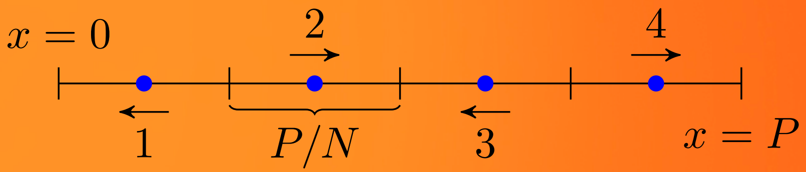


When Human Intuition Fails: Using Formal Methods to Find an Error in the "Proof" of a Multi-Agent Protocol

Midwest Verification Days

September 29, 2018

Jen Davis, Derek Kingston, Laura Humphrey

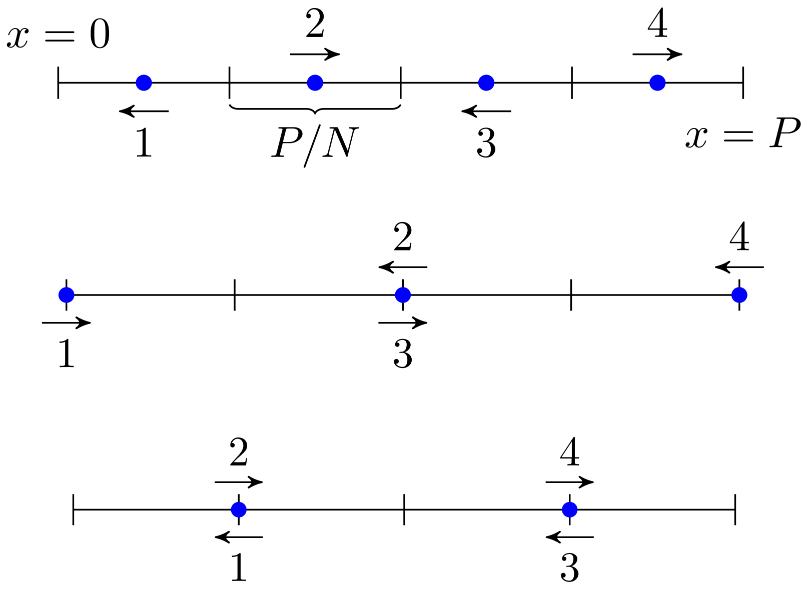
Approved for Public Release. Case Number: 88ABW-2018-4275

Building trust every day

Linear perimeter surveillance

- Want to divide surveillance of a perimeter across UAVs under communication constraints
- Assumptions
  - UAVs only communicate at short range
  - UAVs can leave and join the team
  - UAVs travel at the same constant speed
  - Perimeter can change over time
- <u>Goal</u> – A decentralized protocol that converges in finite time
- <u>Solution</u> – Decentralized Perimeter Surveillance System (DPSS)

DPSS convergence



Definition of convergence

N UAVs on a perimeter of length P

UAVs oscillate between two sets of locations synchronously:

1) UAV $i \in 1 \ldots N$ is located at $\lfloor i + \frac{1}{2}(-1)^i \rfloor P/N$
2) UAV $i \in 1 \ldots N$ is located at $\lfloor i - \frac{1}{2}(-1)^i \rfloor P/N$

We call this the *optimal configuration*.

DPSS overview

- Each UAV i stores the following "coordination variables":

