Formalization of Self-Organizing Multi-Agent Systems with Event-B and Design Patterns (Tool usage)

Zeineb GRAJA
zeineb.graja@irit.fr

Frédéric MIGEON, Christine MAUREL,
Marie-Pierre GLEIZES, Ahmed HADJ KACEM

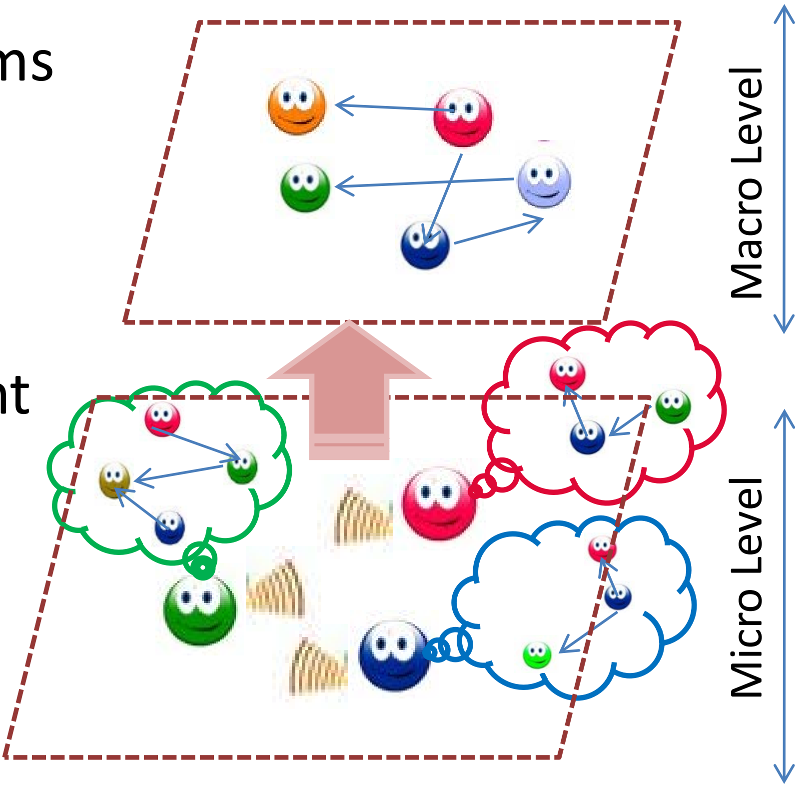
5th Rodin User and Developer Workshop
June 2-3, 2014 Toulouse - France

Talk outline

- Context & Motivations
- Formal Modelling of SOMAS
- Case Study: Foraging Ants
- Conclusion & Open Questions

Context

- Self-Organizing Multi-Agent Systems
  - Autonomous agents
  - Local information and interactions
  - The global function is emergent
- Bottom-up design
- Simulation & testing



How to ensure that the designed entities, when interacting together, will give rise to the desired behavior?

