Distributed runtime verification for CPSs

Giorgio Audrito[*], Roberto Casadei[†], Ferruccio Damiani[*‡], Volker Stolz[‡], Gianluca Torta[*], Mirko Viroli[†]

[*] Department of Computer Science
University of Turin, Italy

[†] Department of Computer Science and Engineering
University of Bologna, Cesena, Italy

[‡] Department of Computer science, Electrical engineering and Mathematical sciences
Western Norway University of Applied Sciences (HVL), Bergen, Norway

5th Italian Workshop on Embedded Systems (IWES 2020), February 8-9, 2021

Ferruccio Damiani http://www.di.unito.it/~damiani/

- Associate professor at **Department of Computer Science - University of Turin**
- *Adjunct associate professor (10%) at* ***Department of Computer science, Electrical engineering and Mathematical sciences - Western Norway University of Applied Sciences (HLV)****, Bergen, Norway* https://www.hvl.no/en/research/group/software-engineering/
- Member of the Board of European Association for Programming Languages and Systems (EAPLS) https://eapls.org/

Ferruccio Damiani http://www.di.unito.it/~damiani/

- Associate professor at **Department of Computer Science - University of Turin**
- *Adjunct associate professor (10%) at **Department of Computer science, Electrical engineering and Mathematical sciences - Western Norway University of Applied Sciences (HLV)**, Bergen, Norway* https://www.hvl.no/en/research/group/software-engineering/
- Member of the Board of European Association for Programming Languages and Systems (EAPLS) https://eapls.org/

System Modelling, Verification and Reuse research group https://movere.di.unito.it/

Rigorous Software Engineering and DSLs for

- Variability modeling and SPLs
- Distributed systems, IoT and CPSs
- Self-organisation, collective intelligence, edge AI
- Edge/Fog/Cloud Computing
- Quantum Computing

Ferruccio Damiani http://www.di.unito.it/~damiani/

- Associate professor at **Department of Computer Science - University of Turin**
- *Adjunct associate professor (10%) at* ***Department of Computer science, Electrical engineering and Mathematical sciences - Western Norway University of Applied Sciences (HLV)****, Bergen, Norway* https://www.hvl.no/en/research/group/software-engineering/
- Member of the Board of European Association for Programming Languages and Systems (EAPLS) https://eapls.org/

System Modelling, Verification and Reuse research group https://movere.di.unito.it/

Rigorous Software Engineering and DSLs for

- Variability modeling and SPLs
- Distributed systems, IoT and CPSs
- Self-organisation, collective intelligence, edge AI
- Edge/Fog/Cloud Computing
- Quantum Computing

People:

- 8 members (1 PA, 2 RU, 1 RTDA, 1 TDR, 3 PhD)
- 8 "main collaborators" @UNITO (3 PO, 5 PA)

Ferruccio Damiani http://www.di.unito.it/~damiani/

- Associate professor at **Department of Computer Science - University of Turin**
- *Adjunct associate professor (10%) at* ***Department of Computer science, Electrical engineering and Mathematical sciences - Western Norway University of Applied Sciences (HLV)****, Bergen, Norway* https://www.hvl.no/en/research/group/software-engineering/
- Member of the Board of European Association for Programming Languages and Systems (EAPLS) https://eapls.org/

System Modelling, Verification and Reuse research group https://movere.di.unito.it/

Rigorous Software Engineering and DSLs for

- Variability modeling and SPLs
- Distributed systems, IoT and CPSs
- Self-organisation, collective intelligence, edge AI
- Edge/Fog/Cloud Computing
- Quantum Computing

People:

- 8 members (1 PA, 2 RU, 1 RTDA, 1 TDR, 3 PhD)
- 8 "main collaborators" @UNITO (3 PO, 5 PA)

(Main) industrial collaborations:

- REPLY, Synesthesia, FPT, IVECO, Magneti Marelli
- Raytheon BBN Technologies (USA)

Ferruccio Damiani http://www.di.unito.it/~damiani/

- Associate professor at **Department of Computer Science - University of Turin**
- *Adjunct associate professor (10%) at* ***Department of Computer science, Electrical engineering and Mathematical sciences - Western Norway University of Applied Sciences (HLV)****, Bergen, Norway* https://www.hvl.no/en/research/group/software-engineering/
- Member of the Board of European Association for Programming Languages and Systems (EAPLS) https://eapls.org/

System Modelling, Verification and Reuse research group https://movere.di.unito.it/

Rigorous Software Engineering and DSLs for

- Variability modeling and SPLs
- Distributed systems, IoT and CPSs
- Self-organisation, collective intelligence, edge AI
- Edge/Fog/Cloud Computing
- Quantum Computing

People:

- 8 members (1 PA, 2 RU, 1 RTDA, 1 TDR, 3 PhD)
- 8 "main collaborators" @UNITO (3 PO, 5 PA)

(Main) industrial collaborations:

- REPLY, Synesthesia, FPT, IVECO, Magneti Marelli
- Raytheon BBN Technologies (USA)

(Some) research projects/networks:

- EC H2020 RIA HyVar: Scalable Hybrid Variability for Distributed Evolving Software Systems (2015-2018) www.hyvar-project.eu/
- EC COST Action IC1402 ARVI: Runtime Verification beyond Monitoring (2014-2018) https://www.cost-arvi.eu/
- POR FESR 2007-2013 PIE_VERDE: Piattaforma Ibridi Elettrici. Veicoli E Reti di Distribuzione Ecosostenibili (2013-2015)
- *SAB of EC FP7 IP HATS: Highly Adaptable and Trustworthy Software using Formal Models (2009-2013) www.hats-project.eu/*
- EC COST Action IC0701 FoVeOOS: Formal Verification of Object-Oriented Software (2008-2012) www.cost-ic0701.org/

Ferruccio Damiani http://www.di.unito.it/~damiani/

- Associate professor at **Department of Computer Science - University of Turin**
- *Adjunct associate professor (10%) at* ***Department of Computer science, Electrical engineering and Mathematical sciences - Western Norway University of Applied Sciences (HLV)****, Bergen, Norway* https://www.hvl.no/en/research/group/software-engineering/
- Member of the Board of European Association for Programming Languages and Systems (EAPLS) https://eapls.org/

System Modelling, Verification and Reuse research group https://movere.di.unito.it/

Rigorous Software Engineering and DSLs for

- Variability modeling and SPLs
- Distributed systems, IoT and CPSs
- Self-organisation, collective intelligence, edge AI
- Edge/Fog/Cloud Computing
- Quantum Computing

People:

- 8 members (1 PA, 2 RU, 1 RTDA, 1 TDR, 3 PhD)
- 8 "main collaborators" @UNITO (3 PO, 5 PA)

(Main) industrial collaborations:

- REPLY, Synesthesia, FPT, IVECO, Magneti Marelli
- Raytheon BBN Technologies (USA)

(Some) research projects/networks:

- EC H2020 RIA HyVar: Scalable Hybrid Variability for Distributed Evolving Software Systems (2015-2018) www.hyvar-project.eu/
- EC COST Action IC1402 ARVI: Runtime Verification beyond Monitoring (2014-2018) https://www.cost-arvi.eu/
- POR FESR 2007-2013 PIE_VERDE: Piattaforma Ibridi Elettrici. Veicoli E Reti di Distribuzione Ecosostenibili (2013-2015)
- *SAB of EC FP7 IP HATS: Highly Adaptable and Trustworthy Software using Formal Models (2009-2013) www.hats-project.eu/*
- EC COST Action IC0701 FoVeOOS: Formal Verification of Object-Oriented Software (2008-2012) www.cost-ic0701.org/

