Engineering Safety in Swarm Robotics

Giovanni Beltrame$^{1,2}$, Ettore Merlo$^1$, Jacopo Panerati$^{1,2}$, and Carlo Pinciroli$^3$
Swarm Engineering and its Challenges

Can we tackle **complex problems**, with **very large** collections of **simple agents**?
Swarm Engineering and its Challenges

Can we tackle complex problems, with very large collections of simple agents?

Swarms carry:

1. the joint promises of distributed, large-scale, and autonomous sys.

image from Newsweek (5/15/14)
Swarm Engineering and its Challenges

Can we tackle **complex problems**, with **very large** collections of **simple agents**?

Swarms carry:

1. the joint **promises** of distributed, large-scale, and autonomous sys.
2. the **challenges** of distributed, large-scale, and autonomous sys.

image from Newsweek (5/15/14)
Striving for Safer Software

Key for: 1. commercial applications

Image from www.dji.com/flysafe/geo-map
Striving for Safer Software

*autonomous driving

Key for: 1. commercial applications

Image from www.dji.com/flysafe/geo-map
Striving for Safer Software

Key for: 1. commercial applications
2. life-critical applications

*autonomous driving

image from www.dji.com/flysafe/geo-map
Striving for Safer Software

Key for: 1. commercial applications
2. life-critical applications

*autonomous driving
*disaster-response
MANETs

image from www.dji.com/flysafe/geo-map
Striving for Safer Software

Key for:
1. commercial applications
2. life-critical applications

Public safety enforced by governments’ policies and regulations (OSHA an FAA in the US)

*autonomous driving
*disaster-response
MANETs

image from www.dji.com/flysafe/geo-map
Striving for Safer Software

Key for: 1. commercial applications
2. life-critical applications

Public safety enforced by governments’ policies and regulations (OSHA an FAA in the US)

For the next decade, we foresee the need for tools to certificate robot swarms and their software

*autonomous driving
*disaster-response
MANETs
The Buzz Language

- Domain-specific
The Buzz Language

- Domain-specific
- **Concise and composable** scripting for heterogeneous robots

Example Buzz snippets

```python
1 if (safety.has_privilege(BROADCAST_ALL)) {
2     # the current robot has the
3     # BROADCAST_ALL privilege
4     # send message to all robots
5     neighbors.broadcast("key", value)
6 }

1 include "flight.bzz"
2 if (!safety.has_privilege(TAKE_OFF)) {
3     # stops the execution
4     # if the current robot
5     # doesn't have the TAKE_OFF
6     # privilege
7 }
```
The Buzz Language

- **Domain-specific**
- **Concise and composable** scripting for heterogeneous robots
- **Portability to multiple simulation and hardware platforms** (ARGoS, Gazebo, ROS, kilobots, Khepera IV, Zooids, etc.)

```c
1  if (safety.has_privilege(BROADCAST_ALL)) {
2     # the current robot has the
3     # BROADCAST_ALL privilege
4     # send message to all robots
5     neighbors.broadcast("key", value)
6 }

1  include "flight.bzz"
2  if (!safety.has_privilege(TAKE_OFF)) {
3     # stops the execution
4     # if the current robot
5     # doesn’t have the TAKE_OFF
6     # privilege
7 }
```
Pattern Traversal Flow Analysis (PTFA)

- PTFA extracts **privilege satisfaction** models from the source.
Pattern Traversal Flow Analysis (PTFA)

- PTFA extracts **privilege satisfaction** models from the source
- Exploits a **Buzz-generated Control Flow Graph**
Pattern Traversal Flow Analysis (PTFA)

- PTFA extracts **privilege satisfaction** models from the source
- Exploits a **Buzz-generated Control Flow Graph**
- **Linear time-complexity** with respect to the graph size and the # of privileges
Our Vision for Safer Swarms

- Integrate **swarm-level assertions** and safety keywords/functions within the Buzz language

<table>
<thead>
<tr>
<th>Domain-specific Constructs and Functions</th>
<th>Swarm-shared data-structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>stigmergy.{create(), put(), get()}</td>
<td></td>
</tr>
<tr>
<td>swarm.{create(), join(), ...}</td>
<td>Swarm management primitives</td>
</tr>
<tr>
<td>neighbors.{broadcast(), filter(), ...}</td>
<td>Local robot-to-robot communication</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Security-specific Keywords and Functions</th>
<th>Authorization verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>safety.{has_privilege(), ...}</td>
<td></td>
</tr>
<tr>
<td>behavior.{has_converged(), ...}</td>
<td>Swarm consensus primitives</td>
</tr>
<tr>
<td>preserve(CONNECTIVITY, ...)</td>
<td>Swarm asserts</td>
</tr>
</tbody>
</table>
Our Vision for Safer Swarms

