

Foundations of Composite Event Recognition

Edited by

Alexander Artikis¹, Thomas Eiter², Alessandro Margara³, and Stijn Vansummeren⁴

- 1 University of Piraeus, GR & NCSR Demokritos, GR, a.artikis@iit.demokritos.gr
- 2 TU Wien, AT, eiter@kr.tuwien.ac.at
- 3 Polytechnic University of Milan, IT, alessandro.margara@polimi.it
- 4 Université Libre de Bruxelles, BE, stijn.vansummeren@ulb.ac.be

Abstract

Composite Event Recognition (CER) refers to the activity of detecting patterns in streams of continuously arriving “event” data over, possibly geographically, distributed sources. CER is key in Big Data applications that require the processing of such event streams to obtain timely insights and to implement reactive and proactive measures. Examples include the recognition of emerging stories and trends on the Social Web, traffic and transport incidents in smart cities, and epidemic spread.

Numerous CER languages have been proposed in the literature. While these systems have a common goal, they differ in their data models, pattern languages and processing mechanisms, resulting in heterogeneous implementations with fundamentally different capabilities. Moreover, we lack a common understanding of the trade-offs between expressiveness and complexity, and a theory for comparing the fundamental capabilities of CER systems. As such, CER frameworks are difficult to understand, extend and generalise. It is unclear which of the proposed approaches better meets the requirements of a given application. Furthermore, the lack of foundations makes it hard to leverage established results – from automata theory, temporal logics, etc – thus hindering scientific and technological progress in CER.

The objective of the seminar was to bring together researchers and practitioners working in Databases, Distributed Systems, Automata Theory, Logic and Stream Reasoning; disseminate the recent foundational results across these fields; establish new research collaborations among these fields; thereby start making progress towards formulating such foundations.

Seminar February 9–14, 2007 – <http://www.dagstuhl.de/20071>

2012 ACM Subject Classification Computing methodologies → Knowledge representation and reasoning, Theory of computation, Information systems → Data management systems, Information systems → Data streaming

Keywords and phrases complex event processing, event algebra, pattern matching, stream reasoning, temporal reasoning

Digital Object Identifier 10.4230/DagRep.10.2.19

Edited in cooperation with Elias Alevizos



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 3.0 Unported license

Foundations of Composite Event Recognition, *Dagstuhl Reports*, Vol. 10, Issue 2, pp. 19–49

Editors: Alexander Artikis, Thomas Eiter, Alessandro Margara, and Stijn Vansummeren



DAGSTUHL
REPORTS

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Executive Summary

Alessandro Margara (Polytechnic University of Milan, IT)

Alexander Artikis (University of Piraeus, GR & NCSR Demokritos, GR)

Thomas Eiter (TU Wien, AT)

Stijn Vansummeren (Université Libre de Bruxelles, BE)

License  Creative Commons BY 3.0 Unported license
© Alessandro Margara, Alexander Artikis, Thomas Eiter, and Stijn Vansummeren

This report contains the program and outcomes of Dagstuhl Seminar 20071 on “Foundations of Composite Event Recognition” held at Schloss Dagstuhl, Leibniz Center for Informatics, during February 9-14, 2020.

Composite Event Recognition (CER for short) refers to the activity of detecting patterns in streams of continuously arriving “event” data over, possibly geographically, distributed sources. CER is a key ingredient of many contemporary Big Data applications that require the processing of such event streams in order to obtain timely insights and implement reactive and proactive measures. Examples of such applications include the recognition of attacks in computer network nodes, human activities on video content, emerging stories and trends on the Social Web, traffic and transport incidents in smart cities, error conditions in smart energy grids, violations of maritime regulations, cardiac arrhythmia, and epidemic spread. In each application, CER allows to make sense of streaming data, react accordingly, and prepare for counter-measures.

CER systems become increasingly important as we move from an information economy to an “intelligent economy”, where it is not only the accessibility to information that matters but also the ability to analyse, reason, and act upon information, creating competitive advantage in commercial transactions, enabling sustainable management of communities, and promoting appropriate distribution of social, healthcare, and educational services. Current businesses tend to be unable to make sense of the amounts of data that are generated by the increasing number of distributed data sources that are becoming available daily, and rely more and more on CER. As an example, traffic management in smart cities requires the analysis of data from an increasing number of sensors, both mobile (mounted on public transport vehicles and private cars) and stationary (installed on intersections). Using such data streams, CER may be used to detect or even forecast traffic congestions, thus allowing for proactively changing traffic light policies and speed limits, with the aim of reducing carbon emissions, optimising public transportation, and improving the quality of life and productivity of commuters. As another example, in smart energy grids, streaming information from power grid elements sensors, end-user devices, and diverse other sources such as weather forecasts and event schedules can be combined through CER to improve the grid efficiency and meet the rapidly increasing electricity demand.

Numerous CER systems and languages have been proposed in the literature. While these systems have a common goal, they differ in their architectures, data models, pattern languages, and processing mechanisms, resulting in many heterogeneous implementations with sometimes fundamentally different capabilities. Their comparative assessment is further hindered by the fact that they have been developed in different communities, each bringing in their own terminology and view of the problem.

Moreover, the established CER literature focuses on the practical system aspects of CER. As a result, little work has been done on its formal foundations. Consequently, and in contrast to the situation for more traditional fields in Computer Science, we currently lack a common understanding of the trade-offs between expressiveness and complexity in the design of CER systems, as well as an established theory for comparing their fundamental capabilities.

As such, currently, CER frameworks are difficult to understand, extend and generalise. It is unclear which of the proposed approaches better meets the requirements of a given application domain, in terms of capturing the intended meaning of the composite events of interest, as well as detecting them efficiently. Furthermore, the lack of foundations makes it hard to leverage established results – from automata theory, temporal logics, etc – thus hindering scientific and technological progress in CER.

At the same time, recent years have witnessed increased activities in diverse fields of Computer Science on topics that are related to CER: Inductive and deductive reasoning over streaming data, a field known as Stream Reasoning in Artificial Intelligence. Theoretical complexity results related to processing database queries under updates, associated with advances in Incremental View Maintenance in Database Research. Expressiveness and complexity of logics in the dynamic setting, in Logic research.

The seminar brought together 39 researchers and practitioners working in domains that are strictly related to CER. The first days of the seminar mainly focused on tutorials and talks that gave an overview of the approaches, techniques, methodologies, and vocabularies used in different communities to refer to CER problems. In particular, the following tutorials were presented:

- Applications and requirements for CER
- CER in data management
- CER in distributed event-based systems
- Stream reasoning
- CER in logic and AI
- CER in business process management

The seminar continued by alternating sessions with focused research talks and group discussions on the following topics, that the participants identified as the most relevant for future investigations and research efforts:

- CER language formalisms
- Towards a common framework for CER expressiveness and complexity
- Evaluation strategies: parallel and distributed processing
- Uncertainty in CER
- Pattern induction and composite event forecasting
- Benchmarking

The final sessions of the seminar focused on reporting the results of the group discussions and in planning follow-up activities, including co-organized workshops and events, joint publications, and projects.

2 Table of Contents

Executive Summary

Alessandro Margara, Alexander Artikis, Thomas Eiter, and Stijn Vansummeren . . . 20

Overview of Talks


Complex Event Forecasting <i>Elias Alevizos</i>	24
Stream Logic <i>François Bry</i>	24
Complex Event Recognition in Logic and AI: A Tutorial <i>Diego Calvanese</i>	24
Whole-system Provenance and Composite Event Recognition <i>David Eyers</i>	25
Distributed Data Streaming and the Power of Geometry <i>Minos Garofalakis</i>	25
Towards streaming evaluation of queries with correlation in complex event processing <i>Alejandro J. Grez</i>	26
Event Stream Processing with BeepBeep <i>Sylvain Hallé</i>	26
Probabilistic and Predictive Stream Reasoning <i>Fredrik Heintz</i>	27
Stream Reasoning: A Tutorial <i>Fredrik Heintz</i>	27
Semantic Stream Reasoning For Online Visual Sensor Fusion <i>Danh Le Phuoc</i>	27
Distributed Event-Based Systems: A Tutorial <i>Ruben Mayer and Avigdor Gal</i>	28
Streaming Graph Partitioning <i>Ruben Mayer</i>	28
Interval Temporal Logic <i>Angelo Montanari</i>	29
Modular Materialisation and Incremental Reasoning <i>Boris Motik</i>	29
Trade-offs in Static and Dynamic Evaluation of Hierarchical Queries <i>Dan Olteanu</i>	30
Instance Trees: A data structure for complex logical expressions <i>Thomas Prokosch</i>	30
Elevating the Edge to be a Peer of the Cloud <i>Umakishore Ramachandran</i>	30
Time-sensitive Complex Event Processing <i>Kurt Rothermel</i>	32

Applications & Requirements of CER: A Tutorial <i>Sabri Skhiri</i>	32
CER in Data Management: A Tutorial <i>Martin Ugarte and Cristian Riveros</i>	33
Complex Event Recognition in Business Process Management: A Tutorial <i>Matthias Weidlich</i>	33
Short Introduction to Dynamic Complexity Theory <i>Thomas Zeume</i>	34
Working groups	
Pattern induction and composite event forecasting <i>Daniele Dell’Aglío</i>	34
Process strategies, parallelization and geo-distribution <i>Daniele Dell’Aglío</i>	36
Expressiveness, Compositionality & Hierarchies, and Common Framework <i>Boris Motik and Martin Ugarte</i>	37
Benchmarking <i>Riccardo Tommasini</i>	44
Uncertainty in Complex Event Recognition <i>Han van der Aa, Avigdor Gal, Sylvain Hallé, Annika M. Hinze, and Holger Ziekow</i>	47
Participants	49

3 Overview of Talks

3.1 Complex Event Forecasting

Elias Alevizos (NCSR Demokritos – Athens, GR)

License  Creative Commons BY 3.0 Unported license
© Elias Alevizos


Complex Event Processing (CEP) systems have appeared in abundance during the last two decades. Their purpose is to detect in real-time interesting patterns upon a stream of events and to inform an analyst for the occurrence of such patterns in a timely manner. However, there is a lack of methods for forecasting when a pattern might occur before such an occurrence is actually detected by a CEP engine. We present Wayeb, a framework that attempts to address the issue of Complex Event Forecasting. Wayeb employs symbolic automata as a computational model for pattern detection and variable-order Markov models for deriving a probabilistic description of a symbolic automaton.