Department of Computer Science - University of Turin http://www.di.unito.it/

Research infrastructures

- Interdepartmental Competence Centre for Scientific Computing (C3S) https://c3s.unito.it/
- The Turin's High-Performance Centre for Artificial Intelligence (HPC4AI) https://hpc4ai.unito.it/ (POR-FESR 2014-2020)
- Planned "IoT, Robotics and 5G lab" with the Deparment Agricultural, Forest and Food Sciences **(SSD: AGR/09 - meccanica agraria)**

Ferruccio Damiani http://www.di.unito.it/~damiani/

- Associate professor at **Department of Computer Science - University of Turin**
- *Adjunct associate professor (10%) at **Department of Computer science, Electrical engineering and Mathematical sciences - Western Norway University of Applied Sciences (HLV)**, Bergen, Norway* https://www.hvl.no/en/research/group/software-engineering/
- Member of the Board of European Association for Programming Languages and Systems (EAPLS) https://eapls.org/

System Modelling, Verification and Reuse research group https://movere.di.unito.it/

Rigorous Software Engineering and DSLs for

- Variability modeling and SPLs
- Distributed systems, IoT and CPSs
- Self-organisation, collective intelligence, edge AI
- Edge/Fog/Cloud Computing
- Quantum Computing

People:

- 8 members (1 PA, 2 RU, 1 RTDA, 1 TDR, 3 PhD)
- 8 "main collaborators" @UNITO (3 PO, 5 PA)

(Main) industrial collaborations:

- REPLY, Synesthesia, FPT, IVECO, Magneti Marelli
- Raytheon BBN Technologies (USA)

(Some) research projects/networks:

- EC H2020 RIA HyVar: Scalable Hybrid Variability for Distributed Evolving Software Systems (2015-2018) www.hyvar-project.eu/
- EC COST Action IC1402 ARVI: Runtime Verification beyond Monitoring (2014-2018) https://www.cost-arvi.eu/
- POR FESR 2007-2013 PIE_VERDE: Piattaforma Ibridi Elettrici. Veicoli E Reti di Distribuzione Ecosostenibili (2013-2015)
- *SAB of EC FP7 IP HATS: Highly Adaptable and Trustworthy Software using Formal Models (2009-2013) www.hats-project.eu/*
- EC COST Action IC0701 FoVeOOS: Formal Verification of Object-Oriented Software (2008-2012) www.cost-ic0701.org/

Department of Computer Science - University of Turin http://www.di.unito.it/

Research infrastructures

- Interdepartmental Competence Centre for Scientific Computing (C3S) https://c3s.unito.it/
- The Turin's High-Performance Centre for Artificial Intelligence (HPC4AI) https://hpc4ai.unito.it/ (POR-FESR 2014-2020)
- Planned "IoT, Robotics and 5G lab" with the Deparment Agricultural, Forest and Food Sciences **(SSD: AGR/09 - meccanica agraria)**

Collaborations with industries and institutions

- Partner of Competence Industry Manufacturing 4.0 (CIM 4.0) https://cim40.com/
- Partner of Torino City Lab https://www.torinocitylab.it/
- Interdepartmental Center of for technology transfer to local enterprises and institutions (ICxT) https://icxt.di.unito.it/

Outline

1. ***Distributed*** Runtime Verification
2. A Language for Specifying ***Distributed*** Monitors
3. A Language for Implementing ***Distributed*** Monitors
4. Automatic Generation of ***Distributed*** Monitors
5. Case study: crowd safety
6. Ongoing/Planned Work

Outline

1. ***Distributed*** Runtime Verification
2. A Language for Specifying ***Distributed*** Monitors
3. A Language for Implementing ***Distributed*** Monitors
4. Automatic Generation of ***Distributed*** Monitors
5. Case study: crowd safety
6. Ongoing/Planned Work

A lightweight formal method

Runtime Verification (RV)[1,2]

[1] M. Leucker, C. Schallhart (2009). A brief account of runtime verification. J. Log. Algebr. Program. 78 (5), 293–303.http://dx.doi.org/10.1016/j.jlap.2008.08.004
[2] COST Action "ARVI: Runtime Verification beyond Monitoring" https://www.cost-arvi.eu/

A lightweight formal method

Runtime Verification (RV)[1,2]

- observing a system at runtime
- by monitors generated from formal specifications

[1] M. Leucker, C. Schallhart (2009). A brief account of runtime verification. J. Log. Algebr. Program. 78 (5), 293–303.http://dx.doi.org/10.1016/j.jlap.2008.08.004
[2] COST Action "ARVI: Runtime Verification beyond Monitoring" https://www.cost-arvi.eu/

A lightweight formal method

Runtime Verification (RV)[1,2]

- observing a system at runtime
- by monitors generated from formal specifications
  - linear time logic, regular expressions,...
  - trace- or stream-based (with events mapped to atomic propositions)

[1] M. Leucker, C. Schallhart (2009). A brief account of runtime verification. J. Log. Algebr. Program. 78 (5), 293–303.http://dx.doi.org/10.1016/j.jlap.2008.08.004
[2] COST Action "ARVI: Runtime Verification beyond Monitoring" https://www.cost-arvi.eu/

A lightweight formal method

Runtime Verification (RV)[1,2]

- observing a system at runtime
- by monitors generated from formal specifications
  - linear time logic, regular expressions,...
  - trace- or stream-based (with events mapped to atomic propositions)

RV for CPSs

- feasible despite complexity of deployment scenarios[3]

[1] M. Leucker, C. Schallhart (2009). A brief account of runtime verification. J. Log. Algebr. Program. 78 (5), 293–303.http://dx.doi.org/10.1016/j.jlap.2008.08.004

[2] COST Action "ARVI: Runtime Verification beyond Monitoring" https://www.cost-arvi.eu/

[3] A. Bennaceur et al. (2019). *Modelling and analysing resilient cyber-physical systems.* In: IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2019), pp. 70-76. https://doi.org/10.1109/SEAMS.2019.00018

State-of-the-art and Contribution

STATE-OF-THE-ART:[1] Distribute RV includes both

- monitoring of distributed systems
- use of distributed systems for monitoring

[1] A. Francalanza, J.A. Pérez, C. Sánchez (2018). *Runtime verification for decentralised and distributed systems. In: Lectures on Runtime Verification - Introductory and Advanced Topics.* LNCS 10457, Springer, pp. 176-210.
https://doi.org/10.1007/978-3-319-75632-5_6

State-of-the-art and Contribution

STATE-OF-THE-ART:[1] Distribute RV includes both

- monitoring of distributed systems
- use of distributed systems for monitoring

Simplifying assumptions

- absence of failures
- absence of mobility

[1] A. Francalanza, J.A. Pérez, C. Sánchez (2018). *Runtime verification for decentralised and distributed systems. In: Lectures on Runtime Verification - Introductory and Advanced Topics.* LNCS 10457, Springer, pp. 176-210.
https://doi.org/10.1007/978-3-319-75632-5_6

State-of-the-art and Contribution

STATE-OF-THE-ART:[1] Distribute RV includes both

- monitoring of distributed systems
- use of distributed systems for monitoring

Simplifying assumptions

- absence of failures
- absence of mobility

CONTRIBUTION:[2,3] distributed RV of open systems of mobile agents – unreliability/failure issues

[1] A. Francalanza, J.A. Pérez, C. Sánchez (2018). *Runtime verification for decentralised and distributed systems. In: Lectures on Runtime Verification - Introductory and Advanced Topics.* LNCS 10457, Springer, pp. 176-210. https://doi.org/10.1007/978-3-319-75632-5_6