  N_{Ri} – Number of UAVs to its right

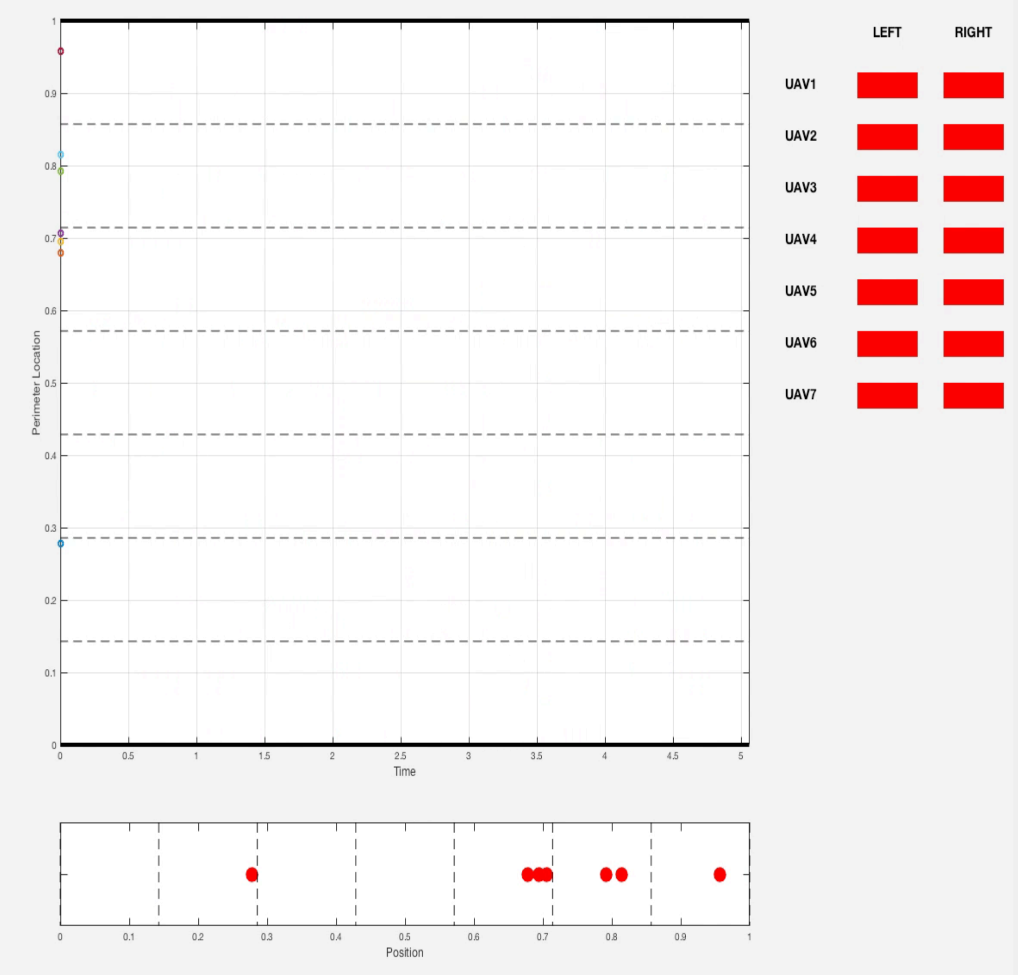
  N_{Li} – Number of UAVs to its left

  P_{Ri} – Amount of perimeter to its right

  P_{Li} – Amount of perimeter to its left

- UAVs exchange information when they meet or are "co-located"
- When UAVs meet, they estimate their shared boundary location, "escort" each other there, then break apart
- UAVs can only ever change direction at the start of an escort, the end of an escort, or at a perimeter boundary

Video of protocol



Approved for Public Release. Case Number: 88ABW-2018-4275

DPSS Protocol

Algorithm B

if agent i (left) rendezvous with neighbor j (right) **then**
  Update perimeter length and team size:
    $P_{R_i} = P_{R_j}$
    $N_{R_i} = N_{R_j} + 1$
  Calculate team size $N = N_{R_i} + N_{L_i} + 1$.
  Calculate perimeter length $P = P_{R_i} + P_{L_i}$.
  Calculate relative index $n = N_{L_i} + 1$.
  Calculate segment endpoints:
    $\mathcal{S}_i = \{\lfloor n - \frac{1}{2}(-1)^n \rfloor P/N, \lfloor n + \frac{1}{2}(-1)^n \rfloor P/N\}$.
  Communicate $\mathcal{S}_i$ to neighbor j and receive $\mathcal{S}_j$.
  Calculate shared border position $p_{i,j} = \mathcal{S}_i \cap \mathcal{S}_j$.
  Travel with neighbor j to shared border $p_{i,j}$.
  Set direction to monitor own segment.
else if reached left perimeter endpoint **then**
  Reset perimeter length to the left $P_{L_i} = 0$.
  Reset team size to the left $N_{L_i} = 0$.
  Reverse direction.
else if reached right perimeter endpoint **then**
  Reset perimeter length to the right $P_{R_i} = 0$.
  Reset team size to the right $N_{R_i} = 0$.
  Reverse direction.
else
  Continue in current direction keeping track of traversed perimeter length.
end if