References

- 1 Elias Alevizos, Alexander Artikis, Georgios Paliouras. *Event Forecasting with Pattern Markov Chains*. DEBS, 2017.
- 2 Elias Alevizos, Alexander Artikis, Georgios Paliouras. *Wayeb: a Tool for Complex Event Forecasting*. LPAR, 2018.

3.2 Stream Logic

François Bry (LMU München, DE)

License  Creative Commons BY 3.0 Unported license
© François Bry

Joint work of François Bry, Thomas Prokosch

Stream Logic is an attempted formalisation of data streams in predicate logic aimed at enhancing logic programming with streams of (possibly complex) events. The talk motivates Stream Logic with applications, sketches a syntax and a semantics based on a model theory, and relates Stream Logic to meta-programming and non-well-founded sets.

3.3 Complex Event Recognition in Logic and AI: A Tutorial

Diego Calvanese (Free University of Bozen-Bolzano, IT)

License  Creative Commons BY 3.0 Unported license
© Diego Calvanese

In this tutorial we discuss different frameworks, formalisms, and languages, that have been developed within the communities of knowledge representation and reasoning, formal verification, and also database theory and that are in one way or another relevant for the area of Complex Event Recognition (CER). Such formalisms typically rely on combining variants of temporal logics with logics used in knowledge representation and reasoning, and such combination poses challenges with respect to both semantics and computability. The challenges have been addressed by adopting a variety of techniques and by making various

assumptions, but the area is still very fragmented and there is no unifying or consolidated framework. The aim of the presentation is to identify opportunities for collaboration and for cross-fertilization with the CER community.

3.4 Whole-system Provenance and Composite Event Recognition

David Eyers (University of Otago, NZ)

License © Creative Commons BY 3.0 Unported license
© David Eyers

Joint work of Thomas F. J.-M. Pasquier, Jatinder Singh, David M. Eyers, Jean Bacon

Main reference Thomas F. J.-M. Pasquier, Jatinder Singh, David M. Eyers, Jean Bacon: “Camflow: Managed Data-Sharing for Cloud Services”, *IEEE Trans. Cloud Computing*, Vol. 5(3), pp. 472–484, 2017.

URL <https://doi.org/10.1109/TCC.2015.2489211>

Whole-system provenance is becoming increasingly practical as a means to track and audit the way in which data travels throughout computer systems. For example, Pasquier’s CamFlow provenance system is implemented as a Linux Security Module so as to facilitate collecting fine-grained provenance data across both kernel and user-space. However, many forms of provenance query risk rapidly generating unmanageably large volumes of logging information, much of which is typically of little value. The CamQuery extension to CamFlow provides means to facilitate run-time, in-kernel data filtering, involving the detection of patterns of interest within the graph data that is generated during provenance tracking. CamQuery already achieves some forms of composite event recognition (CER), but there is significant potential to integrate CER approaches more directly into CamQuery, to ease querying whole-system provenance.

3.5 Distributed Data Streaming and the Power of Geometry

Minos Garofalakis (Technical University of Crete – Chania, GR)

License © Creative Commons BY 3.0 Unported license
© Minos Garofalakis

Effective Big Data analytics pose several difficult challenges for modern data management architectures. One key such challenge arises from the naturally streaming nature of big data, which mandates efficient algorithms for querying and analyzing massive, continuous data streams (that is, data that is seen only once and in a fixed order) with limited memory and CPU-time resources. In addition to memory- and time-efficiency concerns, the inherently distributed nature of such applications also raises important communication-efficiency issues, making it critical to carefully optimize the use of the underlying network infrastructure. In this talk, we introduce the distributed data streaming model, and discuss techniques for tracking complex queries over distributed streams that rely on novel insights from convex geometry. We also outline possible research directions in this space.

3.6 Towards streaming evaluation of queries with correlation in complex event processing

Alejandro J. Grez (PUC – Santiago de Chile, CL)

License © Creative Commons BY 3.0 Unported license
© Alejandro J. Grez

Joint work of Alejandro J. Grez, Riveros, Cristian

Main reference Alejandro Grez, Cristian Riveros, Martín Ugarte: “A Formal Framework for Complex Event Processing”, in Proc. of the 22nd International Conference on Database Theory, ICDT 2019, March 26-28, 2019, Lisbon, Portugal, LIPIcs, Vol. 127, pp. 5:1–5:18, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.

URL <https://doi.org/10.4230/LIPIcs.ICDT.2019.5>

Complex event processing (CEP) has gained a lot of attention for evaluating complex patterns over high-throughput data streams. Recently, new algorithms for the evaluation of CEP patterns have emerged with strong guarantees of efficiency, i.e. constant update-time per tuple and constant-delay enumeration. Unfortunately, these techniques are restricted for patterns with local filters, limiting the possibility of using joins for correlating the data of events that are far apart.

In this work, we embark on the search for efficient evaluation algorithms of CEP patterns with joins. We start by formalizing the so-called partition-by operator, a standard operator in data stream management systems to correlate contiguous events on streams. Although this operator is a restricted version of a join query, we show that partition-by (without iteration) is equally expressive as hierarchical queries, the biggest class of full conjunctive queries that can be evaluated with constant update-time and constant-delay enumeration over streams. To evaluate queries with partition-by we introduce an automata model, called chain complex event automata (chain-CEA), an extension of complex event automata that can compare data values by using equalities and disequalities. We show that this model admits determinization and is expressive enough to capture queries with partition-by. More importantly, we provide an algorithm with constant update time and constant delay enumeration for evaluating any query definable by chain-CEA, showing that all CEP queries with partition-by can be evaluated with these strong guarantees of efficiency.

3.7 Event Stream Processing with BeepBeep

Sylvain Hallé (University of Quebec at Chicoutimi, CA)

License © Creative Commons BY 3.0 Unported license
© Sylvain Hallé

Main reference Sylvain Hallé: “Event Stream Processing with BeepBeep 3: Log Crunching and Analysis Made Easy”. Presses de l’Université du Québec, ISBN 978-2-7605-5101-5, 332 pages, 2018.


BeepBeep is simple general purpose event stream processing library. This talk will give a short introduction to the system, and highlight some of its distinguishing features. BeepBeep is based on the concept of *processors*, which are simple, stateful units of computation that can be composed to perform complex processing chains. These chains are created using Java code, but can usually be represented graphically using standardized pictograms, as in Figure 1.

In particular, in this presentation we have shown how it is possible to write Domain-Specific Languages in a few lines of code, and how BeepBeep integrates some elements of explainability and traceability for its computed results.

processing pipeline like object tracking has to hard-wire an engineered set of DNN models to a fixed processing logic. To remedy this problem, we propose a novel semantic reasoning approach that uses stream reasoning programmes for representing commonsense and domain knowledge using non-monotonic rules in Answer Set Programming (ASP) where uncertainty of probabilistic inference operations is incorporated by weights. Our approach is realised by a dynamic reasoning framework which enables probabilistic planning to adapt the sensor fusion pipeline under operational constraints expressed in ASP. Via this talk, we will share our current implementation experience and experiment results together with our visions towards open challenges on this research direction.

3.11 Distributed Event-Based Systems: A Tutorial


Ruben Mayer (TU München, DE) and Avigdor Gal (Technion – Haifa, IL)

License  Creative Commons BY 3.0 Unported license
© Ruben Mayer and Avigdor Gal

Distributed event-based systems offer a well-established way to gain high-level insights from low-level streaming data in real-time. These systems may come in different flavors and stem from different communities, yet they share common principles and concepts. In this tutorial, we overview these common concepts. In addition, we focus on the concept of windowing, the notion of time and the trade-off between latency and accuracy.

3.12 Streaming Graph Partitioning

Ruben Mayer (TU München, DE)

License  Creative Commons BY 3.0 Unported license
© Ruben Mayer

Joint work of Ruben Mayer, Kamil Orujzade, Hans-Arno Jacobsen
Main reference Ruben Mayer, Kamil Orujzade, Hans-Arno Jacobsen: “2PS: High-Quality Edge Partitioning with Two-Phase Streaming”, CoRR, Vol. abs/2001.07086, 2020.
URL <https://arxiv.org/abs/2001.07086>

Graph partitioning is an important preprocessing step to distributed graph processing. In edge partitioning, the edge set of a given graph is split into k equally-sized partitions, such that the replication of vertices across partitions is minimized. Streaming is a viable approach to partition graphs that exceed the memory capacities of a single server. The graph is ingested as a stream of edges, and one edge at a time is immediately and irrevocably assigned to a partition based on a scoring function. However, streaming partitioning suffers from the uninformed assignment problem: At the time of partitioning early edges in the stream, there is no information available about the rest of the edges. As a consequence, edge assignments are often driven by balancing considerations, and the achieved replication factor is comparably high. In this paper, we propose 2PS, a novel two-phase streaming algorithm for high-quality edge partitioning. In the first phase, vertices are separated into clusters by a lightweight streaming clustering algorithm. In the second phase, the graph is re-streamed and edge partitioning is performed while taking into account the clustering of the vertices from the first phase. Our evaluations show that 2PS can achieve a replication factor that is comparable to heavy-weight random access partitioners while inducing orders of magnitude lower memory overhead.