[2] G. Audrito, R. Casadei, F. Damiani, V. Stolz, M. Viroli (2021). *Adaptive distributed monitors of spatial properties for cyber–physical systems.* Journal of Systems and Software, 2021, Volume 175. https://doi.org/10.1016/j.jss.2021.110908

[3] G. Audrito, F. Damiani, V. Stolz, G. Torta, M. Viroli (SUBMITTED FOR PUBLICATION). *Distributed Runtime Verification for Edge Computing by Past-CTL and the Field Calculus.*

State-of-the-art and Contribution

STATE-OF-THE-ART:[1] Distribute RV includes both

- monitoring of distributed systems
- use of distributed systems for monitoring

Simplifying assumptions

- absence of failures
- absence of mobility

CONTRIBUTION:[2,3] distributed RV of open systems of mobile agents – unreliability/failure issues

- number of participants

[1] A. Francalanza, J.A. Pérez, C. Sánchez (2018). *Runtime verification for decentralised and distributed systems. In: Lectures on Runtime Verification - Introductory and Advanced Topics.* LNCS 10457, Springer, pp. 176-210. https://doi.org/10.1007/978-3-319-75632-5_6

[2] G. Audrito, R. Casadei, F. Damiani, V. Stolz, M. Viroli (2021). *Adaptive distributed monitors of spatial properties for cyber–physical systems.* Journal of Systems and Software, 2021, Volume 175. https://doi.org/10.1016/j.jss.2021.110908

[3] G. Audrito, F. Damiani, V. Stolz, G. Torta, M. Viroli (SUBMITTED FOR PUBLICATION). *Distributed Runtime Verification for Edge Computing by Past-CTL and the Field Calculus.*

State-of-the-art and Contribution

STATE-OF-THE-ART:[1] Distribute RV includes both

- monitoring of distributed systems
- use of distributed systems for monitoring

Simplifying assumptions

- absence of failures
- absence of mobility

CONTRIBUTION:[2,3] distributed RV of open systems of mobile agents – unreliability/failure issues

- number of participants
- communication topology

[1] A. Francalanza, J.A. Pérez, C. Sánchez (2018). *Runtime verification for decentralised and distributed systems. In: Lectures on Runtime Verification - Introductory and Advanced Topics.* LNCS 10457, Springer, pp. 176-210. https://doi.org/10.1007/978-3-319-75632-5_6

[2] G. Audrito, R. Casadei, F. Damiani, V. Stolz, M. Viroli (2021). *Adaptive distributed monitors of spatial properties for cyber–physical systems.* Journal of Systems and Software, 2021, Volume 175. https://doi.org/10.1016/j.jss.2021.110908

[3] G. Audrito, F. Damiani, V. Stolz, G. Torta, M. Viroli (SUBMITTED FOR PUBLICATION). *Distributed Runtime Verification for Edge Computing by Past-CTL and the Field Calculus.*

State-of-the-art and Contribution

STATE-OF-THE-ART:[1] Distribute RV includes both

- monitoring of distributed systems
- use of distributed systems for monitoring

Simplifying assumptions

- absence of failures
- absence of mobility

CONTRIBUTION:[2,3] distributed RV of open systems of mobile agents – unreliability/failure issues

- number of participants
- communication topology
- performance of (broadcast) messages

[1] A. Francalanza, J.A. Pérez, C. Sánchez (2018). *Runtime verification for decentralised and distributed systems. In: Lectures on Runtime Verification - Introductory and Advanced Topics.* LNCS 10457, Springer, pp. 176-210. https://doi.org/10.1007/978-3-319-75632-5_6

[2] G. Audrito, R. Casadei, F. Damiani, V. Stolz, M. Viroli (2021). *Adaptive distributed monitors of spatial properties for cyber–physical systems.* Journal of Systems and Software, 2021, Volume 175. https://doi.org/10.1016/j.jss.2021.110908

[3] G. Audrito, F. Damiani, V. Stolz, G. Torta, M. Viroli (SUBMITTED FOR PUBLICATION). *Distributed Runtime Verification for Edge Computing by Past-CTL and the Field Calculus.*

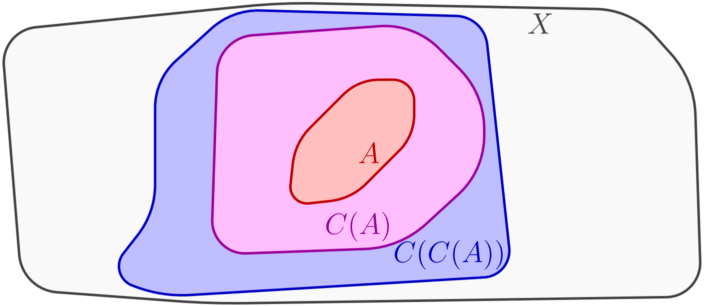
Outline

Quasi-discrete Closure Spaces

Definition

c A quasi-discrete closure space is a set X with a *closure operator* $C : 2^X \to 2^X$ such that:

$C(\emptyset) = \emptyset$ $\quad A \subseteq C(A)$ $\quad C(A \cup B) = C(A) \cup C(B)$ $\quad C(A) = \bigcup_{x \in A} C(\{x\})$

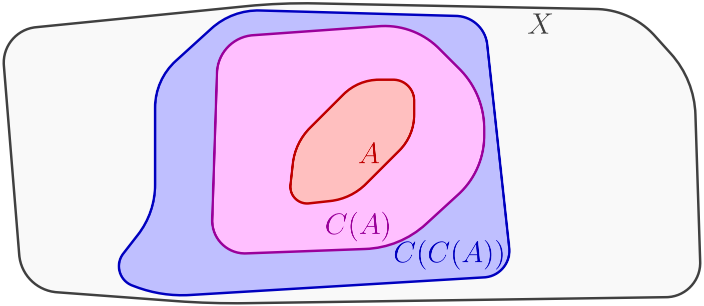


Quasi-discrete Closure Spaces

Definition

c A quasi-discrete closure space is a set X with a *closure operator* $C : 2^X \to 2^X$ such that:

$C(\emptyset) = \emptyset$ $\quad A \subseteq C(A)$ $\quad C(A \cup B) = C(A) \cup C(B)$ $\quad C(A) = \bigcup_{x \in A} C(\{x\})$



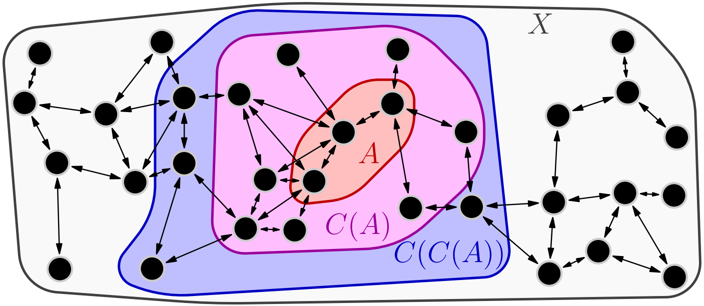
Characterised as graphs where $C(v)$ are the neighbours of v

Quasi-discrete Closure Spaces

Definition

c A quasi-discrete closure space is a set X with a *closure operator* $C : 2^X \to 2^X$ such that:

$C(\emptyset) = \emptyset$ $A \subseteq C(A)$ $C(A \cup B) = C(A) \cup C(B)$ $C(A) = \bigcup_{x \in A} C(\{x\})$



Characterised as graphs where $C(v)$ are the neighbours of v

Spatial Logic of Closure Spaces (SLCS)

