that we need to analyse the execution of 6 concurrent agents
- find bugs (in the team, in $\mathcal{S}$-$\mathcal{M}$OISE$^+$, ...), start again
- tuning of parameters (the cluster size?), start again
- give up the idea, start again

Typical development cycle

- have a brilliant idea
- code it
- basic test (JUnit, ASUnit, ...)
- watch the result in the contest scenarios
- read and analyse long logs, minds dumps, traces, performance...

  note that we need to analyse the execution of 6 concurrent agents
- find bugs (in the team, in $\mathcal{S}$-$\mathcal{M}$OISE$^+$, ...), start again
- tuning of parameters (the cluster size?), start again
- give up the idea, start again

Typical development cycle

- have a brilliant idea
- code it
- basic test (JUnit, ASUnit, ...)
- watch the result in the contest scenarios
- read and analyse long logs, minds dumps, traces, performance...

  note that we need to analyse the execution of 6 concurrent agents
- find bugs (in the team, in $\mathcal{S}$-$\mathcal{M}$OISE$^+$, ...), start again
- tuning of parameters (the cluster size?), start again
- give up the idea, start again

Summary I

- Team
  - agents are autonomous to
    - adopt roles
    - decide how to achieve goals
  - coordination is essentially spacial (follow leader and formation)
  - communication is used to share information (speech act based)

Summary II

- **Jason**
  - declarative and goal oriented programming
  - goal patterns (maintenance goal)
  - meta-programming (`.drop_intention(_[group(g1)])`)
  - customisations (integration with the simulator and the organisation)
  - internal actions (code in Java)

  $\therefore$ good programming style

Summary III

- $\mathcal{M}$OISE$^+$
  - definition of groups and roles
  - allocation of goals to agents based on their roles
  - to change the team, we (developers) 'simply' change the organisation
  - global orchestration

  $\therefore$ team strategy defined at a high level

Good points

- New scenario of the contest
- Use of 3 programming paradigms
- Improve several issues of **Jason**, $\mathcal{M}$OISE$^+$, and their integration
  - New type of goal in $\mathcal{M}$OISE$^+$ (maintenance goal)
  - More suitable for collaborative systems (group deletion)

Weak points

- Too much time in 'debug, test, and tuning mode'
  we rather prefer analysis and programming
- The organisation dynamics is specified inside the agents
  it is coded and mixed in the agent's plans
  → new language to define it from a global perspective
- The functional dimension of the team is quite simple
  it allows the definition of global plans useful to achieve shared goals
  → more complex team strategies
  → changes in the scenario
- It is quite difficult to map an idea into different levels of analysis
  what is organisation and what is agent planning; what is $\mathcal{M}$OISE$^+$, **Jason**, or Java

Weak points

- Too much time in 'debug, test, and tuning mode'
  we rather prefer analysis and programming
- The organisation dynamics is specified inside the agents
  it is coded and mixed in the agent's plans
  → new language to define it from a global perspective
- The functional dimension of the team is quite simple
  it allows the definition of global plans useful to achieve shared goals
  → more complex team strategies
  → changes in the scenario
- It is quite difficult to map an idea into different levels of analysis
  what is organisation and what is agent planning; what is $\mathcal{M}$OISE$^+$, **Jason**, or Java

Weak points

- Too much time in 'debug, test, and tuning mode'
  we rather prefer analysis and programming
- The organisation dynamics is specified inside the agents
  it is coded and mixed in the agent's plans
  - $\longrightarrow$ new language to define it from a global perspective
- The functional dimension of the team is quite simple
  it allows the definition of global plans useful to achieve shared goals
  - $\longrightarrow$ more complex team strategies
  - $\longrightarrow$ changes in the scenario
- It is quite difficult to map an idea into different levels of analysis
  what is organisation and what is agent planning; what is $\mathcal{M}$OISE$^+$, **Jason**, or Java

Weak points

- Too much time in 'debug, test, and tuning mode'
  we rather prefer analysis and programming
- The organisation dynamics is specified inside the agents
  it is coded and mixed in the agent's plans
  $\longrightarrow$ new language to define it from a global perspective
- The functional dimension of the team is quite simple
  it allows the definition of global plans useful to achieve shared goals
  $\longrightarrow$ more complex team strategies
  $\longrightarrow$ changes in the scenario
- It is quite difficult to map an idea into different levels of analysis
  what is organisation and what is agent planning; what is $\mathcal{M}$OISE$^+$, **Jason**, or Java

More information

- `http://moise.sf.net`
- `http://jason.sf.net`
  (the code of our agents is available there)

- J. F. Hübner, J. S. Sichman, and O. Boissier. Developing organised multi-agent systems using the $\mathcal{M}$OISE$^+$ model: Programming issues at the system and agent levels. *Int. J.Agent-Oriented Software Engineering*, 1(3/4):370–395, 2007.