3.13 Interval Temporal Logic

Angelo Montanari (University of Udine, IT)

License © Creative Commons BY 3.0 Unported license
© Angelo Montanari

Joint work of Laura Bozzelli, Adriano Peron, Pietro Sala, and many others

In the talk, I give a gentle introduction to interval temporal logic. I start with a short account of its distinctive features as well as of interval modalities. Then, I present the general picture of the satisfiability and model checking problems for interval temporal logic. Links to more detailed presentations are provided for interested people. Next, I briefly compare the expressiveness of interval temporal logic (in model checking) with that of LTL, CTL, and CTL*, and describe a generalization of the proposed model checking framework with regular expressions. In the last part of the talk, I outline recent and ongoing research work. In particular, I introduce and briefly compare interval temporal logics of prefixes, suffixes, and infixes, suggest possible ways of going beyond finite Kripke structures in model checking, and illustrate the problem of model checking a single interval model. I conclude the talk by discussing the appropriateness of interval temporal logic for composite event recognition.

3.14 Modular Materialisation and Incremental Reasoning

Boris Motik (University of Oxford, GB)

License © Creative Commons BY 3.0 Unported license
© Boris Motik

Maintenance of materialisation of Datalog programs plays a key role in many applications of stream reasoning. In our recent work in the KRR group at Oxford University, we have developed and thoroughly evaluated a number of algorithms that can efficiently address this problem on a range of Datalog programs commonly found in practice. Despite this progress, certain programs can still be hard for incremental materialisation; for example, programs containing rules that axiomatise a binary predicate as transitive can be hard.

In this talk, I will present an outline of some of our recent work on modular materialisation and incremental maintenance of Datalog programs. We combine standard seminaive Datalog evaluation with custom modules that implement the semantics of a program subset in an arbitrary (presumably more efficient) way. We thus obtain a general framework that can integrate specialised algorithms (e.g., algorithms for the maintenance of transitive closure) with general Datalog reasoning. I will present the results of a performance evaluation showing the benefits of such a hybrid approach.

3.15 Trade-offs in Static and Dynamic Evaluation of Hierarchical Queries

Dan Olteanu (University of Oxford, GB)

License © Creative Commons BY 3.0 Unported license
© Dan Olteanu

Joint work of Dan Olteanu, Ahmet Kara, Milos Nikolic, Haozhe Zhang

In this talk I will discuss trade-offs in static and dynamic evaluation of hierarchical queries with arbitrary free variables. In the static setting, the trade-off is between the time to partially compute the query result and the delay needed to enumerate its tuples. In the dynamic setting, I also consider the time needed to update the query result in the presence of single-tuple inserts and deletes to the input database.

I put forward one evaluation approach that unifies both settings. This approach observes the degree of values in the database and uses different computation and maintenance strategies for high-degree and low-degree values. For the latter it partially computes the result, while for the former it computes enough information to allow for on-the-fly enumeration.

The main result of this work defines the preprocessing time, the update time, and the enumeration delay as functions of the light/heavy threshold and of the factorization width of the hierarchical query. By conveniently choosing this threshold, the approach can recover a number of prior results when restricted to hierarchical queries.

3.16 Instance Trees: A data structure for complex logical expressions

Thomas Prokosch (LMU München, DE)

License © Creative Commons BY 3.0 Unported license
© Thomas Prokosch

Complex-event recognition relies upon storing and retrieving complex expressions (representing events) in a time- and space-efficient manner. This presentation introduces ongoing research on such a data structure: Instance Trees. The presentation first motivates, then sketches the concepts of this data structure.

3.17 Elevating the Edge to be a Peer of the Cloud

Umakishore Ramachandran (Georgia Institute of Technology – Atlanta, US)

License © Creative Commons BY 3.0 Unported license
© Umakishore Ramachandran

Joint work of Umakishore Ramachandran, Harshit Gupta, Adam Hall, Enrique Saurez, Zhuangdi Xu
Main reference Umakishore Ramachandran, Harshit Gupta, Adam Hall, Enrique Saurez, Zhuangdi Xu: “Elevating the Edge to Be a Peer of the Cloud”, in Proc. of the 12th IEEE International Conference on Cloud Computing, CLOUD 2019, Milan, Italy, July 8-13, 2019, pp. 17–24, IEEE, 2019.
URL <https://doi.org/10.1109/CLOUD.2019.00016>

Technological forces and novel applications are the drivers that move the needle in systems and networking research, both of which have reached an inflection point. On the technology side, there is a proliferation of sensors in the spaces in which humans live that become more intelligent with each new generation. This opens immense possibilities to harness the potential of inherently distributed multimodal networked sensor platforms (aka Internet of Things – IoT platforms) for societal benefits. On the application side, large-scale situation

awareness applications (spanning healthcare, transportation, disaster recovery, and the like) are envisioned to utilize these platforms to convert sensed information into actionable knowledge. The sensors produce data 24/7. Sending such streams to the cloud for processing is sub-optimal for several reasons. First, often there may not be any actionable knowledge in the data streams (e.g., no action in front of a camera), wasting limited backhaul bandwidth to the core network. Second, there is usually a tight bound on latency between sensing and actuation to ensure timely response for situation awareness. Lastly, there may be other non-technical reasons, including sensitivity for the collected data leaving the locale. Sensor sources themselves are increasingly becoming mobile (e.g., self-driving cars). This suggests that provisioning application components that process sensor streams cannot be statically determined but may have to occur dynamically.

All the above reasons suggest that processing should take place in a geo-distributed manner near the sensors. Fog/Edge computing envisions extending the utility computing model of the cloud to the edge of the network. We go further and assert that the edge should become a peer of the cloud. This talk is aimed at identifying the challenges in accomplishing the seamless integration of the edge with the cloud as peers. Specifically, we want to raise questions pertaining to (a) frameworks (NOSQL databases, pub/sub systems, distributed programming idioms) for facilitating the composition of complex latency sensitive applications at the edge; (b) geo-distributed data replication and consistency models commensurate with network heterogeneity while being resilient to coordinated power failures; and (c) support for rapid dynamic deployment of application components, multi-tenancy, and elasticity while recognizing that both computational, networking, and storage resources are limited at the edge.

References

- 1 Zhuangdi Xu, Sayan Sinha, Harshil Shah, and Umakishore Ramachandran. *Space-Time Vehicle Tracking at the Edge of the Network*. ACM Mobicom Workshop on Hot Topics in Video Analytics and Intelligent Edges (HotEdgeVideo'19), October 2019, Los Cabos, Mexico.
- 2 Umakishore Ramachandran, Harshit Gupta, Adam Hall, Enrique Saurez Apuy, and Zhuangdi Xu. *Elevating the Edge to be a Peer of the Cloud*. IEEE International Conference on Cloud Computing (CLOUD 2019), July 9-12, 2019, Milano, Italy.
- 3 Adam Hall and Umakishore Ramachandran. *An Execution Model for Serverless Functions at the Edge*. ACM/IEEE IoT Design and Implementation (IoTDI), April 16-18, 2019, Montreal, Canada.
- 4 Harshit Gupta, Zhuangdi Xu, and Umakishore Ramachandran. *DataFog: Towards a Holistic Data Management Platform for the IoT Age at the Network Edge*. USENIX Workshop on Hot Topics in Edge Computing, HotEdge 2018, , Boston, MA, July 10, 2018.
- 5 Harshit Gupta and Umakishore Ramachandran. *FogStore: A Geo-Distributed Key-Value Store Guaranteeing Low Latency for Strongly Consistent Access*. Proceedings of the 12th ACM International Conference on Distributed and Event-based Systems (DEBS '18) , June 2018, Hamilton, New Zealand.
- 6 Zhuangdi Xu, Harshit Gupta, and Umakishore Ramachandran. *STTR: A System for Tracking All Vehicles All the Time At the Edge of the Network*. Proceedings of the 12th ACM International Conference on Distributed and Event-based Systems (DEBS '18), June 2018, Hamilton, New Zealand.
- 7 Enrique Saurez, Kirak Hong, Dave Lillethun, Beate Ottenwaelder, Umakishore Ramachandran. *Incremental Deployment and Migration of Geo-Distributed Situation Awareness Applications in the Fog*. ACM DEBS '16, June 20 – 24, 2016, Irvine, CA, USA. Note: winner of the “Honorable mention” award at DEBS 2016.

3.18 Time-sensitive Complex Event Processing

Kurt Rothermel (Universität Stuttgart, DE)

License  Creative Commons BY 3.0 Unported license
© Kurt Rothermel

For many CEP applications, a limited end-to-end latency is of paramount importance. Buffering of events in front of operators is a major source of end-to-end latency. An increasing load may cause buffers to fill up rapidly leading to higher end-to-end latencies.