$\phi ::= \bot \mid \top \mid q \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \Rightarrow \phi) \mid (\phi \Leftrightarrow \phi)$	logical op.
$\mid (\Box\, \phi) \mid (\Diamond\, \phi) \mid (\partial\, \phi) \mid (\partial^{-}\, \phi) \mid (\partial^{+}\, \phi)$	local modalities
$\mid (\phi\, \mathcal{R}\, \phi) \mid (\phi\, \mathcal{T}\, \phi) \mid (\phi\, \mathcal{U}\, \phi) \mid (\mathcal{G}\, \phi) \mid (\mathcal{F}\, \phi)$	global modalities

Spatial Logic of Closure Spaces (SLCS)

$$\phi ::= \bot \mid \top \mid q \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \Rightarrow \phi) \mid (\phi \Leftrightarrow \phi)$$ logical op.

$$\mid (\Box\, \phi) \mid (\Diamond\, \phi) \mid (\partial\, \phi) \mid (\partial^{-}\, \phi) \mid (\partial^{+}\, \phi)$$ local modalities

$$\mid (\phi\, \mathcal{R}\, \phi) \mid (\phi\, \mathcal{T}\, \phi) \mid (\phi\, \mathcal{U}\, \phi) \mid (\mathcal{G}\, \phi) \mid (\mathcal{F}\, \phi)$$ global modalities

Local modalities

- $\Diamond\, \phi$ (closure) holds at points with some neighbour satisfying ϕ
- $\Box\, \phi$ (interior) holds at points with all neighbours satisfying ϕ
- $\partial\, \phi$ (boundary) holds at points with some (not all) neighbours satisfying ϕ
- $\partial^{-}\, \phi$ (interior boundary) holds where ϕ and some neighbour $\neg\phi$
- $\partial^{-}\, \phi$ (closure boundary) holds where $\neg\phi$ and some neighbour ϕ

Spatial Logic of Closure Spaces (SLCS)

$$\phi ::= \bot \mid \top \mid q \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \Rightarrow \phi) \mid (\phi \Leftrightarrow \phi)$$ logical op.

$$\mid (\Box\, \phi) \mid (\Diamond\, \phi) \mid (\partial\, \phi) \mid (\partial^{-}\, \phi) \mid (\partial^{+}\, \phi)$$ local modalities

$$\mid (\phi\, \mathcal{R}\, \phi) \mid (\phi\, \mathcal{T}\, \phi) \mid (\phi\, \mathcal{U}\, \phi) \mid (\mathcal{G}\, \phi) \mid (\mathcal{F}\, \phi)$$ global modalities

Local modalities

- $\Diamond\, \phi$ (closure) holds at points with some neighbour satisfying ϕ

Global modalities

- $\phi\, \mathcal{R}\, \psi$ (reaches) holds at the end of paths satisfying ϕ and starting in a point satisfying ψ
- $\phi\, \mathcal{T}\, \psi$ (touches) like reachability but ϕ may not hold in the starting point
- $\phi\, \mathcal{U}\, \psi$ (surrounded by) holds in points in an area satisfying ϕ whose neighbours satisfy ψ
- $\mathcal{G}\, \phi$ (everywhere) holds in points where ϕ is true in every point of every incoming path
- $\mathcal{F}\, \phi$ (somewhere) holds in points where ϕ is true in some point of some incoming path

Spatial Logic of Closure Spaces (SLCS)[1]

$\phi ::= \top \mid q \mid (\neg\phi) \mid (\phi \vee \phi) \mid (\Diamond\, \phi) \mid (\phi\, \mathcal{R}\, \phi)$ fundamental op.

$\Box\, \phi \triangleq \neg(\Diamond(\neg\phi))$ $\quad \partial\, \phi \triangleq (\Diamond\, \phi) \wedge \neg(\Box\, \phi)$ $\quad \partial^-\, \phi \triangleq \phi \wedge \neg(\Box\, \phi)$ $\quad \partial^+\, \phi \triangleq (\Diamond\, \phi) \wedge \neg\phi$

$\phi\, \mathcal{T}\, \psi \triangleq \phi\, \mathcal{R}(\Diamond\, \psi)$ $\quad \phi\, \mathcal{U}\, \psi \triangleq \phi \wedge \Box\neg(\neg\psi\, \mathcal{R}\, \neg\phi)$ $\quad \mathcal{F}\, \phi \triangleq \top\, \mathcal{R}\, \phi$ $\quad \mathcal{G}\, \phi \triangleq \neg\, \mathcal{F}\, \neg\phi$

Local modalities

- $\Diamond\, \phi$ (closure) holds at points with some neighbour satisfying ϕ

Global modalities

- $\phi\, \mathcal{R}\, \psi$ (reaches) holds at the end of paths satisfying ϕ and starting in a point satisfying ψ

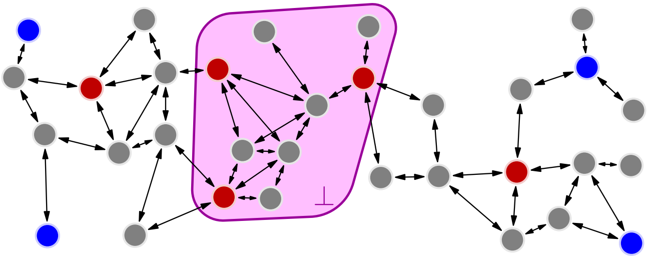
two modalities are fundamental, the rest is derived
($\mathcal{R}$ chosen for presentation convenience)

[1] Ciancia, V. et a. (2014). *Specifying and verifying properties of space.* In: IFIP 8th International Conference in Theoretical Computer Science (TCS 2014), LNCS 8705, Springer, pp. 222–235. http://dx.doi.org/10.1007/978-3-662-44602-7_18

Sample Emergency Applications

- D: true on devices in a **dangerous** area (TRUE in red points)
- R: true on devices in **recovery** points (TRUE in blue points)

$\neg D \, \mathcal{R} \, R$: true on devices that can reach a recovery point through devices in non-dangerous areas (FALSE in purple area)

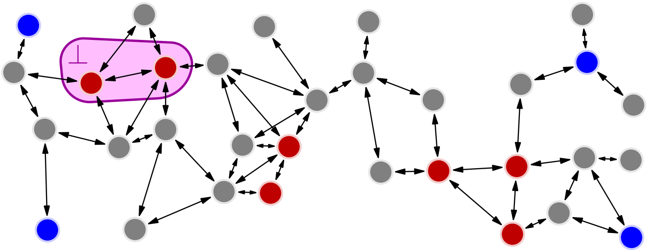


Sample Emergency Applications

- D: true on devices in a **dangerous** area (TRUE in red points)
- R: true on devices in **recovery** points (TRUE in blue points)
- $\neg D \mathcal{R} R$: true on devices that can reach a recovery point through devices in non-dangerous areas

$$D \Rightarrow (D\,\mathcal{U}(\neg D\,\mathcal{R}\,R)):$$

true on devices in dangerous areas that are surrounded by devices in non-dangerous areas from which devices can reach a recovery point without passing through any other device in dangerous areas (FALSE in purple area)



Outline

Aggregate Programming[1] (a modern model of computation for proximity-based interaction networks)

Network of (possible mobile) devices

- dynamic topology — induced by (physical or logical) proximity of devices
- each device can receive/send message to/from devices in the neighbourhood

[1] J. Beal, D. Pianini, M. Viroli. *Aggregate programming for the Internet of Things.* IEEE Computer, 48(9), 2015. https://doi.org/10.1109/MC.2015.261

Aggregate Programming[1] (a modern model of computation for proximity-based interaction networks)

Network of (possible mobile) devices

- dynamic topology — induced by (physical or logical) proximity of devices
- each device can receive/send message to/from devices in the neighbourhood

