# Complex Event Recognition in the Big Data Era

Nikos Giatrakos[1], Alexander Artikis[2,3], Antonios Deligiannakis[1]
and Minos Garofalakis[1,4]
[1]Technical University of Crete, Chania, Greece
[2]University of Piraeus, Athens, Greece
[3]NCSR Demokritos, Athens, Greece
[4]ATHENA Research and Innovation Centre, Athens, Greece
{ngiatrakos,adeli,minos}@softnet.tuc.gr, a.artikis@unipi.gr

## Abstract and Goal

The concept of event processing is established as a generic computational paradigm in various application fields, ranging from data processing in Web environments, over maritime and transport, to finance and medicine. Events report on state changes of a system and its environment. Complex Event Recognition (CER) in turn, refers to the identification of complex/composite events of interest, which are collections of simple events that satisfy some pattern, thereby providing the opportunity for reactive and proactive measures. Examples 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, fraud in electronic marketplaces, cardiac arrhythmias, and epidemic spread. In each scenario, CER allows to make sense of Big event Data streams and react accordingly. The goal of this tutorial is to provide a step-by-step guide for realizing CER in the Big Data era. To do so, it elaborates on major challenges and describes algorithmic toolkits for optimized manipulation of event streams characterized by high volume, velocity and/or lack of veracity, placing emphasis on distributed CER over potentially heterogeneous (data variety) event sources. Finally, we highlight future research directions in the field.

## 1. INTRODUCTION & MOTIVATION

Systems for symbolic Complex Event Recognition (CER) accept as input a stream of time-stamped 'simple, derived events'. These are the result of applying a computational derivation process to some other event, such as a measurement coming from a sensor. Using simple events as input, CER systems identify complex/composite events of interest, which are collections of events that satisfy some pattern [4, 11, 12]. The pattern of a complex event imposes temporal and, possibly, atemporal constraints on its subevents, i.e. simple or other complex events. Thus, contrary to "classical" data stream querying that leaves to the clients the responsibility of attaching a particular meaning on the result set, CER encompasses the ability to query for complex patterns through predefined rules that match incoming event notifications on the basis of their content and other spatiotemporal constraints [10, 13]. Hence, CER introduces peculiarities and significantly differentiates itself from traditional streaming conceptualizations [15].

Consider, for example, maritime surveillance, where preventing accidents at sea by monitoring vessel activity results in substantial financial savings for shipping companies and averts maritime ecosystem damages. CER allows for expressing patterns that attach meaning to fused, context-deprived streaming tuples for the real-time detection of suspicious or potentially dangerous situations that may have a serious impact on the environment and on safe navigation at sea [25]. Simple position signals are continuously collected from hundreds of thousands of ships worldwide. Vessels report their position in different time scales, while the messages are often noisy, offering contradicting information. Vessel movement is combined with static geographical information while, for effective vessel identification and tracking, additional data sources are taken into account, such as weather reports and frequently updated satellite images of the surveillance areas. Moreover, the properties of the event patterns also add to the complexity of CER. Patterns may be required to adapt to dynamic environments, integrate various event sources, and (consequently) be inherently uncertain.

Therefore, the unique characteristics of Big event Data, impose novel challenges to efficient stream management. This tutorial presents the end-to-end modeling pipeline and a step-by-step guide for realizing CER in the Big Data era. We elaborate on research challenges and practical algorithmic toolkits for: (a) parallel (volume), (b) elastic (volatile velocity/distribution), (c) uncertainty-aware (veracity) and (d) scalable distributed solutions over potentially heterogeneous (variety) event sources. Special emphasis will be placed on (c), (d) given their limited study so far and notable recent advancements. Finally, we will elaborate on contemporary research trends and point out takeaway messages for the audience. To illustrate the reviewed approaches, we will use real-world applications from the FERARI[1] and datACRON[2] projects, that employ CER techniques for Big Data analytics.