One approach to reduce latency is to allocate more compute resources for the operators in the CEP network. A promising way to do that is data parallelism, i.e. the input streams of an operator are partitioned and each partition is executed by a single operator instance in parallel with the other partitions. The challenges associated with this approach are manifold, such as appropriate models to predict overload, to decide where to allocate additional resources in the network of operators, or to determine how much resources should be added. Similar problem arises when the load is decreasing and resources can be deallocated.

Another approach to reduce the end-to-end latency is load shedding. This is the only alternative if resources are scarce (e.g., mobile devices or fog) or the monetary budget limits the available compute resources. The goal is to shed no more than needed to meet the given latency bound. Moreover, shedding should be done in a way so that the quality of CEP processing suffers least. Load shedding in CEP is associated with various interesting questions related to “Where to shed, how much to shed and what to shed” (e.g., which events or partial matches).

In the talk, we will report about some of our research results and ongoing work in this field.

3.19 Applications & Requirements of CER: A Tutorial

Sabri Skhiri (EURA NOVA – Mont-Saint-Guibert, BE)

License  Creative Commons BY 3.0 Unported license
© Sabri Skhiri

The CER/CEP have been on the market for more than 10 years. However, we have seen the last five years the emergence of a new class of real time use cases. In these use cases, we have seen a significant increase of the event throughput, a need for expressing new queries, and a need to make these CER usable and manageable in operations.

The aim of this tutorial is to give an overview of this new generation of use cases while answering these questions: (1) Which are the application domains of CER? (2) What are the key requirements of CER concerning data models, recognition language expressiveness, performance (latency, throughput, predictive accuracy)? (3) How do existing approaches address these requirements? (4) What are the classes of applications that can take advantage of CER? We answer these questions by first describing the typical Streaming architecture where CER are deployed. Then, we illustrate these challenges within Industrial use cases in crowd management, banking, Telecom, Security & Surveillance and finally SOA & microservice architecture. From these cases, we summarise the key requirements and the opportunities for contributions in CER/CEP. Finally, in order to discuss the open challenges in research, we start from the open challenges in Stream processing and we project them on CER/CEP. Interestingly, the same challenges apply but in a completely different manner.

3.20 CER in Data Management: A Tutorial

Martin Ugarte (Millenium Institute – Santiago de Chile, CL) and Cristian Riveros (PUC – Santiago de Chile, CL)

- License** © Creative Commons BY 3.0 Unported license
© Martin Ugarte and Cristian Riveros
- Joint work of** Grez, Alejandro; Vansummeren, Stijn
- Main reference** Alejandro Grez, Cristian Riveros, Martín Ugarte: “A Formal Framework for Complex Event Processing”, in Proc. of the 22nd International Conference on Database Theory, ICDT 2019, March 26-28, 2019, Lisbon, Portugal, LIPIcs, Vol. 127, pp. 5:1–5:18, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.
- URL** <http://dx.doi.org/10.4230/LIPIcs.ICDT.2019.5>
- Main reference** Martín Ugarte, Stijn Vansummeren: “On the Difference between Complex Event Processing and Dynamic Query Evaluation”, in Proc. of the 12th Alberto Mendelzon International Workshop on Foundations of Data Management, Cali, Colombia, May 21-25, 2018, CEUR Workshop Proceedings, Vol. 2100, CEUR-WS.org, 2018.
- URL** <http://ceur-ws.org/Vol-2100/paper13.pdf>
- Main reference** Muhammad Idris, Martín Ugarte, Stijn Vansummeren, Hannes Voigt, Wolfgang Lehner: “Efficient Query Processing for Dynamically Changing Datasets”, SIGMOD Rec., Vol. 48(1), pp. 33–40, 2019.
- URL** <https://doi.org/10.1145/3371316.3371325>

Composite Event Recognition (CER) has emerged as the unifying field for technologies that require processing and correlating distributed data sources in real-time. CER finds applications in diverse domains, which has resulted in a large number of proposals for expressing and processing complex events. In this context, the objective of the tutorial is to give a theoretical perspective of the most common features found in CER. We will start by presenting a basic setting for CER that will serve to discuss what are the fundamental properties that, from a Data-Management perspective, could be asked from a CER language: well-defined syntax and semantics, composability, and denotational declarative semantics. These properties will then be exemplified by means of a particular language called CEL, which will also be used to present the main operations found in CER systems. We will discuss what are the challenges associated to defining these operations formally while satisfying the mentioned properties. Having a principled perspective on CER languages, we will move to the problem of evaluating these languages. We will discuss what are the relevant notions of efficiency and complexity, to then present the kind of lower bounds that can be obtained for evaluating CER patterns. Finally, we will show particular examples that will serve to introduce fundamental open problems.

3.21 Complex Event Recognition in Business Process Management: A Tutorial


Matthias Weidlich (HU Berlin, DE)

- License** © Creative Commons BY 3.0 Unported license
© Matthias Weidlich

Business processes represent consumers as well as producers of events in many application scenarios. Common process modelling languages, therefore, include constructs to incorporate events. At the same time, event-based systems may be used as a basis for process execution and the analysis of processes. Against this background, the tutorial reviews the relation between the fields of business process management and complex event recognition. In particular, opportunities for research at the intersection of the two fields are outlined.

3.22 Short Introduction to Dynamic Complexity Theory

Thomas Zeume (TU Dortmund, DE)

License  Creative Commons BY 3.0 Unported license
© Thomas Zeume

Joint work of Samir Datta, Anish Mukherjee, Nils Vortmeier, Thomas Schwentick
Main reference Samir Datta, Anish Mukherjee, Nils Vortmeier, Thomas Zeume: “Reachability and Distances under Multiple Changes”, in Proc. of the 45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic, LIPIcs, Vol. 107, pp. 120:1–120:14, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018.
URL <http://dx.doi.org/10.4230/LIPIcs.ICALP.2018.120>
Main reference Samir Datta, Raghav Kulkarni, Anish Mukherjee, Thomas Schwentick, Thomas Zeume: “Reachability Is in DynFO”, J. ACM, Vol. 65(5), pp. 33:1–33:24, 2018.
URL <http://dx.doi.org/10.1145/3212685>

Dynamic descriptive complexity theory studies how query results can be updated in a highly parallel fashion, that is, by constant-depth circuits or, equivalently, by first-order formulas, or by the relational algebra. After gently introducing dynamic complexity theory, I will discuss recent results regarding the dynamic complexity of the reachability query.

4 Working groups

4.1 Pattern induction and composite event forecasting

Daniele Dell’Aglio (Universität Zürich, CH)

License  Creative Commons BY 3.0 Unported license
© Daniele Dell’Aglio

Working group participants:

- Han van der Aa (HU Berlin, DE).
- Alexander Artikis (University of Piraeus, GR & NCSR Demokritos, GR).
- François Bry (LMU München, DE).
- Daniele Dell’Aglio (Universität Zürich, CH).
- Emanuele Della Valle (Polytechnic University of Milan, IT).
- Avigdor Gal (Technion – Haifa, IL).
- Minos Garofalakis (TU Crete, GR).
- Fredrik Heintz (Linköping University, SE).
- Annika M. Hinze (University of Waikato, NZ).
- Kurt Rothermel (Universität Stuttgart, DE).
- Matthias Weidlich (HU Berlin, DE).
- Holger Ziekow (HFU – Furtwangen, DE).

Machine learning is a powerful framework to process and analyse temporal and dynamic data and can find application in composite event recognition as well. In this session we focused on two tasks. The first is composite event specification learning: what are the rules that define relevant composite events? The second is composite event forecasting: what are the next events (or composite events) that will appear from the stream? We identified in hybrid intelligence and explainability two of the main benefits that machine learning and CER can lead when combined.

Defining composite events is a challenging task, which requires domain knowledge, as well as an in-depth understanding of the data itself. As data is increasing in size, variety and heterogeneity over time, the complexity to define the composite events is growing as well.

For example, it is often the case that almost matching composite events are not recognised, as minimal discrepancies between the event definition and the input stream are sufficient to lead to negative responses. In such a scenario, machine learning can provide useful support to domain experts. For example, a human may identify the variables of interest, using machine learning to infer correlations among them. A complementary approach consists in combining human- and machine-defined rules, where the former introduces rules related to the domain knowledge, and the latter can timely infer rules related to data shifts and drifts.

At the same time, composite events represent knowledge which can be interpreted by humans and is a potential solution to achieve explainable machine learning processes. We envision the adoption of composite event formalisms in the scenarios where the input are event streams since composite events allow expressing temporal relations.

Combining machine learning and techniques for composite event recognition also poses challenges for future research. Machine learning processes require a training phase, where they learn regularities and patterns by observing the input data. We traditionally distinguish between offline and online learning, depending on the fact that the training happens before or during the analysis of the stream. This depends on the type of machine learning technique, as well as the data itself. When data shows features that do not vary over time, or that vary regularly, offline learning usually leads to better outcomes. However, when the data is characterised by shifts and drifts, the data used to learn become stale over time, degrading the quality of the analyses over time. In those cases, online learning may be a more suitable solution. Creating online learning techniques, however, is challenging, since the construction of the model may be time-consuming, too slow with regards to the drifts happening in the data, or may require the definition of complex scheduling policies to retrain the algorithms. A relevant dimension to consider when thinking about machine learning and composite event recognition is distribution. In scenarios where data has high throughput, is physically distributed and controlled by different stakeholders, distributed techniques may bring several advantages. Relevant phenomena could be identified on the edge, exploiting, for example, federated learning and function shipping solutions to push decentralisation. This may also lead advantages in terms of privacy, robustness and efficiency. However, the distribution requires coordination and reconciliation mechanisms, leading to overheads and not applicable to every use case.