Each device (asynchronously periodically) fires:

1. collects sensor data and received messages
2. executes a program
3. sends messages (and performs some actuation)

[1] J. Beal, D. Pianini, M. Viroli. *Aggregate programming for the Internet of Things.* IEEE Computer, 48(9), 2015. https://doi.org/10.1109/MC.2015.261

Aggregate Programming[1] (a modern model of computation for proximity-based interaction networks)

Network of (possible mobile) devices

- dynamic topology — induced by (physical or logical) proximity of devices
- each device can receive/send message to/from devices in the neighbourhood

Each device (asynchronously periodically) fires:

1. collects sensor data and received messages
2. executes a program — the same for all devices[2]
3. sends messages (and performs some actuation)

[1] J. Beal, D. Pianini, M. Viroli. *Aggregate programming for the Internet of Things.* IEEE Computer, 48(9), 2015. https://doi.org/10.1109/MC.2015.261

[2] Supports integrated development of every part of a distribute application – cf. *macroprogramming* and *multi-tier programming*:
• R. Newton and M.Welsh (2004). *Region streams: Functional macroprogramming for sensor networks.* In: 1st Workshop on Data Management for Sensor Networks (MSN 2004), pp. 78–87. https://doi.org/10.1145/1052199.1052213
• P. Weisenburger, M. Köhler, G. Salvaneschi (2018). *Distributed system development with Scalaloci.* In: ACM Programming Languages, 2 (OOPSLA):129:1–129:30, 2018. https://doi.org/10.1145/3276499

Aggregate Programming[1] (a modern model of computation for proximity-based interaction networks)

Network of (possible mobile) devices

- dynamic topology — induced by (physical or logical) proximity of devices
- each device can receive/send message to/from devices in the neighbourhood

Each device (asynchronously periodically) fires:

1. collects sensor data and received messages
2. executes a program — the same for all devices[2]
3. sends messages (and performs some actuation)

Event: beginning of a firing — event ϵ happens on device $\delta(\epsilon)$

[1] J. Beal, D. Pianini, M. Viroli. *Aggregate programming for the Internet of Things.* IEEE Computer, 48(9), 2015. https://doi.org/10.1109/MC.2015.261

[2] Supports integrated development of every part of a distribute application – cf. *macroprogramming* and *multi-tier programming*:
• R. Newton and M.Welsh (2004). *Region streams: Functional macroprogramming for sensor networks.* In: 1st Workshop on Data Management for Sensor Networks (MSN 2004), pp. 78–87. https://doi.org/10.1145/1052199.1052213
• P. Weisenburger, M. Köhler, G. Salvaneschi (2018). *Distributed system development with Scalaloci.* In: ACM Programming Languages, 2 (OOPSLA):129:1–129:30, 2018. https://doi.org/10.1145/3276499

Aggregate Programming[1] (a modern model of computation for proximity-based interaction networks)

Network of (possible mobile) devices

- dynamic topology — induced by (physical or logical) proximity of devices
- each device can receive/send message to/from devices in the neighbourhood

Each device (asynchronously periodically) fires:

1. collects sensor data and received messages
2. executes a program — the same for all devices[2]
3. sends messages (and performs some actuation)

Event: beginning of a firing — event ϵ happens on device $\delta(\epsilon)$

Supplier event: ϵ' supplier of ϵ ($\epsilon' \rightsquigarrow \epsilon$) iff at ϵ:
a (not expired) message sent by ϵ' was the last from $\delta(\epsilon')$ able to reach $\delta(\epsilon)$

---

[1] J. Beal, D. Pianini, M. Viroli. *Aggregate programming for the Internet of Things.* IEEE Computer, 48(9), 2015. https://doi.org/10.1109/MC.2015.261

[2] Supports integrated development of every part of a distribute application – cf. *macroprogramming* and *multi-tier programming*:
• R. Newton and M.Welsh (2004). *Region streams: Functional macroprogramming for sensor networks.* In: 1st Workshop on Data Management for Sensor Networks (MSN 2004), pp. 78–87. https://doi.org/10.1145/1052199.1052213
• P. Weisenburger, M. Köhler, G. Salvaneschi (2018). *Distributed system development with Scalaloci.* In: ACM Programming Languages, 2 (OOPSLA):129:1–129:30, 2018. https://doi.org/10.1145/3276499

Aggregate Programming framework and toolchain

1. Minimal core language (field calculus)

- Aggregate Programming = Macroprogramming + Field-based coordination[1]
- formal semantics
- properties

$\mathtt{P} ::= \overline{\mathtt{F}}\ \mathtt{e}$	program
$\mathtt{F} ::= \mathtt{def}\ \mathtt{d}(\overline{\mathtt{x}})\ \{\mathtt{e}\}$	function declaration
$\mathtt{e} ::= \mathtt{x} \mid \mathtt{f}(\overline{\mathtt{e}}) \mid \mathtt{v} \mid \mathtt{if(e)\{e\}else\{e\}} \mid \mathtt{nbr\{e\}} \mid \mathtt{share(e)\{(x)=>e\}}$	expression
$\mathtt{f} ::= \mathtt{d} \mid \mathtt{b}$	function name
$\mathtt{v} ::= \ell \mid \phi$	value
$\ell ::= \mathtt{c}(\overline{\ell})$	local value
$\phi ::= \overline{\delta} \mapsto \overline{\ell}$	neighbouring value

[1] G. Audrito, J. Beal, F. Damiani, D. Pianini, M. Viroli (2020). *Field-based coordination with the share operator.* Logical Methods in Computer Science 16(4),1, pp. 1-41 https://doi.org/10.23638/LMCS-16(4:1)2020

Aggregate Programming framework and toolchain

$\mathtt{P} ::= \overline{\mathtt{F}}\ \mathtt{e}$	program
$\mathtt{F} ::= \mathtt{def}\ \mathtt{d}(\overline{\mathtt{x}})\ \{\mathtt{e}\}$	function declaration
$\mathtt{e} ::= \mathtt{x} \mid \mathtt{f}(\overline{\mathtt{e}}) \mid \mathtt{v} \mid \mathtt{if}(\mathtt{e})\{\mathtt{e}\}\mathtt{else}\{\mathtt{e}\} \mid \mathtt{nbr}\{\mathtt{e}\} \mid \mathtt{share}(\mathtt{e})\{(\mathtt{x})\text{=>}\mathtt{e}\}$	expression
$\mathtt{f} ::= \mathtt{d} \mid \mathtt{b}$	function name
$\mathtt{v} ::= \ell \mid \phi$	value
$\ell ::= \mathtt{c}(\overline{\ell})$	local value
$\phi ::= \overline{\delta} \mapsto \overline{\ell}$	neighbouring value

1. Minimal core language (field calculus)

- Aggregate Programming = Macroprogramming + Field-based coordination[1]
- formal semantics
- properties

2. Implementations

- *Protelis*: a Java external DSL http://protelis.github.io/
- *ScaFi*: a Scala internal DSL (library) https://scafi.github.io/
- *FCPP*: a C++ internal DSL (library) https://fcpp.github.io/

[1] G. Audrito, J. Beal, F. Damiani, D. Pianini, M. Viroli (2020). *Field-based coordination with the share operator.* Logical Methods in Computer Science 16(4),1, pp. 1-41 https://doi.org/10.23638/LMCS-16(4:1)2020

Aggregate Programming framework and toolchain

1. Minimal core language (field calculus)

- Aggregate Programming = Macroprogramming + Field-based coordination[1]
- formal semantics
- properties

