Model Checking of Robot Gathering

Ha Thi Thu Doan, François Bonnet, and Kazuhiro Ogata

JAIST, Nomi, Japan
Osaka University, Osaka, Japan

OPODIS, December 18, 2018
Outline

• Motivation
• Gathering Problems on Ring
• Algorithm for Robot Gathering
• Formal Specification
• Model Checking
• Counterexamples
• Conclusion
Motivation

Challenges

- **Informal proofs:**
  Difficult to study all cases -> Easy to make mistakes

- **New area of research:**

The approach to the formal verification of these algorithms
Motivation

Challenges

- Informal proofs:
  Difficult to study all cases -> Easy to make mistakes

- New area of research:

Distributed mobile robot algorithms

Prove correctness

Model Checking of Robot Gathering
Computational Model – Ring

- **Robot**
  - *Anonymous*: identical
  - *No sense of direction*

- **Anonymous ring**
  - Neither nodes nor links are labeled

  ![Diagram of anonymous rings]

  *Oblivious*: no memory of the past actions.

Fig 1. These configurations are considered the same
Computational Model – Ring

• Look-Compute-Move Phase Behavior
  
i. Look: Take a snapshot
  
ii. Compute: Compute a move
  
iii. Move (instantaneous): Execute the computed move.

• Asynchronous scheduler
  
  ▪ Phases are asynchronous.
  
  ▪ Pending move
Computational Model – Ring

- Multiple robots can be located at the same node

- *Multiplicity detection* capacity:
  - empty,
  - single robot or
  - more than one (but not the exact number).
Gathering Problem

Robots, initially located at different locations, have to gather at the same location (not determined in advance) and remain in it
An Algorithm for Robot Gathering

Four phases:

- **MULTIPLICITY-CREATION**: creates either *one* or *two* symmetric multiplicities
- **COLLECT**: moves all but *four robots*
- **MULTIPLICITY-CONVERGENCE**: makes the two multiplicities to *merge into one*
- **CONVERGENCE**: allows the remaining single robots to *join the unique multiplicity*

A first look to the algorithm

Two examples of possible executions of the gathering algorithm. *Black nodes* represent multiplicities.
A first look to the algorithm

Two examples of possible executions of the gathering algorithm. *Black nodes* represent multiplicities.
Classifying Configurations

**Procedure**: MULTIPLICITY-CREATION

**Input**: CT, $C = Q(r) = (q_0, q_1, \ldots, q_j)$

1. case $CT = W1$
2. if $C = \overline{(C_j)}$ then
3. move towards $q_0$;
4. case $CT = W2$
5. if GATHER-FOUR-NODES($C$) then
6. REDUCTION ($C$);
Formal Specification

1. **State Expression:**

ops FC FC- nil : -> Pending [ctor] .

op <_,_> : Int Pending -> Node [ctor] .

op __ : Seq Seq -> Seq [ctor assoc] .

\[
\begin{align*}
&\{< 1, \text{nil} > < 0, \text{nil} > < 5, \text{nil} > < -1, \text{nil} > < 1, \text{nil} > < 3, \text{nil} > \} \\
&(a)
\end{align*}
\]

\[
\begin{align*}
&\{< 1, \text{nil} > < 0, \text{nil} > < 5, \text{FC} > < -1, \text{nil} > < 1, \text{nil} > < 3, \text{nil} > \} \\
&(b)
\end{align*}
\]

Maude specification language is used!
Formal Specification

2. State Transitions:
Formal Specification

\[ rl \text{ [FC2-pending]} : \]
\[ \{S_1 < I_1, P > < I_2, FC > S_2\} \Rightarrow \{S_1 < I_1 + 1, P > < I_2 - 1, \text{nil} > S_2\} \]

\[ \{< 1, \text{nil} > < 0, \text{nil} > < 5, FC > <-1, \text{nil} > < 1, \text{nil} > < 3, \text{nil} >\} \]
Formal Specification

\( \text{rl [FC2-pending]} : \{S_1 < I_1, P >= I_2, FC > S_2\} \Rightarrow \{S_1 < I_1 + 1, P >= I_2 - 1, \text{nil} > S_2\} \).

\{< 1, \text{nil} > < 0, \text{nil} > < 5, FC > < -1, \text{nil} > < 1, \text{nil} > < 3, \text{nil} >\} \rightarrow \{< 1, \text{nil} > < 1, \text{nil} > < 4, FC > < -1, \text{nil} > < 1, \text{nil} > < 3, \text{nil} >\} \)
Formal Specification

mod W1-MOVE is
  pr CONFIG-CALCULATION .

  crl [w1-fo] : {S1 < I, nil > S2} => {S1 < I, FC > S2}
    if checkEqual({< I, nil > S2 S1}) and w1({S1 < I, nil > S2}) .