**Target Audience.** The intended audience consists of academics, students and practitioners investigating the open issues of CER, and/or are willing to apply event recognition techniques for extracting knowledge from structured and

[1]http://www.ferari-project.eu/
[2]http://datacron-project.eu/


*Proceedings of the VLDB Endowment,* Vol. 10, No. 12

unstructured datasets. Familiarity with distributed systems and/or artificial intelligence techniques is desirable.

## 2. TOPICS & DESCRIPTION

While modern Big Data applications related to CER are characterized by a variety of challenging features, research approaches tend to focus on one or few of them, often restricting the processing to take place at a single site/cluster, thus ignoring the inherent distribution in the collected data, or avoiding to tackle the lack of veracity of the data. The recent development of stream processing frameworks, such as Apache STORM, Spark, AKKA, Apache Flink, etc, have made simpler to design approaches for distributed processing, but they are not on their own sufficient for CER. Thus, in this tutorial we will show how to bridge the gap between expressive (relational) CER languages describing application queries and the aforementioned Big Data-oriented frameworks. In the tutorial, we opt for presenting the end-to-end modeling pipeline for materializing and optimizing CER at large and massive scales dealing with all Vs and the D (for distribution) of Big Data streams. Below we present and describe the main topics that are covered, while an outline of the tutorial follows in the upcoming section.

### 2.1 Complex Event Recognition Languages

Numerous CER systems and languages have been proposed in the literature [3, 10, 5]. These systems have a common goal, but differ in their architectures, data models, pattern languages, and processing mechanisms. For example, many CER systems provide users with a pattern language that is later compiled into some form of automaton. The automaton model is generally used to provide the semantics of the language and/or as an execution framework for pattern matching [8, 24, 33]. Apart from automata-based models, some CER systems employ tree-based models [21, 1]. Again, tree-based formalisms are used for both modeling and recognition, i.e., they may describe the complex event patterns to be recognized as well as the applied recognition algorithm. Recently, logic-based approaches to CER have been attracting considerable attention, since they exhibit a formal, declarative semantics, and at same have been proven efficient enough for Big Data applications [4].

### 2.2 Uncertainty Handling

The events arriving at a CER system almost always carry a certain degree of uncertainty and/or ambiguity. Information sources might be heterogeneous, with data of different schemata, they might fail to respond or even send corrupt data. Even if we assume perfectly accurate sensors, the domain under study might be difficult or impossible to model precisely, thereby leading to another type of uncertainty. Until recently, most CER systems did not make any effort to handle uncertainty (instructively, see the relevant discussion on uncertainty in [10]). This need is gradually being acknowledged and uncertainty handling has become a major line of research for CER.

Most efforts targeting the problem of uncertainty in CER are based on extensions of crisp engines, the majority of which employ automata [32, 23]. Input events are usually in the form of streams/sequences of events, similar to strings of characters recognized by (Non-)Deterministic Finite Automata. Complex events are usually expressed in a declarative way with the *sequence* operator (time ordered conjunction) playing a central role. These expressions are transformed into automata, using the stream of simple events as input. In the probabilistic versions of automata-based methods, it is usually the simple events that are uncertain, accompanied by probability values with respect to their occurrence and/or attributes, as opposed to the complex event patterns. The goal is to use these probabilistic simple events in order to determine the probabilities of complex events.

Another line of research revolves around methods employing probabilistic graphical models in order to handle uncertainty [29, 22, 7]. These models take the form of networks whose nodes represent random variables and edges encode probabilistic dependencies. The two main classes of probabilistic graphical models used in CER are Markov Networks and Bayesian Networks, the former being undirected models whereas the latter are directed. When used for CER, Markov Networks may be combined with first-order logic, in which case they are called Markov Logic Networks. The nodes in a Markov Logic Network represent ground logical predicates, as determined by the (weighted) formulas that express complex event patterns. When Bayesian Networks are used, the nodes usually represent simple/complex events.