P ::= $\overline{\mathtt{F}}$ e	program
F ::= def d($\overline{\mathtt{x}}$) {e}	function declaration
e ::= x | f($\overline{\mathtt{e}}$) | v | if(e){e}else{e} | nbr{e} | share(e){(x)=>e}	expression
f ::= d | b	function name
v ::= ℓ | ϕ	value
ℓ ::= c($\overline{\ell}$)	local value
ϕ ::= $\overline{\delta} \mapsto \overline{\ell}$	neighbouring value

2. Implementations

- *Protelis*: a Java external DSL `http://protelis.github.io/`
- *ScaFi*: a Scala internal DSL (library) `https://scafi.github.io/`
- *FCPP*: a C++ internal DSL (library) `https://fcpp.github.io/`

```
DEF() double abf(ARGS, bool source) { CODE
    return nbr(CALL, INF, [&] (field<double> d) {
        double v = source ? 0.0 : INF;
        return min_hood(CALL, d + node.nbr_dist(), v);
    });
}
```

[1] G. Audrito, J. Beal, F. Damiani, D. Pianini, M. Viroli (2020). *Field-based coordination with the share operator.* Logical Methods in Computer Science 16(4),1, pp. 1-41 `https://doi.org/10.23638/LMCS-16(4:1)2020`

Aggregate Programming framework and toolchain

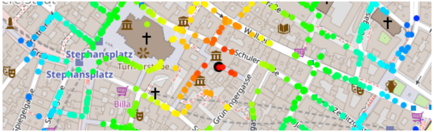
1. Minimal core language (field calculus)
   - Aggregate Programming = Macroprogramming + Field-based coordination[1]
   - formal semantics
   - properties

P ::= $\overline{\text{F}}$ e		program
F ::= def d($\overline{\text{x}}$) {e}		function declaration
e ::= x | f($\overline{\text{e}}$) | v | if(e){e}else{e} | nbr{e} | share(e){(x)=>e}		expression
f ::= d | b		function name
v ::= ℓ | ϕ		value
ℓ ::= c($\overline{\ell}$)		local value
ϕ ::= $\overline{\delta} \mapsto \overline{\ell}$		neighbouring value

2. Implementations
   - *Protelis*: a Java external DSL http://protelis.github.io/
   - *ScaFi*: a Scala internal DSL (library) https://scafi.github.io/
   - *FCPP*: a C++ internal DSL (library) https://fcpp.github.io/

```
DEF() double abf(ARGS, bool source) { CODE
    return nbr(CALL, INF, [&] (field<double> d) {
        double v = source ? 0.0 : INF;
        return min_hood(CALL, d + node.nbr_dist(), v);
    });
}
```

3. Alchemist simulator (UniBo), FCPP simulator (UniTo), deployment (Raytheon BBN Technologies)[2]



https://alchemistsimulator.github.io/

[1] G. Audrito, J. Beal, F. Damiani, D. Pianini, M. Viroli (2020). *Field-based coordination with the share operator.* Logical Methods in Computer Science 16(4),1, pp. 1-41 https://doi.org/10.23638/LMCS-16(4:1)2020

[2] DARPA project (http://www.swarmtactics.com/)

Properties

- Self-stabilisation: behaviour is guaranteed to eventually attain a correct and stable final state despite any transient perturbation in state or topology[1]

[1] M. Viroli, G. Audrito, J. Beal, F. Damiani, D. Pianini (2018). *Engineering resilient collective adaptive systems by self-stabilisation.* ACM Transactions on Modeling and Computer Simulation, 28(2), 2018. https://doi.org/10.1145/3177774

Properties

- Self-stabilisation: behaviour is guaranteed to eventually attain a correct and stable final state despite any transient perturbation in state or topology[1]
- Eventual consistency: behaviour converges to a final state that approximates a predictable limit, based on the continuous environment, as the density and speed of devices increases[2]

[1] M. Viroli, G. Audrito, J. Beal, F. Damiani, D. Pianini (2018). *Engineering resilient collective adaptive systems by self-stabilisation.* ACM Transactions on Modeling and Computer Simulation, 28(2), 2018. `https://doi.org/10.1145/3177774`
[2] J. Beal, M. Viroli, D. Pianini, F. Damiani (2017). *Self-adaptation to device distribution in the internet of things.* ACM Transactions on Autonomous and Adaptive Systems 12(3), 2017. `https://doi.org/10.1145/3105758`

Properties

- Self-stabilisation: behaviour is guaranteed to eventually attain a correct and stable final state despite any transient perturbation in state or topology[1]
- Eventual consistency: behaviour converges to a final state that approximates a predictable limit, based on the continuous environment, as the density and speed of devices increases[2]
- Certified error bounds: linking the quality of the services to the amount of computing resources dedicated[3]
- ...

[1] M. Viroli, G. Audrito, J. Beal, F. Damiani, D. Pianini (2018). *Engineering resilient collective adaptive systems by self-stabilisation.* ACM Transactions on Modeling and Computer Simulation, 28(2), 2018. `https://doi.org/10.1145/3177774`
[2] J. Beal, M. Viroli, D. Pianini, F. Damiani (2017). *Self-adaptation to device distribution in the internet of things.* ACM Transactions on Autonomous and Adaptive Systems 12(3), 2017. `https://doi.org/10.1145/3105758`
[3] G. Audrito, F. Damiani, M. Viroli, E. Bini (2018). *Distributed Real-Time Shortest-Paths Computations with the Field Calculus.* In: IEEE 39th Real-Time Systems Symposium (RTSS 2018). `https://doi.org/10.1109/RTSS.2018.00013`

Outline

SLCS Translation in Field Calculus (hence in C++, Protelis, Scala)

$[\![\top]\!]$	= `true`	$[\![\phi_1 \vee \phi_2]\!]$	= $[\![\phi_1]\!]$ `||` $[\![\phi_2]\!]$
$[\![\bot]\!]$	= `false`	$[\![\phi_1 \wedge \phi_2]\!]$	= $[\![\phi_1]\!]$ `&&` $[\![\phi_2]\!]$
$[\![q]\!]$	= `q()`	$[\![\phi_1 \Rightarrow \phi_2]\!]$	= $[\![\phi_1]\!]$ `<=` $[\![\phi_2]\!]$
$[\![\neg\phi]\!]$	= `!`$[\![\phi]\!]$	$[\![\phi_1 \Leftrightarrow \phi_2]\!]$	= $[\![\phi_1]\!]$ `==` $[\![\phi_2]\!]$
$[\![\lozenge\, \phi]\!]$	= `anyHoodPlusSelf(`**`nbr`**`{`$[\![\phi]\!]$`})`	$[\![\phi_1\, \mathcal{R}\, \phi_2]\!]$	= **`reaches`**`(`$[\![\phi_1]\!]$`, `$[\![\phi_2]\!]$`)`
$[\![\square\, \phi]\!]$	= `allHoodPlusSelf(`**`nbr`**`{`$[\![\phi]\!]$`})`	$[\![\mathcal{F}\, \phi]\!]$	= **`somewhere`**`(`$[\![\phi]\!]$`)`

```
def distanceTo(dest) {
  share (infinity){ (d) => mux (dest) {0} else {minHood(d)+1} }
}
def somewhere(x) {
  distanceTo(x) <= D
}
def reaches(x, y) {
  if (x) {somewhere(y)} else {false}
}
```

SLCS Translation in Field Calculus (hence in C++, Protelis, Scala)