Algorithm A

if UAV i rendezvous with neighbor j **then**
  Calculate team size $N = N_{R_i} + N_{L_i} + 1$.
  Calculate perimeter length $P = P_{R_i} + P_{L_i}$.
  Calculate UAV i's relative index $n = N_{L_i} + 1$.
  Calculate UAV i's segment endpoints:
    $\mathcal{S}_i = \{\lfloor n - \frac{1}{2}(-1)^n \rfloor P/N, \lfloor n + \frac{1}{2}(-1)^n \rfloor P/N\}$.
  Communicate $\mathcal{S}_i$ to neighbor j and receive $\mathcal{S}_j$.
  Calculate shared border position $p_{i,j} = \mathcal{S}_i \cap \mathcal{S}_j$.
  Travel with neighbor j to shared border position $p_{i,j}$.
  Set direction to monitor own segment.
else if reached perimeter endpoint **then**
  Reverse direction.
else
  Continue in current direction.
end if

- Algorithm B – UAVs do not have correct coordination variables
- Algorithm A – UAVs have correct coordination variables

DPSS proof outline

P – Perimeter length

V – UAV speed

T=P/V – Time for one UAV to travel whole perimeter

- Lemma 1 - Algorithm A converges in 2T
  (UAVs start with correct coordination variables)

- Lemma 2 - Algorithm B achieves correct coordination variables in 3T

- Theorem : Algorithm B converges in 5T

- Proof - Algorithm B converts to Algorithm A once the UAVs have correct coordination variables, so by Lemma 2 and Lemma 1, Algorithm B converges in 3T + 2T = 5T

DPSS "proof" of Lemma 2

- UAVs will learn correct coordination variables.
  - Since UAVs only turn around at perimeter endpoints or when they meet their neighbors:
    - UAV 1 will discover left perimeter in finite time either before or after meeting UAV 2, obtaining correct "left" coordination variables
    - UAV 2 will later meet UAV 1 again, obtaining correct "left" variables

      …
    - UAV N will meet UAV N-1, obtaining correct "left" variables
    - Similar argument holds for "right" variables
- Worst case occurs when all UAVs are stacked on the left or right
- In that case, the correct coordination variables are achieved in 3T

DPSS in AGREE

- Modeled protocol in Assume Guarantee Reasoning Environment (AGREE)
  - Annex to the Architecture Analysis & Design Language (AADL)
  - Leverages k-induction model checking and SMT solvers
- AGREE analyzes architectures that have a top-level system and lower-level components, each having an assume-guarantee contract with assumptions on inputs and guarantees on outputs
- Taking system-level assumptions as true, AGREE verifies that
  - Component assumptions hold given the system-level assumptions
  - System-level guarantees hold given the component guarantees
- System-level AGREE model for DPSS consists of *N* instantiations of a component-level UAV model

