GDT4MAS: a formal model and language to specify and verify multiagent systems

Bruno Mermet, Gaëlle Simon
GREYC – UMR 6072
Université du Havre
Sketch of the presentation

- General presentation
- The GDT4MAS model
- Proofs
- Small case study
- Conclusion and Perspectives
- Holonic extension
General Presentation
GDT4MAS model
Motivations

- Main goal
  - Writing executable and provable specifications
- What about standard MAS models?
  - Lack of required properties for formal specifications
- What about standard Formal Methods?
  - Not adapted to agents developed independently
  - Not « goal oriented »
- Properties studied
  - Safety properties (« nothing bad happens »)
  - Liveness properties (« something good will happen »)
GDT4MAS model
Main concepts

- Theorem proving instead of Model checking
  - May fail, but more general
  - Complexity reduced
  - Can be performed on partial specifications

- Compositional system

- Refinement Principle

- Using experienced systems as much as possible
  - Set theory (like in B, Z or VM)
  - Existing provers (PVS, krt, etc.)
  - Proof schemas are integrated to the method
Adequation of the GDT4MAS model to complex systems

- Specifying numerous entities/agents is easy to perform
- Goal execution may fail
- Using theorem proving rather than model checking reduces the complexity of the proof
- The main part of the proofs can be performed on entities/agents specified independently
The GDT4MAS model
Environment and Agent types

- Environment
  - A set of typed variables
  - An invariant property
  - A set of agents, instances of given agent types

- Agent type
  - A set of internal variables
  - A set of Surface Variables
  - An invariant
  - A behaviour specified by a GDT
Goal Decomposition Tree (1)

- A tree of goals
- A goal is specified by a Satisfaction Condition
- A non-leaf goal is decomposed into subgoals thanks to decomposition operators
- An NNS goal may fail
- A GPF specifies what happens if a goal resolution fails (GPFs are inferred from leaf goals)
Goal Decomposition Tree (2)

- **Leaf goals**
  - Action goals: the goal is achieved by an action of the agent
  - External goals: the goal must be achieved by another agent
  - Unrefined goals

- ** Decomposition operators**
  - And/SeqAnd
  - Or/SeqOr
  - Case
  - Iter
  - SyncSeqAnd/SyncSeqOr
Decomposition
Informal Semantics

- $N \leq\leq N_1 \text{ SeqAnd } N_2$

```java
boolean achieve(Goal N) {
    boolean success, subGoalSuccess;
    subGoalSuccess = achieve(N1);
    if (subGoalSuccess) {
        subGoalSuccess = achieve(N2);
        if (subGoalSuccess) {
            return true;
        }
    }
    return evaluate(sc_N);
}
```
Proofs
Main concepts

- Proof Schemas
  - Formulae integrated to the method
  - Express necessary conditions to guarantee the correctness of the specification
- Reuse
  - Good theorem provers exist in first order-logic
  - Proof schemas must generate predicates, called proof obligations, in order to be able to use any existing first-order logic prover
- Maximization of the proof success rate
  - The simplest the proofs are, the more the prover may succeed
  - => proofs must be as independent as possible
Proving decomposition correctness

\[ x' = 2x + 2 \]
\[ x' = x + 1 \]
\[ x' = 2x \]

SeqAnd

x environment variable

\[ \Rightarrow \]
\[ x_2 = ? \]
Simplified Proof Schema

Theory

- **Notations**
  - $p^i_j : v \Rightarrow v_i, v' \Rightarrow v_j$
  - $(x' = x + 1)^0_1 \Rightarrow x_1 = x_0 + 1$
  - $\text{stab}^i_j : \bigvee (v_i = v_j) \forall v \text{ internal/surface variable}$
  - If $x$ unique internal variable $\text{stab}^0_1 : x_0 = x_1$

- **SeqAnd SPS** ($N \iff N_1 \text{ SeqAnd } N_2$)
  - $\text{sc}^{0 \rightarrow 1}_{N_1} \text{ stab}^{1 \rightarrow 2} \text{ sc}^{2 \rightarrow 3}_{N_2} \rightarrow \text{ sc}^{0 \rightarrow 3}_N$

15 Context is missing!
Simplified Proof Schema

Example

- **x** internal variable
  - \((x' = x + 1)^{0 \rightarrow 1}\) \((x_1 = x_2)\) \((x' = 2x)^{2 \rightarrow 3} \rightarrow (x' = 2x + 2)^{0 \rightarrow 3}\)
  - \((x_1 = x_0 + 1)\) \((x_1 = x_2)\) \((x_3 = 2x_2) \rightarrow (x_3 = 2x_0 + 2)\)

- **x** environment variable
  - \((x' = x + 1)^{0 \rightarrow 1}\) \((x' = 2x)^{2 \rightarrow 3} \rightarrow (x' = 2x + 2)^{0 \rightarrow 3}\)
  - \((x_1 = x_0 + 1)\) \((x_3 = 2x_2) \rightarrow (x_3 = 2x_0 + 2)\)
Other kinds of proofs

- Invariants
  - Only at the action level
- Termination
  - Loop termination proven using a variant
- Actions applicability
  - For action goals, thanks to action preconditions
- Action goals resolution
  - Thanks to precondition and postcondition of actions
- External goal achievement
- MAS level proofs
  - ...
Small case study
Description

- A robot must light several rooms (if the room is already lighted, its goal is achieved).
- If it enters in a room using a door with a cellular eye, the room should be lighted.
- But if it detects the room is still in the dark, it must use the switch.
More Formal aspects

Variables

- **Type**
  - NUM : set of Room numbers

- **Environment variable**
  - S  NUM → Bool : set of room states

- **Internal variable**
  - InRoom  NUM : current room
  - n  NUM : room to lighten
More formal aspects
Satisfaction conditions