$[\![\top]\!]$	= `true`	$[\![\phi_1 \vee \phi_2]\!]$	= $[\![\phi_1]\!]$ `||` $[\![\phi_2]\!]$
$[\![\bot]\!]$	= `false`	$[\![\phi_1 \wedge \phi_2]\!]$	= $[\![\phi_1]\!]$ `&&` $[\![\phi_2]\!]$
$[\![q]\!]$	= `q()`	$[\![\phi_1 \Rightarrow \phi_2]\!]$	= $[\![\phi_1]\!]$ `<=` $[\![\phi_2]\!]$
$[\![\neg\phi]\!]$	= `!`$[\![\phi]\!]$	$[\![\phi_1 \Leftrightarrow \phi_2]\!]$	= $[\![\phi_1]\!]$ `==` $[\![\phi_2]\!]$
$[\![\Diamond\, \phi]\!]$	= `anyHoodPlusSelf(nbr{`$[\![\phi]\!]$`})`	$[\![\phi_1\, \mathcal{R}\, \phi_2]\!]$	= `reaches(`$[\![\phi_1]\!]$`, `$[\![\phi_2]\!]$`)`
$[\![\Box\, \phi]\!]$	= `allHoodPlusSelf(nbr{`$[\![\phi]\!]$`})`	$[\![\mathcal{F}\, \phi]\!]$	= `somewhere(`$[\![\phi]\!]$`)`

```
def distanceTo(dest) {
  share (infinity){ (d) => mux (dest) {0} else {minHood(d)+1} }
}
def somewhere(x) {
  distanceTo(x) <= D
}
def reaches(x, y) {
  if (x) {somewhere(y)} else {false}
}
```

Theorem (Lightweightness)

$[\![\phi]\!]$ computes in each node using message size $\mathbf{O}(S)$ and computation time/space $\mathbf{O}(L + S\,N)$, where

- N is neighbourhood size
- L, S are the numbers of logical and spatial operators in ϕ

SLCS Translation in Field Calculus (hence in C++, Protelis, Scala)

$[\![\top]\!]$	= `true`	$[\![\phi_1 \vee \phi_2]\!]$	= $[\![\phi_1]\!]$ `||` $[\![\phi_2]\!]$
$[\![\bot]\!]$	= `false`	$[\![\phi_1 \wedge \phi_2]\!]$	= $[\![\phi_1]\!]$ `&&` $[\![\phi_2]\!]$
$[\![q]\!]$	= `q()`	$[\![\phi_1 \Rightarrow \phi_2]\!]$	= $[\![\phi_1]\!]$ `<=` $[\![\phi_2]\!]$
$[\![\neg\phi]\!]$	= `!`$[\![\phi]\!]$	$[\![\phi_1 \Leftrightarrow \phi_2]\!]$	= $[\![\phi_1]\!]$ `==` $[\![\phi_2]\!]$
$[\![\Diamond\, \phi]\!]$	= `anyHoodPlusSelf(nbr{`$[\![\phi]\!]$`})`	$[\![\phi_1\, \mathcal{R}\, \phi_2]\!]$	= `reaches(`$[\![\phi_1]\!]$`,` $[\![\phi_2]\!]$`)`
$[\![\Box\, \phi]\!]$	= `allHoodPlusSelf(nbr{`$[\![\phi]\!]$`})`	$[\![\mathcal{F}\, \phi]\!]$	= `somewhere(`$[\![\phi]\!]$`)`

```
def distanceTo(dest) {
  share (infinity){ (d) => mux (dest) {0} else {minHood(d)+1} }
}
def somewhere(x) {
  distanceTo(x) <= D
}
def reaches(x, y) {
  if (x) {somewhere(y)} else {false}
}
```

Theorem (Lightweightness)

$[\![\phi]\!]$ computes in each node using message size $\mathbf{O}(S)$ and computation time/space $\mathbf{O}(L + S\,N)$, where

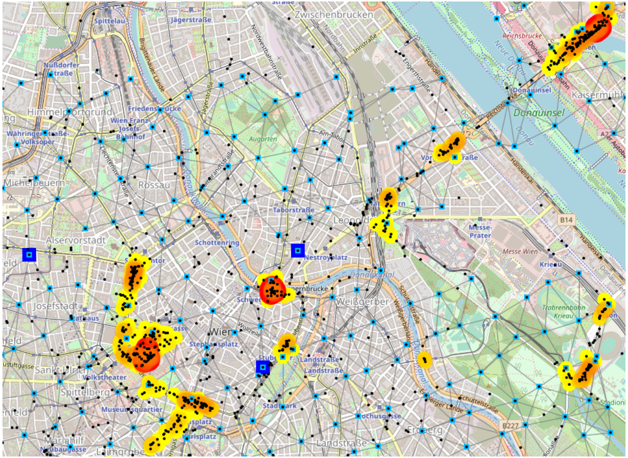
- N is neighbourhood size
- L, S are the numbers of logical and spatial operators in ϕ

Theorem (Self-Stabilisation, Correctness, Optimality)

$[\![\phi]\!]$ self-stabilises to the interpretation of ϕ in the smallest worst case of full rounds of execution.

Outline

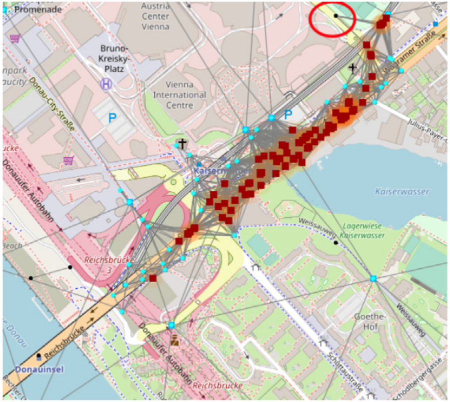
A snapshot of the simulated scenario, as a network of devices in the city of Wien



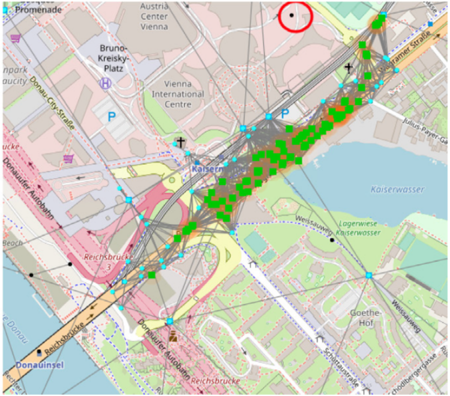
- black dots denote (the smartphones of) people corresponding to the GPS traces of the reference mass event
- grey links represent connectivity (i.e., the neighbouring relationships)
- yellow, orange, and red overlays represent increasing levels of crowding
- blue squares denote safe places (these are real locations of hospital facilities)
- small, light blue squares represent access points

Zoomed snapshots of the scenario meant to illustrate the property checking

(we assume safe areas are only south of the river, and there are no paths that circumvent the Reichsbrücke bridge shown in the picture)

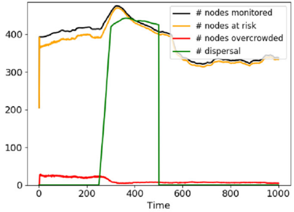


(a) The red-circled node has every path to a safe node hindered by a dangerous, crowded area—which is red-coloured to denote its collective failure in satisfying the property. The cyan circles denote nodes able to reach safety.

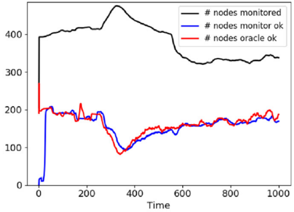


(b) The red-circled node walks away, detaching from the network. All the remaining nodes can reach a safe area (not shown) by passing across the bridge: therefore, the crowded area satisfies the property.