A final challenge which can stimulate future research is the problem of monitoring. In scenarios like security, smart cities and fraud detection, monitoring the stream leads to changes in the stream itself. For example, knowing the situation of the city during traffic jams can lead drivers to opt for different paths, with the potential risk of generating new traffic jams. Possible ways to tackle this challenge are the design of cost functions that focus on the community, e.g. a navigation system may try to optimize the average speed of all the cars in the city. Another solution may come from game theory through the identification of equilibrium points. Another problem related to monitoring is the evaluation: changes in the input stream make it hard to test and systematically compare alternative solutions. One could follow analytical analyses, or exploit simulators to run experimental tests. However, both approaches may be complex and costly to follow.

4.2 Process strategies, parallelization and geo-distribution

Daniele Dell'Aglio (Universität Zürich, CH)

License  Creative Commons BY 3.0 Unported license
© Daniele Dell'Aglio

Working group participants:

- Daniele Dell'Aglio (Universität Zürich, CH).
- David Eyers (University of Otago, NZ).
- Minos Garofalakis (TU Crete, GR).
- Manfred Hauswirth (Fraunhofer FOKUS – Berlin, DE).
- Alessandro Margara (Polytechnic University of Milan, IT).
- Ruben Mayer (TU München, DE).
- Umakishore Ramachandran (Georgia Institute of Technology – Atlanta, US).
- Till Rohrmann (Ververica – Berlin, DE).
- Kurt Rothermel (Universität Stuttgart, DE).
- Sabri Skhiri (EURA NOVA – Mont-Saint-Guibert, BE).
- Riccardo Tommasini (University of Tartu, EE).

Nowadays, networks are usually complex environment, with nodes heterogeneous in computational power, storage capabilities, network connections, geographical locations and ownership. This is the case of modern edge infrastructures, where networks have powerful central nodes (private or public cloud infrastructures) and nodes with limited resources at the edge.

Moreover, network failures (e.g. congestion and broken nodes) bring dynamics in such networks. Dynamics also happen in specific contexts, such as in automotive, where edge nodes move and affect the network topology, varying the connections among the network nodes. On top of those networks, stakeholders want to detect events of interests, based on either the data collected by a node itself or the data exchanged by other nodes.

To recognize composite events over these heterogeneous and dynamic networks, we need flexible processing frameworks that expose the following features. Such frameworks should be flexible, able to move both the data and the recognizing functions across the network. Flexibility is important for runtime performance, to improve time performance metrics (e.g. latency and throughput), as well for privacy purposes (e.g. process the data locally). Another feature processing frameworks should offer robustness. The frameworks should be able to cope with network failures, to do not stop the execution if a node breaks or connection is congested, by exploiting, for example, redundancy. At the same time, the processing should be robust to the presence of noise and wrong data. Detecting the same composite events with data observed by different nodes may offer different perspectives, allowing to detect and isolate wrong results. Finally, such frameworks should be observable, to monitor the execution of the system and potentially detect failures and issues.

To design and build such processing frameworks, we should take into account the existence of features that can hardly co-exist in the same solution. There exists a trade-off between expressiveness and performance: the more expressive the constructs in the programming frameworks, the more complex the business logic, with subsequent loss of performance. There is also a trade-off between generalization and specialization: domain-specific solutions may exploit the context to introduce specific optimizations and solutions, at the price of being usable only in a specific number of use cases. The framework should be able to detect composite events in a continuous fashion, coping with issues in the input streams, such

as noise and out of orders. The detection should also be distributed, with the need for consensus mechanisms to reach agreements on the identification of the events. It is essential to observe that different nodes may have different perceptions of reality, which could be captured through local ontologies.

4.3 Expressiveness, Compositionality & Hierarchies, and Common Framework

Boris Motik (University of Oxford, GB) and Martin Ugarte (Millenium Institute – Santiago de Chile, CL)

License  Creative Commons BY 3.0 Unported license
© Boris Motik and Martin Ugarte

Working group participants:

- Alexander Artikis (University of Piraeus, GR & NCSR Demokritos, GR).
- François Bry (LMU München, DE).
- Emanuele Della Valle (Polytechnic University of Milan, IT).
- Thomas Eiter (TU Wien, AT).
- Alessandro Margara (Polytechnic University of Milan, IT).
- Angelo Montanari (University of Udine, IT).
- Boris Motik (University of Oxford, GB).
- Thomas Prokosch (LMU München, DE).
- Tore Risch (Uppsala University, SE).
- Cristian Riveros (PUC – Santiago de Chile, CL).
- Kostas Stathis (Royal Holloway, University of London, GB).
- Martin Ugarte (Millenium Institute – Santiago de Chile, CL).
- Stijn Vansummeren (Université Libre de Bruxelles, BE).
- Matthias Weidlich (HU Berlin, DE).
- Thomas Zeume (TU Dortmund, DE).

This is a summary of the outcomes of two discussion sessions held during the Dagstuhl 20071 seminar on Foundations of Composite Event Recognition. The topics of the two sessions were quite related, so it is natural to summarise them jointly.

4.3.1 Expressiveness, Compositionality & Hierarchies

The discussion in this session was motivated by an observation that the field of Complex Event Recognition (CER) is very broad and diverse, which makes understanding the relationships between various approaches proposed in the literature quite difficult. The discussion revolved around a number of issues, as summarised next.

CER from an abstract perspective. It was observed that the community has not agreed on a common abstract perspective of the CER problem. The following three possibilities have been proposed and discussed.

- CER is a *model checking problem* – that is, the problem of verifying whether a finite input satisfies a particular property. After a discussion, there was consensus that this perspective most likely does not adequately capture CER.

- CER is a *monitoring problem* – that is, the problem of detecting the instant when monotonically increasing input satisfies a particular property. There was a sentiment that a minority of CER applications may fall into this category.
- CER is a *synthesis problem* – that is, the problem of transforming an infinite input into an infinite output using a predetermined specification. This view was identified as closest to most CER applications.

Providing a common model. It was noted that agreeing on a common model had a tremendous impact in areas such as databases and description logics, and so doing the same in the context of CER might produce similar benefits: it would allow for an easier comparison of the expressiveness and the capabilities of different approaches, both formally and informally, and standardisation might foster interoperability of tools and systems produced by different groups. To achieve these goals, both data and query models should be agreed upon. This raised a question of what should CER systems produce as output, and the following views were put forward.

- An answer is a *sequence of time-annotated facts*.
- An answer is a sequence of *time-annotated sets of tuples*. That is, database queries produce sets of tuples, so by analogy CER systems should produce streams of sets of tuples.
- An answer is a *sequence of time-annotated sets of facts from the input* that constitute a complex event.

Who would use a common model? It was noted that defining a common model was difficult partly because of a lack of clarity about who its intended users would be. The following possibilities were discussed.

- A common model would be used by end-users (i.e., practitioners) in the field. In this view, a common model would play the role analogous to SQL by defining an interface that CER systems would provide. In such a case, an important design guideline would be to produce an intuitive model that closely reflects the end-users' problems.
- A common model would be used mainly by researchers as an agreed-upon yardstick for analysing various approaches from a conceptual and/or theoretical standpoint. Thus, a common model would have to be very expressive and general, whereas its practical applicability would be less important. It was noted that such a model would play a role similar to that of a Turing machine.
- A common model would be used by both end-users and researchers. This would be an ideal outcome, but it might be difficult to attain.

The role of time. There was considerable discussion about the role of time in such a model. The following opinions were voiced.

- Time is *not special*. In this view, time is just another piece of data that may or may not be present in the stream elements. The common model could be just the relational model, where time instants and intervals could be modelled using one or two temporal attributes, respectively.
- Time is *immanent* to CER, and it provides an *orthogonal dimension* to the stream content. That is, the data in the stream can be seen as 'opaque': we just need to be able to manipulate data items using an appropriate algebra. For example, data items can be relational, in which case they are manipulated using the relational algebra; or data items can be XML documents, in which case they are manipulated using XPath and XQuery. A CER system can be built on top of any data model by considering a time-annotated sequence of data items. The CER system should provide constructs for manipulating the

temporal component of the stream, which can be integrated with the underlying algebra in a modular fashion. It was observed that this underpins the CQL approach by Widom et al.

Time instants vs. time intervals. There was a long discussion about whether a common model for CER should be based on time instants or time intervals. The following arguments were put forward in favour of each view.

- In the former view, input events are instantaneous occurrences on a discrete, partially ordered timeline. Complex events can be thought of as patterns in the input, and they are also instantaneous in the sense that they occur at instants at which the patterns are recognized. In this way, there is no distinction between input events and complex events, so the latter can be used as input to create event hierarchies. It was argued that such a viewpoint is appropriate for CER because sensor readings are instantaneous and the observed value between two successive readings is unknown. Finally, it was argued that duration of an event can be defined as the time period between the event's onset and cessation. For example, one can identify the time points at which the 'Fire is detected' event commences and stops, which gives the duration of the event.
- The latter view assumes that events are inherently durative in nature, so a CER system should have the notion of a duration built into it. It was pointed out that this view is more general than the time point view, and it was argued that durations are strictly necessary for temporal aggregation. Finally, it was argued that an interval view might provide a more natural user interface to a CER system.

Proposals for a common model. Several different formalisms were proposed as candidates for a common model.