AGREE system guarantees

- Lemma 2 – Algorithm B achieves correct coordination variables in 3T

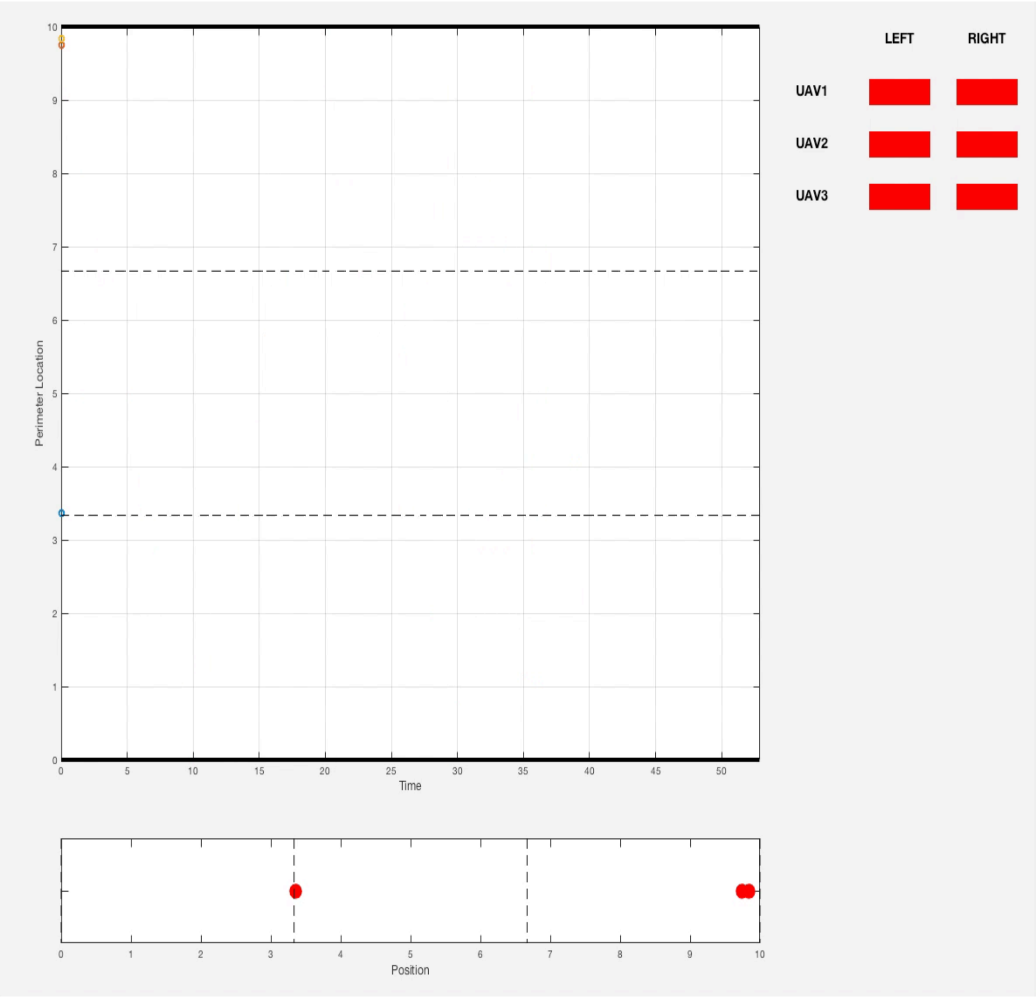
```
lemma “(Invalid) Time to correct coordination variables is < 3T":
    (correct_coordination_variables and not
    (pre(correct_coordination_variables))) =>
                  (time < 3.0*T);
```

- Theorem : Algorithm B converges in 5T

```
lemma “Time to optimal configuration is less than 5T":
    (optimal and not (pre(optimal))) => (time < 5.0*T);
```


Approved for Public Release. Case Number: 88ABW-2018-4275

Video of counterexample



Rockwell Collins

AGREE system guarantees—Revised!

- Lemma 2 (for 3 vehicles) – Algorithm B achieves correct coordination variables in ~~*3T*~~ (3 + ¼)T

```
lemma "Time to correct coordination variables is < (3 + 1/4)T":
    (correct_coordination_variables and not
    (pre(correct_coordination_variables))) =>
                  (time < (3.0 + 1.0/4.0)*T);
```

- Theorem (for 3 vehicles): Algorithm B converges in ~~5T~~ 4T

```
lemma “Time to optimal configuration is less than 4T":
    (optimal and not (pre(optimal))) => (time < 4.0*T);
```


Approved for Public Release. Case Number: 88ABW-2018-4275

Summary

- By formally modeling a decentralized multi-UAV surveillance protocol, were we able to
  - Find an error in the manual proof
  - Potentially show that the overall convergence bound is tighter than the originally claimed upper bound
- However, we were only able to prove this for 3 UAVs
  - 20 hours on a machine with 256 GB RAM and 80 cores
- Next: use a theorem prover like ACL2, PVS, or Coq to prove the convergence bound for an arbitrary number of UAVS

Approved for Public Release. Case Number: 88ABW-2018-4275

References

- J. A. Davis, D. B. Kingston, L. R. Humphrey, “When Human Intuition Fails: Using Formal Methods to Find an Error in the ‘Proof’ of a Multi-Agent Protocol”
  - Preprint available at http://loonwerks.com/publications/davis2018.html
  - Submitted to IEEE for possible publication
- D. B. Kingston, R. W. Beard, and R. S. Holt, “Decentralized perimeter surveillance using a team of UAVs,” IEEE Transactions on Robotics, vol. 24, no. 6, pp. 1394–1404, 2008.
- OpenUxAS GitHub repository, architecture branch. [Online]. Available: https://github.com/afrl-rq/OpenUxAS/tree/architecture
  - See the AADL_sandbox_projects/DPSS-3-AlgB-for-paper folder

Approved for Public Release. Case Number: 88ABW-2018-4275