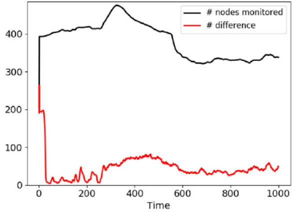
Simulation results



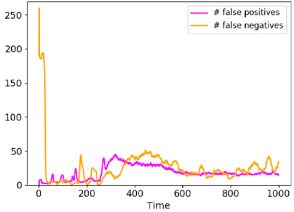
(a) Number of devices that perceive a dangerous (red line) or moderate (orange line) overcrowding, monitored (black line), and following dispersal advices (green line).



(b) Number of monitored devices (black line) that satisfy the property according to the oracle (red line) and the monitor (blue line).



(c) Number of monitored devices (black line) for which the oracle and the monitor provide a different response (red line).



(d) Detail of the error in terms of the number of devices providing false positives (magenta line) and false negatives (orange line).

Outline

Aggregate Programming: some ongoing/planned work

Deployment of FCPP on MIOSIX (`https://miosix.org/`) – ONGOING (implemented and tested on a network of Wandstem nodes)

- joint work UniTo - PoliMi (W. Fornaciari, F. Terraneo)

Aggregate Programming: some ongoing/planned work

Deployment of FCPP on MIOSIX (https://miosix.org/) – ONGOING (implemented and tested on a network of Wandstem nodes)

- joint work UniTo - PoliMi (W. Fornaciari, F. Terraneo)

Deployment of FCPP on Contiki OS and DecaWave DWM1001 Development Board (https://www.decawave.com/product/dwm1001-development-board/) – ONGOING

- joint work UniTo - **REPLY** (M. Griva,...)

Aggregate Programming: some ongoing/planned work

Deployment of FCPP on MIOSIX (`https://miosix.org/`) – ONGOING (implemented and tested on a network of Wandstem nodes)

- joint work UniTo - PoliMi (W. Fornaciari, F. Terraneo)

Deployment of FCPP on Contiki OS and DecaWave DWM1001 Development Board (`https://www.decawave.com/product/dwm1001-development-board/`) – ONGOING

- joint work UniTo - **REPLY** (M. Griva,...)

Kotlin internal DSL (library) – ONGOING

- joint work UniTo - **REPLY** (M. Griva,...)

Aggregate Programming: some ongoing/planned work

Deployment of FCPP on MIOSIX (https://miosix.org/) – ONGOING (implemented and tested on a network of Wandstem nodes)

- joint work UniTo - PoliMi (W. Fornaciari, F. Terraneo)

Deployment of FCPP on Contiki OS and DecaWave DWM1001 Development Board (https://www.decawave.com/product/dwm1001-development-board/) – ONGOING

- joint work UniTo - **REPLY** (M. Griva,...)

Kotlin internal DSL (library) – ONGOING

- joint work UniTo - **REPLY** (M. Griva,...)

AP for Control Theory – ONGOING

- joint work UniTo - University of Iowa (S. Dasgupta,...) - **Raytheon BBN Technologies** (J. Beal,...)

Aggregate Programming: some ongoing/planned work

Deployment of FCPP on MIOSIX (https://miosix.org/) – ONGOING (implemented and tested on a network of Wandstem nodes)

- joint work UniTo - PoliMi (W. Fornaciari, F. Terraneo)

Deployment of FCPP on Contiki OS and DecaWave DWM1001 Development Board (https://www.decawave.com/product/dwm1001-development-board/) – ONGOING

- joint work UniTo - **REPLY** (M. Griva,...)

Kotlin internal DSL (library) – ONGOING

- joint work UniTo - **REPLY** (M. Griva,...)

AP for Control Theory – ONGOING

- joint work UniTo - University of Iowa (S. Dasgupta,...) - **Raytheon BBN Technologies** (J. Beal,...)

AP as enabling technology for Edge AI – PLANNED

- joint work UniTo - UniBo (M. Viroli,...) - **Synesthesia** (A. Basso,...)

Aggregate Programming: some ongoing/planned work

Deployment of FCPP on MIOSIX (`https://miosix.org/`) – ONGOING (implemented and tested on a network of Wandstem nodes)
- joint work UniTo - PoliMi (W. Fornaciari, F. Terraneo)

Deployment of FCPP on Contiki OS and DecaWave DWM1001 Development Board (`https://www.decawave.com/product/dwm1001-development-board/`) – ONGOING
- joint work UniTo - **REPLY** (M. Griva,...)

Kotlin internal DSL (library) – ONGOING
- joint work UniTo - **REPLY** (M. Griva,...)

AP for Control Theory – ONGOING
- joint work UniTo - University of Iowa (S. Dasgupta,...) - **Raytheon BBN Technologies** (J. Beal,...)

AP as enabling technology for Edge AI – PLANNED
- joint work UniTo - UniBo (M. Viroli,...) - **Synesthesia** (A. Basso,...)

Deployment on MAVtech Q4T/Q4E DRONES (`https://www.mavtech.eu/`) for precision agriculture – PLANNED
- joint work UniTo - UniBo (M. Viroli,...)

Aggregate Programming: some ongoing/planned work

Deployment of FCPP on MIOSIX (`https://miosix.org/`) – ONGOING (implemented and tested on a network of Wandstem nodes)

- joint work UniTo - PoliMi (W. Fornaciari, F. Terraneo)

Deployment of FCPP on Contiki OS and DecaWave DWM1001 Development Board (`https://www.decawave.com/product/dwm1001-development-board/`) – ONGOING

- joint work UniTo - **REPLY** (M. Griva,...)

Kotlin internal DSL (library) – ONGOING

- joint work UniTo - **REPLY** (M. Griva,...)

AP for Control Theory – ONGOING

- joint work UniTo - University of Iowa (S. Dasgupta,...) - **Raytheon BBN Technologies** (J. Beal,...)

AP as enabling technology for Edge AI – PLANNED

- joint work UniTo - UniBo (M. Viroli,...) - **Synesthesia** (A. Basso,...)

Deployment on MAVtech Q4T/Q4E DRONES (`https://www.mavtech.eu/`) for precision agriculture – PLANNED

- joint work UniTo - UniBo (M. Viroli,...)

A DSL for Civil Enginneers on top of AP – PLANNED

- joint work UniTo - HVL (V. Stolz,...) - Coventry University (Siraj Shaikh,..)

. . .

Aggregate Programming: some ongoing/planned work

Deployment of FCPP on MIOSIX (https://miosix.org/) – ONGOING (implemented and tested on a network of Wandstem nodes)
- joint work UniTo - PoliMi (W. Fornaciari, F. Terraneo)

Deployment of FCPP on Contiki OS and DecaWave DWM1001 Development Board (https://www.decawave.com/product/dwm1001-development-board/) – ONGOING
- joint work UniTo - **REPLY** (M. Griva,...)

Kotlin internal DSL (library) – ONGOING
- joint work UniTo - **REPLY** (M. Griva,...)

AP for Control Theory – ONGOING
- joint work UniTo - University of Iowa (S. Dasgupta,...) - **Raytheon BBN Technologies** (J. Beal,...)

AP as enabling technology for Edge AI – PLANNED
- joint work UniTo - UniBo (M. Viroli,...) - **Synesthesia** (A. Basso,...)

Deployment on MAVtech Q4T/Q4E DRONES (https://www.mavtech.eu/) for precision agriculture – PLANNED
- joint work UniTo - UniBo (M. Viroli,...)

A DSL for Civil Enginneers on top of AP – PLANNED
- joint work UniTo - HVL (V. Stolz,...) - Coventry University (Siraj Shaikh,..)

...

Thanks!

http://www.di.unito.it/~damiani/

https://movere.di.unito.it/ (**System Modelling, Verification and Reuse** research group)

http://www.di.unito.it/ (**Department of Computer Science**)