- Goal LightedRoom (LR)
  - $s_{LR} = S(n) = true$

- Goal UsingCellularEye (UCE)
  - $s_{UCE} = S'(n) = true \land \text{inRoom}'=n \land n'=n$

- Goal UsingSwitch (US)
  - $s_{US} = S(n) = false \rightarrow S'(n) = true \land n'=n$
More formal aspects

Actions

- Trigger (NNS)
  - Precondition
    - True
  - Postcondition
    - $S'(n) = true \quad inRoom' = n$
  - GPF
    - $S'(n) = false \quad inRoom' = n \quad n' = n$

- Switch (NS)
  - Precondition
    - $InRoom = n$
  - Postcondition
    - $S'(n) = S(n) \quad n' = n$
Proof obligations generated (1)

lumieres THEORY BEGIN
max : nat
roomSet(l : nat) : TYPE = \{n : nat / n ≤ l\}
N, n_3, n_2, n_1, n0, n1 : VAR roomSet(max)
S, S_3, S_2, S_1, S0, S1 : VAR [roomSet(max) → bool]
InRoom, inRoom_3, inRoom_2, inRoom_1, inRoom0, inRoom1 : VAR
roomSet(max)

LR01 : THEOREM
(¬ S_1(n_1)) & (S_1(n_1) = S0(n_1)) &
(inRoom_1 = inRoom0) & (n_1 = n0) & S1(n0) & (inRoom1 = n0) & (n1 = n0)
⇒
S1(n1)

LR02 : THEOREM (¬ S_3(n_3)) & (S_3(n_3) = S_2(n_3)) &
(inRoom_3 = inRoom_2) & (n_3 = n_2) & (¬ S_1(n_2)) & (inRoom_1 = n_2) &
(n_1 = n_2) & (inRoom0 = inRoom_1) & (S0(n_1) = S_1(n_1)) &
(n0 = n_1) & (¬ (S0(n0)) ⇒ S1(n0)) & (n1 = n0)
⇒
S1(n1)
Proof obligations generated (2)

UCE02 : THEOREM \(\neg S_1(n_1)\) \&
\((S_1(n_1) = S0(n_1)) \& (inRoom_1 = inRoom0) \& (n_1 = n0) \& S1(n0) \&
(inRoom1 = n0) \& (n1 = n0)\)
\(\Rightarrow\)
\((S1(n0) \& (inRoom1 = n0) \& (n1 = n0))\)

US01 : THEOREM
\((\neg S_3(n_3)) \& (S_3(n_3) = S_2(n_3)) \&
(inRoom_3 = inRoom_2) \& (n_3 = n_2) \&
(\neg S_1(n_2)) \& (inRoom_1 = n_2) \&
(n_1 = n_2) \& (inRoom = inRoom_1) \& (S(n_1) = S_1(n_1)) \& (n = n_1)\)
\(\Rightarrow\)
\((inRoom = n)\)

US02 : THEOREM \((\neg S_3(n_3)) \&
(S_3(n_3) = S_2(n_3)) \& (inRoom_3 = inRoom_2) \&
(n_3 = n_2) \& (\neg S_1(n_2)) \&
(inRoom_1 = n_2) \& (n_1 = n_2) \&
(inRoom0 = inRoom_1) \& (S0(n_1) = S_1(n_1)) \&
(n0 = n_1) \& (S1(n0) = \neg S0(n0)) \& (n1 = n0)\)
\(\Rightarrow\)
\(((\neg S0(n0) \Rightarrow S1(n_0)) \& (n1 = n0))\)

END lumieres
Proof?

Performed automatically by PVS (with its basic strategy, grind)
Conclusion

• An expressive model
• A compositional model
• A compositional formal verification system
• An operational semantics and a translation process in any imperative language
• A proven model
• A complexity-limited proof process, using existing theorem provers
• http://users.info.unicaen.fr/~bmermet/GDT/
Perspectives

- Increasing expressiveness using holonic agents (see next slides)
- Designing communication protocols
- Developing of an IDE (on-going work)
Holonic extension

• Main concept
  • Parts of an agent behaviour may be implemented by MAS

• New decomposition operators
  • ParAND
    – To solve a goal of the « parent », several subagents must achieve their main goal
  • ParOR
    – To solve a goal of the « parent » agent, several agents start their execution, but the success of one is enough
Holonic agents Principle
The *Robots On Mars* problem

- **2 robots (R1 and R2)**

- **Robot R1**
  - explore Mars (a grid) looking for pieces of garbage
  - When one is found, it brings it to R2, comes back the cell it was, and continues its exploration

- **Robot R2**
  - Does not move
  - Burns pieces of garbage brought by R1 on its cell
RoM New version

- Several robots R1 and R2
- R1 made of several components
- R2 made of several components
New version: R2 is made of:
- An arm throwing wastes picked on its cell to a tank R limited to 10 wastes
- A conveyor which, as soon as 3 wastes are in R, brings them one by one to the incinerator
- New version: R1 brings waste to the nearest robot R2 that is « up »
  - A tracker detects and maintains the coordinates of the nearest « up » Robot R2;
  - A motor moves R1 either to explore the grid, or, if it holds a garbage, to the target chosen by the tracker.
Structure of the MAS
Robot R1
Robot R2
Simplifying usage of PDOs
Design patterns

- Several actuators
- Several achievement goals
- Roles composition
- Interruptible task
- Combination of cognitive and reactive behaviour
- Communication reception
JFSMA'12
Journées Francophones sur les Systèmes Multi-Agents 2012

• XXème édition
• Lieu : Honfleur
• Dates : 24-26 octobre
• Program Committee
  • Chair : Pierre Chevaillier
• Organizing Committee
  • Chair : Bruno Mermet