● Integrate **swarm-level assertions** and safety keywords/functions **within the Buzz language**
● More robust and fine-grained management through the Buzz compiler and Virtual Machine

<table>
<thead>
<tr>
<th>Domain-specific Constructs and Functions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stigmergy.{create(), put(), get()}</td>
<td>Swarm-shared data-structure</td>
</tr>
<tr>
<td>swarm.{create(), join(), ...}</td>
<td>Swarm management primitives</td>
</tr>
<tr>
<td>neighbors.{broadcast(), filter(), ...}</td>
<td>Local robot-to-robot communication</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Security-specific Keywords and Functions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>safety.{has_privilege(), ...}</td>
<td>Authorization verification</td>
</tr>
<tr>
<td>behavior.{has_converged(), ...}</td>
<td>Swarm consensus primitives</td>
</tr>
<tr>
<td>preserve(CONNECTIVITY, ...)</td>
<td>Swarm asserts</td>
</tr>
</tbody>
</table>
Our Vision for Safer Swarms

- Integrate **swarm-level assertions** and safety keywords/functions **within the Buzz language**
- More robust and fine-grained management through the Buzz compiler and Virtual Machine
- Analysis tool to **automate safety checks** on large scale projects (>100k-1M LOC)

### Domain-specific Constructs and Functions

<table>
<thead>
<tr>
<th>Construct</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stigmergy.{create(), put(), get()}</td>
<td>Swarm-shared data-structure</td>
</tr>
<tr>
<td>swarm.{create(), join(), ...}</td>
<td>Swarm management primitives</td>
</tr>
<tr>
<td>neighbors.{broadcast(), filter(), ...}</td>
<td>Local robot-to-robot communication</td>
</tr>
</tbody>
</table>

### Security-specific Keywords and Functions

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>safety.{has_privilege(), ...}</td>
<td>Authorization verification</td>
</tr>
<tr>
<td>behavior.{has_converged(), ...}</td>
<td>Swarm consensus primitives</td>
</tr>
<tr>
<td>preserve(CONNECTIVITY, ...)</td>
<td>Swarm asserts</td>
</tr>
</tbody>
</table>
References


Supplementary Material (i)

FAQs:

● In what robotic swarms differ from traditional multi-agent systems?
  ○ Large numbers of individuals
  ○ Purely decentralized control
  ○ No predefined roles
  ○ Cooperation

● Why go through the compiler and not just use new libraries?
  ○ To allow for compile-time (as opposed to run-time) privilege checks, when possible
Supplementary Material (ii)

- Buzz tutorials: http://the.swarming.buzz/ICRA2017/
- Buzz GitHub: https://github.com/MISTLab/Buzz
- Buzz Wiki: http://the.swarming.buzz/wiki/doku.php
Supplementary Material (iii)

Swarms as Programmable Collections

Buzz Virtual Machine
Supplementary Material (iv_a)

PTFA: Nodes rewrite

\[ v_X \rightarrow q_{x,0,0} \rightarrow q_{x,0,1} \rightarrow q_{x,1,0} \rightarrow q_{x,1,1} \]
Supplementary Material (iv_b)

PTFA: Generic edge rewrite

PTFA: Generic edge rewrite
Supplementary Material (iv_c)

PTFA: Positive authorization edge rewrite

\[ \nu_X \quad \xrightarrow{\text{\textbullet}} \quad \nu_Y \]

\[ q_{X,0,0} \quad q_{X,0,1} \quad q_{X,1,0} \quad q_{X,1,1} \]

\[ q_{Y,0,0} \quad q_{Y,0,1} \quad q_{Y,1,0} \quad q_{Y,1,1} \]
Supplementary Material (iv_d)

PTFA: Negative authorization edge rewrite

\[ v_X \quad \Rightarrow \quad \begin{array}{cccc}
q_{x,0,0} & q_{x,0,1} & q_{x,1,0} & q_{x,1,1} \\
q_{y,0,0} & q_{y,0,1} & q_{y,1,0} & q_{y,1,1}
\end{array} \]
Supplementary Material (iv_e)

PTFA: Inter-procedural call rewrite

Supplementary
Supplementary Material (iv\textsubscript{f})

PTFA: Inter-procedural return rewrite
Contacts

Prof. Giovanni Beltrame  giovanni.beltrame@polymtl.ca
Prof. Ettore Merlo  ettore.merlo@polymtl.ca
Jacopo Panerati  jacopo.panerati@polymtl.ca
Prof. Carlo Pinciroli  cpinciroli@wpi.edu