Many approaches to CER using Markov Logic Networks are concerned with human activity recognition, with input events derived mostly from video sources (less frequently from GPS or RFID traces). As a result, many of them develop solutions that are domain-dependent. In the tutorial, we will focus on those representative papers that are more closely related to CER, by providing a more generic way for handling events, such as the combination of Markov Logic Networks with Allen's Interval Algebra and the Event Calculus.

Contrary to automata-based solutions, Markov Logic Networks focus on encoding probabilistic rules. This allows both for incorporating background knowledge and for building hierarchies of complex events with correct probability propagation. On the other hand, they use the less intuitive weights instead of probabilities, which indicate how strong a rule is compared to the others. While it might be possible for certain simple domains to manually define weights, usually a learning phase is required to optimize them.

The manual development of complex event patterns is a tedious, time-consuming and error-prone process. Moreover, it is often necessary to update such patterns during the event recognition process, due to new information about the application under consideration. Consequently, methods for automatically generating and refining event patterns from data are highly desirable. Both supervised and unsupervised techniques have been employed to automatically adapt and construct event patterns [22]. In addition to learning the structure of an event pattern, the weight (confidence value) of the pattern can be learned from data. To avoid sub-optimal results, the tasks of structure learning and weight learning should not be separated.

### 2.3 Complex Event Recognition at Scale

The volume and velocity of Big Data streams pose unprecedented challenges in designing scalable algorithms that can process massive amounts of data online so as to support real-time, CER related, decision making. In fact, current distributed stream processing systems like Apache Storm and Spark Streaming leverage Big Data fundamentals, running on commodity clusters and clouds, supporting fast processing

over voluminous data. CER at large scale is realized by distributing event recognition tasks among multiple processing nodes. When the aforementioned nodes are collocated, i.e., within a cluster and can be assumed to share resources (such as memory), scalability places its emphasis on efficient *parallelization strategies*, tailored to CER [6, 9, 13, 18, 20, 26], as well as elastic resource utility [13, 17, 31]. In a nutshell, *elasticity* refers to the fundamental property of dynamically adjusting resource allocation or the parallelization strategy itself, so that throughput (rate of tuples being processed) is continuously optimized.

There is a number of approaches tailored for parallel CER in the literature [13, 20, 30]. Partition-based parallelization [18] uses one or more attributes of incoming event tuples as the keys in a partition-map that assigns certain streaming tuples to a processing unit. The partition-map ensures partition contiguity and partition isolation during the extraction of event pattern matches. State-based parallelization [6] entails that each processing unit is assigned a thread that is responsible for a certain Non-deterministic Finite Automaton (NFA) state in the logical pattern matching execution plan. The processing is distributed in the sense that both buffer management as well as predicate-related optimizations concerning individual NFA states are performed in each node individually. Partial pattern matches derived in each processing node, need to be pipelined to the node that processes tuples involving the next state in the NFA to extract full matches. Operator-based parallelization [26] assigns operators to processing units while operators can send events to each other within a processing node and also to operators on other nodes so as to detect full pattern matches. Run-based parallelization [6] performs the processing in batches of event tuples. Each batch includes ordered, i.e., pattern matching takes place on a given sequence of input events, tuples and possesses an identifier. Based on this identifier, it is assigned to a particular processing node where it initiates a run. To extract full matches that span multiple nodes, the end of a batch must be replicated at the beginning of the runs belonging to nodes receiving contiguous batches. A key observation is that there is no one size fit all solution for parallel CER, while, surprisingly, most approaches lack provisions for elastic resource allocation.

One step further, CER applications at vast scale operate over a set of *distributed*, i.e., geographically dispersed, sites each accumulating event streams which later need to be efficiently synthesized to provide holistic answers to application queries. In this case, apart from retaining the advantages of parallelization and elasticity locally per site, communication scalability issues are of additional essence. To achieve that, CER algorithms generally incorporate a couple of tools in their arsenal. On the one hand, *in-situ processing* constructs qualifying conditions and places local filters to minimize communication by performing as much of the computation as possible in the event sources (potentially, subset of sites) [6, 16, 19, 27]. While these conditions may be easy to set for simple aggregates, they are much more challenging to invent for generic, non-linear event query operators. On the other hand, *in-network processing* [28] pushes the evaluation of query operations, defined on the union of local data streams, to sites that are near to the relevant event sources [14, 2], thus also exploiting data stream variety. In that, the qualified portion of local data streams is synthesized early, extracting compact aggregates to be further forwarded in the network.