- Relational model seemed to be a natural candidate. It was argued that no specific treatment of time would be needed in such a setting as temporal information could be encoded using one (for point-based events) or two (for interval-based events) temporal attributes.
- First-order logic could be used analogously, and it could be extended with second-order features such as Kleene closure.
- Various temporal logics (either point- or interval-based) were also put forward as natural candidates.

A key question was what kind of object should be produced by query evaluation. One view was that queries can be seen as formulas with free variables, so answers are variable substitutions that make the formula true.

A common model is not needed. It was also argued that agreeing on a common model for CER might be too difficult or even undesirable. The argument was that the model should be chosen to fit the application at hand; however, the application space seems to be too heterogenous to be unified, so it might be difficult to come up with a one-size-fits-all solution. Instead, different approaches should be compared directly – for example, by providing pairwise translations between approaches.

4.3.2 Common Framework

As the discussion in the 'Expressiveness, Compositionality, & Hierarchies' session revealed, reaching agreement in the community on a single common model for CER might be too ambitious at this point. Therefore, the discussion in the 'Common Framework' session explored the possibility of providing a common *meta-model* for CER. The objective was

to present a more abstract view of CER that would clearly identify key elements of most CER systems. Specifically, this meta-model would specify what a CER system *does*, without focussing on *how* this is done. The meta-model would not necessarily focus on a specific language for encoding events and queries; rather, it would provide a list of conceptual components that could be instantiated in different ways. The main benefit of such a meta-model would be to provide a common way of thinking and talking about CER, which would make discussions in the community easier.

The rest of this section summarises the meta-model that was proposed and discussed at the seminar.

4.3.2.1 Abstract View of CER

Seen from an abstract point of view, a CER system observes an ever-increasing amount of information about an application environment. This information is delivered to the CER system in discrete, finite chunks called *updates*. Thus, at each instant i , a CER system has observed a finite number of such updates. The task of a CER system is to extract useful information from all information observed up to a given instant. More precisely, a CER system should exhibit behaviour that looks as if the following steps were performed at each instant i .

1. All observed updates are *combined* into a *world view* W_i . The role of W_i is to reflect the history (or, in some cases, the relevant part of the history) of the application environment. Often, this step will involve *background knowledge* K about the environment that, for all intents and purposes, can be assumed to be immutable.
2. A fixed *query* Q is evaluated over W_i , and the answer A_i is sent to the user at time instant i . The job of Q is to specify which part of the world view is relevant for the application at hand. In other words, Q is a function that extracts useful information from W_i .

This idea can be summarised formally as follows. A CER system is parameterised by immutable background knowledge K , an aggregation function AGG , and a query Q . Function AGG must be applicable to K and a finite sequence of *updates* $U_1.U_2 \dots U_i$, and query Q is a function that must be applicable to the aggregation result. Then, given an infinite *stream* $S = U_1.U_2 \dots$ of updates, the job of a CER system is to produce the infinite sequence of answers $A_1.A_2 \dots$ where, for each $i \geq 1$, we have $A_i = Q(W_i)$ for $W_i = \text{AGG}(K, U_1.U_2 \dots U_i)$.

It is crucial to understand that this specification *does not* mandate that the system necessarily has to compute W_i and then evaluate Q on W_i at each instant i . Rather, this specification only specifies what the observable behaviour of the system should be, and the system is free to choose any appropriate implementation/evaluation strategy. In fact, it is usually reasonable to introduce various assumptions about the properties of the computation that a CER system should use. We call such assumptions *nonfunctional requirements* in order to stress that these specify certain aspects of a system's operation, rather than restrict the answers that the system must compute. For example, a common nonfunctional requirement might be that a CER system should use a bounded amount of memory during its operation. Such requirements will determine whether a CER system can correctly process the query Q or not.

It is useful to further assume that background knowledge, updates, and queries are all expressed in appropriate *knowledge*, *update*, and *query languages* \mathcal{L}_K , \mathcal{L}_U , and \mathcal{L}_Q , respectively. Formally, these languages can be seen as classes whose members constitute legal values for K , U_i , and Q . In some cases it might be useful to restrict not just the form of each update, but also to place certain *structural constraints* on the stream itself. For example, a

structural constraint might be ‘The time stamps cannot be decreasing’. Then, the objective of CER research can be framed as the task of studying the algorithmic methods that produce the correct answers for specific combinations of nonfunctional requirements, languages \mathcal{L}_K , \mathcal{L}_U , and \mathcal{L}_Q , and structural constraints on the stream.

4.3.2.2 Examples

To clarify these ideas, this section presents several very simple instantiations of the CER meta-model.

► **Example 1.** Let the update language \mathcal{L}_U consist of all finite sets of relational facts of the form $R(a_1, \dots, a_n, t)$, where R is a relation, all a_j are constants, and t is a time point. In other words, each update U_i is a finite set of relational facts with a time stamp. Moreover, ignoring any question of background knowledge for the moment, let the aggregation function be defined by

$$\text{AGG}(U_1.U_2 \dots U_i) = \{\text{Now}(i)\} \cup \bigcup_{j=1}^i U_j. \quad (1)$$

In other words, **AGG** takes the union of all updates, but it also introduces a fact that represents the current time point. Each world view W_i thus contains all information that the CER system observed up to instant i , as well as information about where in the stream we are.

As an example, consider the stream that consists of the following updates:

$$U_1 = \{\text{temp}(\text{burner}_1, 40, 1)\} \quad (2)$$

$$U_2 = \{\text{temp}(\text{burner}_2, 20, 2)\} \quad (3)$$

$$U_3 = \{\text{temp}(\text{burner}_1, 60, 2), \text{temp}(\text{burner}_2, 45, 3)\} \quad (4)$$

Intuitively, each update represents temperature readings of gas burners, where each reading is associated with an instant at which the reading has been produced. Note that the instants inside the updates do not necessarily correspond to the instants at which an update has been received. For example, update U_3 is received at instant 3, but fact $\text{temp}(\text{burner}_1, 60, 2)$ refers to time instant 2. We next investigate languages for querying W_i .

First-order logic provides the formal foundations for a substantial part of SQL, so it is reasonable to try to use it in a streaming setting. Thus, let us define queries over W_i as domain-independent first-order formulas with free variables; moreover, we define an answer to a query on W_i as the set of all substitutions of the free variables that make the query true on W_i . Such a language is very expressive, and in particular it allows us to ask questions about both past and present instants. For example, the following query identifies the most recent time instant after which the temperature reading for burner_1 was above 35 for two consecutive instants:

$$\begin{aligned} Q(t) = & \exists t_{\text{now}} \exists x_1 \exists x_2. [\\ & \text{Now}(t_{\text{now}}) \quad \wedge \\ & \text{temp}(\text{burner}_1, x_1, t) \wedge x_1 \geq 35 \quad \wedge \\ & \text{temp}(\text{burner}_1, x_2, t+1) \wedge x_2 \geq 35 \quad \wedge \\ & \forall t' \forall x'. (t+2 \leq t' \leq t_{\text{now}} \wedge \text{temp}(\text{burner}_1, x', t')) \Rightarrow x' < 35 \\ &] \end{aligned} \quad (5)$$

Note that Q can return answers about past time instants. For example, $A_3 = \{t \mapsto 1\}$ – that is, evaluating Q at time instant 3 produces an answer that refers to time instant 1. This illustrates the benefit of distinguishing time instants at which a query is evaluated from time instants that the query talks about. The former time instant determines what updates have been observed and thus plays a prominent role in the conceptual view of CER. In contrast, associating updates with time stamps can be viewed as a question of modelling – that is, they can be handled as part of the CER “model content”.

► **Example 2.** While the query language from Example 1 is very expressive, it has a significant drawback: answering any first-order query correctly over an arbitrary stream requires storing all observed information. Therefore, it might be interesting to identify query languages for which each query can be processed on an arbitrary stream using a finite amount of memory.

A very simple way to achieve this is to evaluate each query inside a *temporal window*. For the purpose of this example, let us extend the notion of a query to a pair (φ, n) , where φ is a domain-independent first-order formula, and n is an integer specifying the window aperture. To evaluate such a query over a world view W_i , one evaluates φ over the subset of all facts in W_i whose time stamp is between $t_{now} - n$ and t_{now} , for $Now(t_{now}) \in W_i$. One can now investigate the nonfunctional requirements and the structural constraints on the stream that will allow a CER system to answering an arbitrary such query over an arbitrary stream.

Answering queries in this query language is still difficult, for at least the following two reasons.

- Updates are not bounded in size. That is, we cannot answer any first-order query using a finite amount of memory if the number of facts in each update can exceed the available memory.
- We have not placed any restriction on the relation between the instant i of an update U_i and the time stamps of the facts contained inside U_i . As a result, U_i can refer to instants that will become relevant arbitrarily far in the future, and storing all such instants may require an unbounded amount of memory.

To overcome these difficulties, we may place the following two structural constraints on the stream.

- We assume that each update U_i contains no more than ℓ facts for some fixed integer ℓ . For example, this constraint may hold in a setting where the number of sensors is limited to ℓ .
- We assume that the time stamp of each fact in U_i must be between $i - \varphi$ and i for some fixed integer φ . This constraint may hold in a setting where a global clock ensures that no sensor produces “future” time stamps, and that all sensor readings are delivered to the CER system within φ time instants.

The above structural constraints on the stream may not hold in every setting. If, however, they hold, then a CER system can answer every query with window n by storing only the last n updates, which requires a bounded amount of memory. In turn, this conceptual observation opens to door to further investigation of the algorithms and data structures necessary to evaluate such queries efficiently.

