Formally Proving and Enhancing a Self-Stabilising Distributed Algorithm

Camille Coti ♣, Laure Petrucci ♣, Charles Lakos ♠

♣ LIPN, CNRS UMR 7030, SPC, Université Paris 13
♠ Department of Computer Science, University of Adelaide

June 20th, 2016
Roadmap

1. Introduction

2. The algorithm

3. Formal modelling and analysis
   - The Coloured Petri Net model
   - Formal analysis of the algorithm properties

4. Improving the model to improve the algorithm

5. Conclusion
Introduction

Which algorithm, why it is useful self-stabilisation / lengthy cumbersome proofs $\rightarrow$ formal modelling
The algorithm

P0

P1
  P3
  P7
  P6

P2
  P4
  P8
  P9

P5
The algorithm

Formally Proving and Enhancing a Self-Stabilising Distributed Algorithm
The algorithm

- P0
- P1
- P2
- P3
- P4
- P5
- P6
- P7
- P8
- P9

Formally Proving and Enhancing a Self-Stabilising Distributed Algorithm
The algorithm
The algorithm

P0

P1

P2

P3

P4

P5

P6

P7

P8

P9

AC

BC
The algorithm

Formally Proving and Enhancing a Self-Stabilising Distributed Algorithm
The algorithm

P0

P1

P2

P3

P4

P5

P6

P7

P8

P9
1 Introduction

2 The algorithm

3 Formal modelling and analysis
   - The Coloured Petri Net model
   - Formal analysis of the algorithm properties

4 Improving the model to improve the algorithm

5 Conclusion
Conclusion

C’est un vieux truc?

Three protocols for fault-tolerant QR factorization of tall-and-skinny matrices
  • Cornerstone for general QR factorization
  • Three recovery algorithms, one for each semantics

Algorithm for FT update of the trailing matrix
  • Fault-tolerant QR for general matrices ($R$)

**Scalable FT protocol based on scalable algorithms**

Makes use of new features provided by the MPI-3 standard
  • FT API now provided by MPI-3
  • *User-Level Failure Mitigation*

Next step:
  • Apply this to LU, Cholesky (the other *amigos*)
  • Reconstruction of the Householder vectors ($Q$)
  • Full performance analysis