  ...
  endm

Mod MULTIPLICITY-CREATION is
  pr W1-MOVE .
  ...
  endm
Formal Specification

mod GATHERING is
  pr MULTIPLICITY-CREATION .
  pr COLLECT .
  pr MULTIPLICITY-CONVERGENCE .
  pr CONVERGENCE .
endm
Model Checking

• Lemmas: Lemma 5, 6, and 7

**Lemma 5:** Phase multiplicity–creation terminates with either one or two symmetric multiplicities after a finite number of moves.

**Lemma 6:** Phase collect terminates after a finite number of moves by reaching a symmetric configuration with six nodes occupied by two multiplicities, two guards and two single robots adjacent to the multiplicities.
Model Checking

• We took *exactly* the pseudo-code given in the paper

• Informal description of the algorithm (in plain sentence) may be correct, but it is difficult to formalize such informal description!
LTL formulas

• Specifying Lemmas as Linear Temporal Logic (LTL) formulas

\[ C \models \text{endOfColl} = \text{checkOfColl}(C). \]

\[ C \models \text{coll} = \text{checkColl}(C). \]

\[ \text{lemma6} = \square(\text{endOfColl} \rightarrow \text{coll}) \land \lozenge \text{endOfColl}. \]
The model checking: *counterexamples are found.*
Counterexamples

• **Design Errors**

{(a), (b), (c), (d)}

\{<-1,nil,<2,nil,<1,nil,<2,nil,<0,nil,<3,nil,<1,nil,<4,nil>}

\{<-1,nil,<2,nil,<1,nil,<2,nil,<0,FC-,<3,FC-,<1,nil,<4,nil>}

\{<-1,nil,<2,nil,<1,nil,<1,nil,<0,nil,<4,nil,<1,nil,<4,nil>}

20
Counterexamples

• Design Errors

\{< 2,\text{nil} > < 1,\text{nil} > < 0,\text{nil} > < 2,\text{nil} > < 1,\text{nil} > < 2,\text{nil} > < 2,\text{nil} >\}.
\{< 2,\text{nil} > < 1,\text{FC} > < 0,\text{nil} > < 2,\text{nil} > < 1,\text{nil} > < 2,\text{FC-} > < 2,\text{nil} >\}
\{< 3,\text{nil} > < 0,\text{nil} > < 0,\text{nil} > < 2,\text{nil} > < 0,\text{nil} > < 3,\text{nil} > < 2,\text{nil} >\}
Counterexamples

- Omission of Special Cases

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>{&lt; 2, nil }</td>
<td>{&lt; 2, nil }</td>
</tr>
<tr>
<td>{&lt; -1, nil }</td>
<td>{&lt; -1, nil }</td>
</tr>
<tr>
<td>{&lt; 5, nil }</td>
<td>{&lt; 5, nil }</td>
</tr>
<tr>
<td>{&lt; -1, nil }</td>
<td>{&lt; -1, nil }</td>
</tr>
<tr>
<td>{&lt; 2, nil }</td>
<td>{&lt; 2, nil }</td>
</tr>
<tr>
<td>{&lt; 1, nil }</td>
<td>{&lt; 1, nil }</td>
</tr>
<tr>
<td>{&lt; 0, nil }</td>
<td>{&lt; 0, nil }</td>
</tr>
<tr>
<td>{&lt; 1, nil }</td>
<td>{&lt; 1, nil }</td>
</tr>
<tr>
<td>{&lt; 1, nil }</td>
<td>{&lt; 1, nil }</td>
</tr>
</tbody>
</table>

(a) (b) (c) (d) (e) (f)
Counterexamples

• Some minor errors

17 case $CT = W6$
18 $C' := (q_0 + 1, q_1 - 1, \ldots, q_j)$;
19 if $C' = \overline{C'}$ and $q_0$ is odd then move towards $q_0$;
20 ;
21 else
22 $C'' := (q_0, \ldots, q_{j-1} - 1, q_j + 1)$;
23 if $C'' = \overline{(C''_j)}$ and $q_j$ is odd then
24 move towards $q_j$;
25 else
Conclusion

The lemmas are **not satisfied!**
Conclusion

*future directions:*

- Fixing the mistakes that we found
- Come up with a way to make it concisely specify mobile robot algorithms
- Modeling other algorithms