➔Need for rigourous approaches

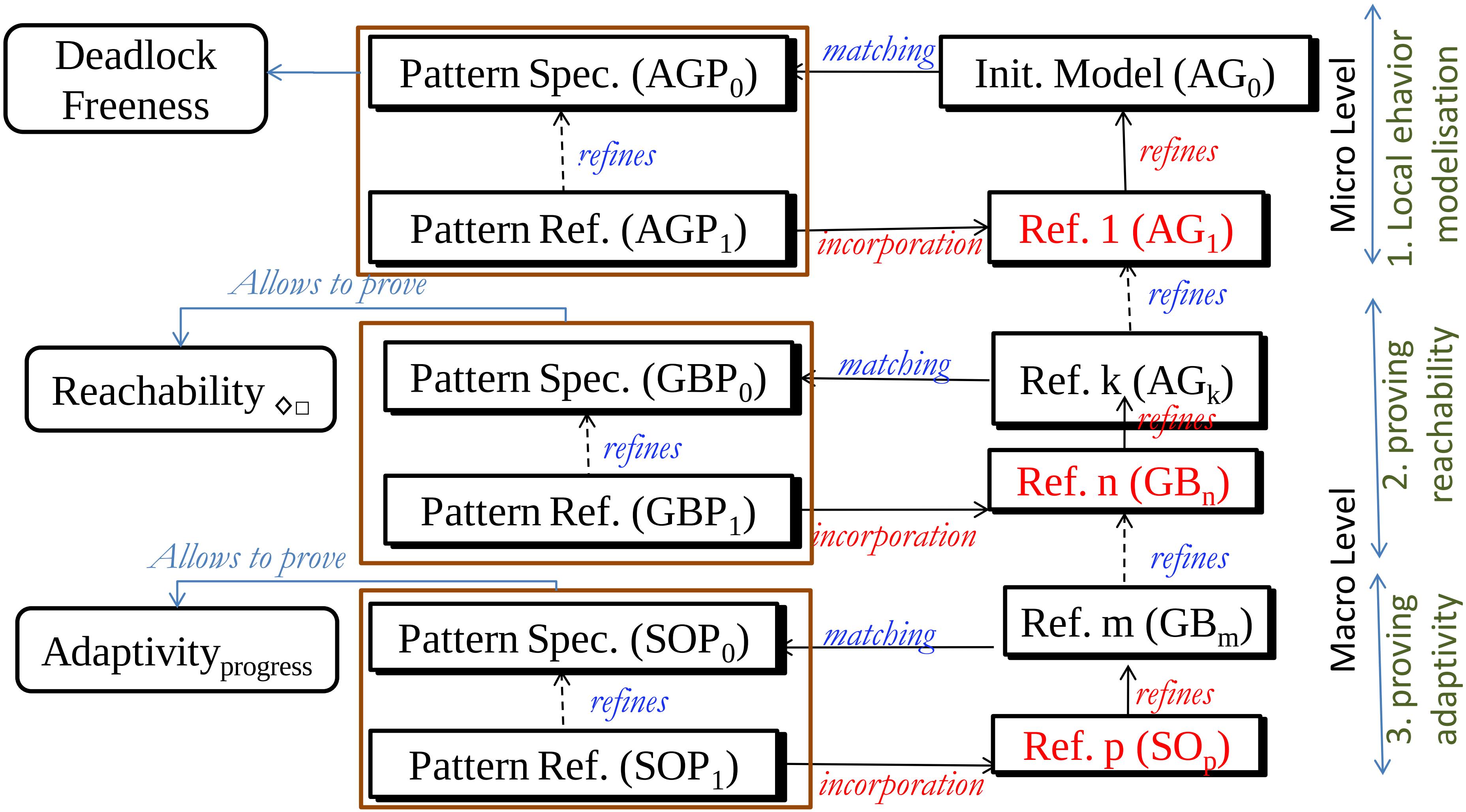
Motivations

➔Our proposition: Starting from a running self-organizing system providing emergent behaviour, we want to formally prove the obtained result

We use:

- Event-B as a modelling language
- Patterns giving guidance for refinement and proof
  - Reachability
  - Resilience
- Tools: Rodin platform and Patterns plug-in

Our Framework in a Nutshell



The Agent Pattern

- Goal: to design a correct local behavior of an agent type
- Pattern refinement

AGP0 Machine
Four events:
- Perceive
- Decide
- Act
- ActivEnv

First Refinement

AGP0_1 Machine
Refining the Act event

Second Refinement

AGP0_11 Machine
❶Introducing the agents' actuators
❷Refining the Deci
event
❸Refining the Act e
- Witnesses

➔ Deadlock free
the decision step

Third Refinement

AGP0_111 Machine
❶Introducing the sensors and the representations
❷Refining the Perceive event
❸ Refining the Decide event
➔Deadlock freeness in the perception step

The Global Behavior Pattern (1/2)

- Goal: reason about the convergence of the system

  Reach ≜◊ □ taskAchieved

- The proof pattern *GBP*
  - The machine GBP_0

```
variables GoalReached, SysState
Invariants
@inv1 GoalReached ∈ STATES → BOOL
@inv2 SysState∈STATES
```

```
anticipated event ObserveSuccess
  where
   @grd1 GoalReached(SysState)=TRUE
 end
```

```
Events
anticipated event Act refines Act
  any agent
  where
   @grd1 agent∈Agents
   @grd2 actionAgent(agent)≠noAction
   then
   @act1 modeAgent(agent):=nonActive
   @act2 GoalReached:|GoalReached'(SysState)∈BOOL
 end
```

The Global Behavior Pattern (2/2)

- The machine GBP_1 refines GBP_0

variables GoalReached, SysState, GoalReachabilityProgression
Invariants
@inv1 GoalReachabilityProgression ∈ STATES → ℤ
Variant GoalReachabilityProgression(SysState))

anticipated event **ObserveSucces**
refines **ObserveSucces**
where
@grd1 GoalReached(SysState)=TRUE
end

Events
convergent event Act **refines** Act
any *agent*
where
@grd1 *agent*∈Agents
@grd2 actionAgent(*agent*)≠***noAction***
@grd3 GoalReached(SysSatte)=**FALSE**
@grd4 GoalReachabilityProgression(SysState) ≠ 0
then
@act1 modeAgent(*agent*)≔***nonActive***
@act2 GoalReached, GoalReachabilityProgression:|GoalReached‘(SysState)∈**BOOL**
∧GoalReachabilityProgression‘(SysState)<GoalReachabilityProgression(SysState)
end

The Self-Organisation Pattern(1/3)

- Goal: reason about the ability of the system to self-adapt

$$\text{Adaptivity} \triangleq \square\,(\text{Perturb} \Rightarrow \Diamond \text{SuccessSO})$$