► **Example 3.** In Examples 1 and 2, new observations were simply appended to the world view (while updating the current time stamp). Our meta-model for CER, however, allows us to also capture a setting where updates can retract information from the world view. For example, let us assume that we equip the gas burners from Example 1 with an additional sensor that can determine that a reading produced at an earlier time instant was invalid. To incorporate this, we can extend the update language so that each update is of the form

$U_i = \circ_i F_i$, where \circ_i is either $+$ or $-$, and F_i is a set of time-stamped facts as in Example 1. Then, we can define the aggregation function inductively as follows, where ϵ is the empty sequence of updates:

$$\text{AGG}(\epsilon) = \emptyset \quad (6)$$

$$\text{AGG}(U_1.U_2 \dots U_1.U_{i-1}. + F_n) = \text{AGG}(U_1.U_2 \dots U_1.U_{i-1}) \cup F_n \quad (7)$$

$$\text{AGG}(U_1.U_2 \dots U_1.U_{i-1}. - F_n) = \text{AGG}(U_1.U_2 \dots U_1.U_{i-1}) \setminus F_n \quad (8)$$

In other words, aggregating updates now involves both adding and retracting facts. Note that each W_i reflects the information that is believed to be true at instant i . For example, consider the following updates:

$$U_1 = +\{temp(burner_1, 40, 1)\} \quad (9)$$

$$U_2 = +\{temp(burner_2, 20, 2)\} \quad (10)$$

$$U_3 = -\{temp(burner_1, 40, 2)\} \quad (11)$$

World view W_2 contains both $temp(burner_1, 40, 1)$ and $temp(burner_2, 20, 2)$, whereas world view W_3 contains only the latter fact and thus takes into account that the former fact was found to be in error.

► **Example 4.** We can further extend Example 3 and assume that each update involves addition or retraction of a logical formula expressed in a temporal logic. The aggregation function can combine all updates into the current world view using belief revision operators. Finally, a query can involve checking entailment of facts/formulas from world views.


► **Example 5.** All examples thus far considered updates and queries encoded using symbolic languages, but the proposed CER meta-model makes no such assumptions. For example, the query can be given as a neural network, updates can consist of readings for various inputs to the network, and the aggregation function should determine how to combine different updates into the input to the network. Then, we can analyse which classes of neural networks can express what kinds of queries.

4.3.2.3 Next Steps

It was suggested during the seminar that the natural next step would be to prepare a survey paper that would (i) describe the meta-model in more detail, (ii) discuss how to instantiate this meta-model in order to capture the various proposals from the literature, and (iii) identify classes of languages and constraints found in the existing body of literature. It was suggested that such a publication might be produced jointly by the interested parties at one of relevant future events, such as the Stream Reasoning Workshop.

4.4 Benchmarking

Riccardo Tommasini (University of Tartu, EE)

License  Creative Commons BY 3.0 Unported license
© Riccardo Tommasini

Working group participants:

- Thomas Eiter (TU Wien, AT).
- David Eyers (University of Otago, NZ).
- Wim Martens (Universität Bayreuth, DE).
- Ruben Mayer (TU München, DE).
- Umakishore Ramachandran (Georgia Institute of Technology – Atlanta, US).
- Till Rohrmann (Ververica – Berlin, DE).
- Riccardo Tommasini (University of Tartu, EE).
- Stijn Vansummeren (Université Libre de Bruxelles, BE).

Domain-specific benchmarks aim to foster technological progress by guaranteeing a fair assessment [5]. To this extent, Gray’s seminal work identifies important principles that drive the design of data system benchmarks.

- *Relevance*: a benchmark must measure the price/performance ration of systems when performing typical operations within its domain.
- *Portability*: a benchmark must be easy to implement on many heterogeneous systems and architectures.
- *Scalability*: a benchmark should apply to small and large computer systems.
- *Simplicity*: a benchmark must be understandable, otherwise it will lack credibility.

In addition, Karl Huppler pushed the benchmark guidelines event further. In “The art of building a good benchmark” [6], he identifies the following three additional principles.

- *Repeatability*: there is confidence that the benchmark can be run a second time with the same result.
- *Fairness*: all systems and/or software being compared can participate equally.
- *Verifiability*: there is confidence that the documented result is real.

Gray and Huppler stressed the economical aspects of benchmarking. Intuitively, a good benchmark should be representative yet sustainable for the community to adopt it. For a proper evaluation, it is not necessary to be compliant with all the listed principles but only to those reflecting the benchmarking purpose [6].

In the context of complex event recognition (CER) a consolidated benchmark is still missing. In the related literature, few attempts tried to identify use-cases, key performance indicators (KPI), and relevant challenges to the research community to address [8, 2, 1].

Nevertheless, performance evaluations are still not homogeneous. In absence of real-world event streams, researchers are forced to adapt analytic benchmarks like the Linear Road [1], database benchmarks like BEAST [3], or benchmarks designed for Message-Oriented Middleware [7]. Intuitively, this handcrafted approach to benchmarking limits the repeatability and reproducibility of the results. The cost of maintenance of the customized benchmarks is entirely on the individual research groups and, as a result, it is hard to guarantee long-term support.

To this extent, the working group focused on identifying a sustainable path with concrete operational steps that could lead to the design of a domain-specific benchmark for CEP that is maintainable by the community.

4.4.1 Types of benchmarks (relevance, simplicity, and fairness)

Initially, the discussion focused on identifying interesting types of benchmarks. Two main areas emerged, i.e., macro- (also known as use-case driven) and micro-benchmarks. The former focuses on evaluating systems with respect to specific workloads, typically inspired by real-world scenarios. The latter, instead, focuses on evaluating the performance of single operators.

Macro-benchmarks directly relate with the ongoing effort behind the DEBS Grand Challenges, which yearly provide interesting use-cases and workloads for the community to solve. On the other hand, micro-benchmarks relate to the definition of a common model for CER e/o a core algebra of operation that CER engines must support.

4.4.2 Lack of standards (portability)

Industrial and academic research on CER highlights the lack of shared data and query models towards a standardization. These agreements are crucial to develop and maintain a benchmarking suite for the community.

In particular, during the meeting is emerged that the identification of data formats and query languages is of paramount importance to move forwards any activity in the context of benchmarking.

4.4.3 Technical support (scalability, repeatability, and verifiability)

Last but not least, the working group has highlighted the issue related to technical supports. To this extent, similar criteria used for data publishing should be adopted. Community benchmarks should be FAIR [9], i.e.,

- *Findable*: it must be identifiable and registered in searchable resources.
- *Accessible*: it must be retrievable by their identifiers using open and standard protocols.
- *Interoperable*: it must use a formal and shared language for its representation and reference other relevant resources.
- *Reusable*: it must be described with a plurality of accurate and relevant attributes, e.g., licenses, and provenance metadata.

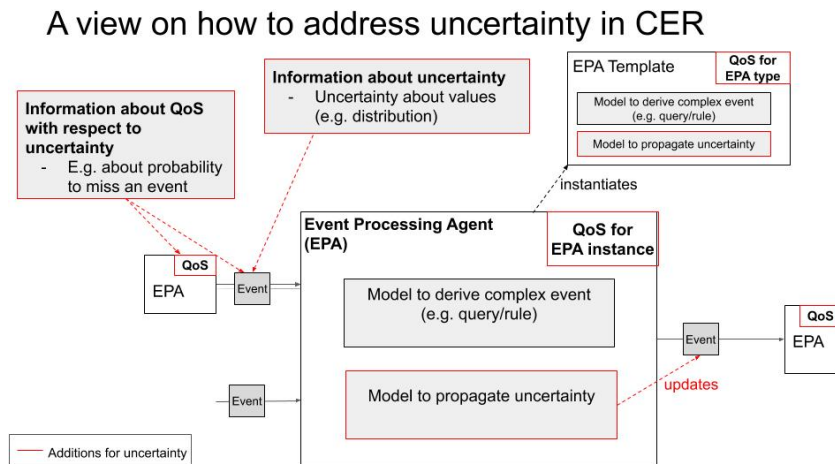
4.4.4 Conclusion and roadmap

In conclusion, the working group has identified two concrete steps towards the definition of a standard benchmark for CER. The two steps plan is described below by indicating, for each step, a minimum set of subtask and potential outcomes:

1. Design a replication study and literature analysis (including DEBS Grand Challenges)
 - Identify the dimensions of interest for the literature starting from the recent works [4];
 - identify the systems and experiments of interest for the replication study starting from the DEBS Grand Challenges;
 - consider as input the ongoing work on uniforming languages and models.
2. Start a working group that has the extent of creating a community benchmark.
 - Contact the Linked Data Benchmark Council
 - Invite people from DEBS Grand Challenge community.
 - Integrate this into the Stream Reasoning COST Action organized by Thomas Eiter.
 - Identify other academics and industrial groups.