- The proof pattern *SOP*
  - The machine SOP_0

```
variables SysState, Perturb, SuccessSO
Invariants
@inv1 SysState∈STATES
@inv2 SuccessSO∈ STATES → BOOL
@inv3 Perturb∈ STATES → BOOL
```

```
event Perturbation
refines ActEnvironement
   then
     @act1 Perturb(SysState):=TRUE
 end
```

```
Events
anticipated event Act refines Act
   any agent
   where
    @grd1 agent∈Agents
    @grd2 actionAgent(agent)≠noAction
   then
    @act1 modeAgent(agent):=nonActive
    @act2 SuccessSO:|SuccessSO‘(SysState)∈BOOL
end
```

```
anticipated event ObserveSOSuccess
   where
    @grd1 SuccessSO(SysState) =TRUE
  end
```

The Self-Organisation Pattern(2/3)

- The machine SOP_1 refines SOP_0

variables SysState, Perturb, SuccessSO, SOProgression
Invariants
@inv1 SOProgression∈ STATES → ℤ
Variant SOProgression(SysState)

event Perturbation
refines ActEnvironement
then
@act1 Perturb(SysState):=TRUE
end

Events
convergent event ApplySO **refines** Act
any *agent*
where
@grd1 *agent*∈Agents
@grd2 actionAgent(*agent*)≠***noAction***
@grd3 SuccessSO =**FALSE**
@grd4 SOProgression ≠ 0
then
@act1 modeAgent(*agent*):=***nonActive***
@act2 SuccessSO,SOProgression:|
SuccessSO‘(SysState)∈**BOOL**
∧SOProgression‘(SysState)<SOProgression(SysState)
end

anticipated event ObserveSOSuccess
where
@grd1 SuccessSO(SysState) =TRUE
end

The Self-Organisation Pattern(3/3)

- Given the following invariant:

$$\text{Perturb} \wedge \neg \text{SuccessSO} \Rightarrow (\exists\ ag.ag \in \text{Agents} \wedge \text{SOProgression} \neq 0\ )$$

- We need to prove the following theorems:
  - Theorem1: to be proved for every event

$$\exists\ ag.ag \in \text{Agents} \wedge \text{SOProgression} \neq 0 \wedge \neg \text{SuccessSO} \Rightarrow ((\exists\ ag.ag \in \text{Agents} \wedge \text{SOProgression} \neq 0\ ) \vee \text{SuccessSO})$$

  - Theorem2

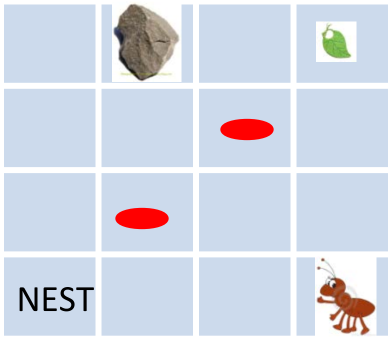
$$\Box \Diamond ((\forall ag.ag \in \text{Agents} \wedge \text{SOProgression} = 0) \vee \text{SuccessSO})$$

➔Every event describing a state where SO is not yet achieved is convergent

➔The machine SOP_1 is deadlock free in a state where SO is not yet achieved

Foraging Ants Case Study: Micro Level Description

- Micro Level
  - Every ant is an agent
  - Property : current location
  - Representations: food, pheromone, obstacles and ants
  - Decision: choose the next location
  - Actions: Move, Drop pheromone, Harvest food, Drop food



➔Agents pattern → Deadlocked free behavior

Foraging ants: Macro Level Properties

- Reach1. The ants are able to bring all the food to the nest

$$\Diamond(\Box(QuantityFood(Nest)=TotalFood(InitDistFood) \wedge \forall\, loc.loc \in Locations \backslash \{Nest\} \Rightarrow (QuantityFood(loc)=0))$$

➔ Global Behavior pattern → Reach1

Foraging ants: Macro Level Properties

- SO1. When a source of food is reached, the ants are able to focus on its exploitation

$$\Box(\forall\, loc.loc \in Locations \setminus \{Nest\} \wedge InitDistFood(loc) \neq 0 \wedge Detected(loc) \Rightarrow \Diamond(QuantityFood(loc)=0))$$

- SO2. When a detected source of food disappears, the ants can continue the environment exploration.

$$\Box((\exists l.l \in Locations \wedge QuantityFood(l)>0 \wedge (\forall\, loc1.loc1 \in Locations \setminus \{Nest\} \wedge EntirelyExploited(loc1) \Rightarrow \Diamond(\exists loc2.loc2 \in Locations \wedge loc2 \neq loc1 \wedge QuantityFood(loc2) \neq 0 \wedge Detected(loc2))$$

➔ Self-Organisation pattern → SO1 and SO2

Conclusion

- The use of the patterns gives an important guidance for self-organizing MAS designers
  - When defining the local behavior of the agents
  - When describing what should be proved for proving global properties

Open Questions

- What about a fully automatic process for applying patterns?
  - Matching is done manually
- The proof patterns do not describe how to prove the global properties
  - How these patterns can be improved in order to make the proofs easier?
  - It is possible to unify simulation and formal verification in one framework in order to reason rigorously about SOMAS?

Thank You For Your Feedback

Thank you For

Linas Laibinis & Elena Troubitsyna