References

- 1 A. Arasu, M. Cherniack, E. F. Galvez, D. Maier, A. Maskey, E. Ryzkina, M. Stonebraker, and R. Tibbetts. Linear road: A stream data management benchmark. In M. A. Nascimento, M. T. Özsu, D. Kossmann, R. J. Miller, J. A. Blakeley, and K. B. Schiefer, editors, *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, VLDB 2004, Toronto, Canada, August 31 – September 3 2004*, pages 480–491. Morgan Kaufmann, 2004. 10.1016/B978-012088469-8.50044-9. URL <http://www.vldb.org/conf/2004/RS12P1.PDF>.
- 2 P. Bizarro. Bicep – benchmarking complex event processing systems. In K. M. Chandy, O. Etzion, and R. von Ammon, editors, *Event Processing, 6.5. – 11.5.2007*, volume 07191 of *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2007. URL <http://drops.dagstuhl.de/opus/volltexte/2007/1143>.
- 3 A. Geppert, S. Gatzui, and K. R. Dittrich. A designer’s benchmark for active database management systems: oo7 meets the BEAST. In T. K. Sellis, editor, *Rules in Database Systems, Second International Workshop, RIDS ’95, Glyfada, Athens, Greece, September 25 - 27, 1995, Proceedings*, volume 985 of *Lecture Notes in Computer Science*, pages 309–326. Springer, 1995. 10.1007/3-540-60365-4_135. URL https://doi.org/10.1007/3-540-60365-4_135.
- 4 N. Giatrakos, E. Alevizos, A. Artikis, A. Deligiannakis, and M. N. Garofalakis. Complex event recognition in the big data era: a survey. *VLDB J.*, 29(1):313–352, 2020. 10.1007/s00778-019-00557-w. URL <https://doi.org/10.1007/s00778-019-00557-w>.
- 5 J. Gray, editor. *The Benchmark Handbook for Database and Transaction Systems (1st Edition)*. Morgan Kaufmann, 1991.
- 6 K. Huppler. The art of building a good benchmark. In R. O. Nambiar and M. Poess, editors, *Performance Evaluation and Benchmarking, First TPC Technology Conference, TPCTC 2009, Lyon, France, August 24-28, 2009, Revised Selected Papers*, volume 5895 of *Lecture Notes in Computer Science*, pages 18–30. Springer, 2009. 10.1007/978-3-642-10424-4_3. URL https://doi.org/10.1007/978-3-642-10424-4_3.
- 7 K. Sachs, S. Kounev, S. Appel, and A. P. Buchmann. Benchmarking of message-oriented middleware. In A. S. Gokhale and D. C. Schmidt, editors, *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems, DEBS 2009, Nashville, Tennessee, USA, July 6-9, 2009*. ACM, 2009. 10.1145/1619258.1619313. URL <https://doi.org/10.1145/1619258.1619313>.
- 8 T. Scharrenbach, J. Urbani, A. Margara, E. D. Valle, and A. Bernstein. Seven commandments for benchmarking semantic flow processing systems. In P. Cimiano, Ó. Corcho, V. Presutti, L. Hollink, and S. Rudolph, editors, *The Semantic Web: Semantics and Big Data, 10th International Conference, ESWC 2013, Montpellier, France, May 26-30, 2013. Proceedings*, volume 7882 of *Lecture Notes in Computer Science*, pages 305–319. Springer, 2013. 10.1007/978-3-642-38288-8_21. URL https://doi.org/10.1007/978-3-642-38288-8_21.
- 9 M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, J. Bouwman, A. J. Brookes, T. Clark, M. Crosas, I. Dillo, O. Dumon, S. Edmunds, C. T. Evelo, R. Finkers, A. Gonzalez-Beltran, A. J. G. Gray, P. Groth, C. Goble, J. S. Grethe, J. Heringa, P. A. C. ’t Hoen, R. Hooft, T. Kuhn, R. Kok, J. Kok, S. J. Lusher, M. E. Martone, A. Mons, A. L. Packer, B. Persson, P. Rocca-Serra, M. Roos, R. van Schaik, S.-A. Sansone, E. Schultes, T. Sengstag, T. Slater, G. Strawn, M. A. Swertz, M. Thompson, J. van der Lei, E. van Mulligen,



■ **Figure 2** Addressing uncertainty in CER

J. Velterop, A. Waagmeester, P. Wittenburg, K. Wolstencroft, J. Zhao, and B. Mons. The fair guiding principles for scientific data management and stewardship. *Scientific Data*, 3 (1):160018, 2016. 10.1038/sdata.2016.18. URL <https://doi.org/10.1038/sdata.2016.18>.

4.5 Uncertainty in Complex Event Recognition

Han van der Aa (HU Berlin, DE), Avigdor Gal (Technion – Haifa, IL), Sylvain Hallé (University of Quebec at Chicoutimi, CA), Annika M. Hinze (University of Waikato, NZ), and Holger Ziekow (HFU – Furtwangen, DE)

License © Creative Commons BY 3.0 Unported license
© Han van der Aa, Avigdor Gal, Sylvain Hallé, Annika M. Hinze, and Holger Ziekow

Working group participants:

- Han van der Aa (HU Berlin, DE).
- Thomas Eiter (TU Wien, AT).
- Avigdor Gal (Technion – Haifa, IL).
- Sylvain Hallé (University of Quebec at Chicoutimi, CA).
- Manfred Hauswirth (Fraunhofer FOKUS – Berlin, DE).
- Fredrik Heintz (Linköping University, SE).
- Annika M. Hinze (University of Waikato, NZ).
- Danh Le Phuoc (TU Berlin, DE).
- Holger Ziekow (HFU – Furtwangen, DE).

Uncertainty is inherent in many use cases for complex event recognition (CER). This uncertainty stems from two main sources. One source is noise and errors in the input data for complex event recognition. For instance, measurements from input sensors may be incorrect or incomplete. The other source for uncertainty lies in the inference for CER. That is, the inference of a complex event from simpler events may be of probabilistic nature. For instance, a system may classify a situation as an emergency, even though it turns out to be a false alarm.

Uncertainty can also come from the deliberate intention of an event publisher to prevent complete disclosure of an event stream to its subscribers. One possible manifestation of this situation is when a source of events is subject to access control constraints, such as privacy policies. In the same way that the access to statistical databases can be restricted in order to preserve anonymity (e.g. query restriction and query perturbation [1]) by the deliberate blocking or insertion of noise, one can imagine that a data stream may be subjected to similar conditions, which themselves may depend on the past content of the stream, as in history-based access control [2]. In such cases, the reason for the presence of uncertainty is not technical or fortuitous in nature; uncertainty may even be inserted with the precise design of actively preventing some processing of the stream by downstream agents.

We argue that uncertainty cannot be avoided in many applications of CER and see the need to explicitly account for it in CER systems. In particular, we see the need for reasoning about uncertainty (of confidence) when reacting to events. If a system detects an emergency with very low confidence, we may react with different measures than if the confidence is high. However, our threshold for acting on a possible emergency is lower than for less consequential situations. CER systems should therefore provide means to make uncertainty explicit. Yet, it is common practice to simply employ – often fixed – thresholds in CER and treat detected events as if they were certain. This not only projects a false sense of certainty, it prohibits any reasoning about uncertainty when taking action. We therefore see the need to expand general frameworks for CER with means to capture and reason about uncertainty. As results, we augmented a general architecture for CER with specific extension points. These extension points show what is missing in current CER frameworks to add the dimension of uncertainty. Figure 2 provides an overview of the extended CER framework.

A range of candidate techniques for implementing the identified extension point as well as many tradeoffs amongst them exists. We believe that the CER would benefit from explicitly capturing the design options along with their strengths and weaknesses in an overarching framework.

References

- 1 N. Adamand, J. Wortmann. *Security-control methods for statistical databases: A comparative study*. ACM Computing Surveys, 1989.
- 2 M. Abadi, C. Fournet. *Access control based on execution history*. In Proceedings of the 10th Annual Network and Distributed System Security Symposium, 2003.

Participants

- Elias Alevizos
Demokritos – Athens, GR
- Alexander Artikis
NCSR Demokritos – Athens, GR
- François Bry
LMU München, DE
- Diego Calvanese
Free University of
• Bozen-Bolzano, IT
- Daniele Dell’Aglio
Universität Zürich, CH
- Emanuele Della Valle
Polytechnic University of
Milan, IT
- Thomas Eiter
TU Wien, AT
- David Eyers
University of Otago, NZ
- Avigdor Gal
Technion – Haifa, IL
- Minos Garofalakis
Technical University of Crete –
Chania, GR
- Alejandro J. Grez
PUC – Santiago de Chile, CL
- Sylvain Hallé
University of Quebec at
Chicoutimi, CA
- Manfred Hauswirth
Fraunhofer FOKUS – Berlin, DE
- Fredrik Heintz
Linköping University, SE
- Annika M. Hinze
University of Waikato, NZ
- Boris Koldehofe
University of Groningen, NL
- Danh Le Phuoc
TU Berlin, DE
- Alessandro Margara
Polytechnic University of
Milan, IT
- Wim Martens
Universität Bayreuth, DE
- Ruben Mayer
TU München, DE
- Angelo Montanari
University of Udine, IT
- Boris Motik
University of Oxford, GB
- Thomas Prokosch
LMU München, DE
- Umakishore Ramachandran
Georgia Institute of Technology –
Atlanta, US
- Tore Risch
Uppsala University, SE
- Cristian Riveros
PUC – Santiago de Chile, CL
- Till Rohrmann
Ververica – Berlin, DE
- Kurt Rothermel
Universität Stuttgart, DE
- Sabri Skhiri
EURA NOVA –
Mont-Saint-Guibert, BE
- Kostas Stathis
Royal Holloway, University of
London, GB
- Riccardo Tommasini
University of Tartu, EE
- Martin Ugarte
Millenium Institute –
Santiago de Chile, CL
- Jacopo Urbani
VU University Amsterdam, NL
- Han van der Aa
HU Berlin, DE
- Stijn Vansummeren
Free University of Brussels, BE
- Matthias Weidlich
HU Berlin, DE
- Thomas Zeume
TU Dortmund, DE
- Holger Ziekow
HFU – Furtwangen, DE