# 3. OUTLINE OF THE TUTORIAL

**Tutorial Duration:** 90 minutes.

**Tutorial Structure.**

*Introduction* (10 minutes).
- Introduction & motivation.
- Relevant research areas.
- Real-world applications & running examples.

Indicative reference: [10].

*Complex Event Recognition Languages* (10 minutes).
- Requirements & properties of CER languages.
- A unifying Event Algebra.

Indicative references: [3, 4, 11, 33].

*Uncertainty Handling* (20 minutes).
- Automata-based methods.
- Probabilistic graphical models.

Indicative references: [7, 22, 23, 29, 32].

*Scalable Distributed CER* (45 minutes).
- Towards scalable CER.
  - Parallelization strategies.
  - Elasticity in CER.
  - CER over distributed sites.
- Optimized in-network CER.
  - Optimal operator push under various metrics.
  - Handling adaptivity - Operator migration.
- Leveraging in-situ processing.
  - Linear vs non-linear operator monitoring.

Indicative references: [2, 6, 9, 13, 14, 17, 18, 20, 26, 27, 31].

*Open Issues & Takeaways* (5 minutes).

# 4. PRESENTER BIOS

*Nikos Giatrakos* is currently a Postdoctoral Research Associate at the School of Electrical & Computer Engineering of the Technical University of Crete. His research interests include database systems, sensor networks, distributed query processing, data synopses and approximate query processing.

*Alexander Artikis* is an Assistant Professor in the University of Piraeus (Greece), and a Research Associate at NCSR "Demokritos" (Athens, Greece), where he leads the Complex Event Recognition lab (`http://cer.iit.demokritos.gr/`). His research interests lie in the areas of artificial intelligence and distributed systems.

*Antonios Deligiannakis* is an Associate Professor of Computer Science at the School of Electrical & Computer Engineering of the Technical University of Crete, which he joined in 2008. His research interests are in the broad area of Big Data Analytics over data streams, including distributed data processing, secure processing and query authentication over data outsourced to untrusted servers, distributed complex event processing in large scale systems, approximate query processing and large scale data mining.

*Minos Garofalakis* is a Professor at the School of Electrical & Computer Engineering of the Technical University of Crete, and the Director of the Institute for the Management of

Information Systems (IMIS) at the ATHENA Research and Innovation Center in Athens, Greece. His research interests lie in the broad areas of Big Data Analytics, including data stream processing, probabilistic data management, data synopses and approximate query processing, network data management, and large-scale machine learning/data mining.

## 5. ACKNOWLEDGMENTS

Part of the authors' work was supported by the European Commission under the H2020 grant *datACRON* (no. 687591) and the FP7 grant *FERARI* (no. 619491).

## 6. REFERENCES

[1] Esper. `http://esper.codehaus.org/`.

[2] M. Akdere, U. Çetintemel, and N. Tatbul. Plan-based complex event detection across distributed sources. *Proc. VLDB Endow.*, pages 66–77, 2008.

[3] A. Artikis, A. Margara, M. Ugarte, S. Vansummeren, and M. Weidlich. Complex event recognition languages. In *DEBS*, 2017.

[4] A. Artikis, M. J. Sergot, and G. Paliouras. An event calculus for event recognition. *IEEE Trans. Knowl. Data Eng.*, 27(4):895–908, 2015.

[5] A. Artikis, A. Skarlatidis, F. Portet, and G. Paliouras. Logic-based event recognition. *Knowledge Engineering Review*, 27(4):469–506, 2012.

[6] C. Balkesen, N. Dindar, M. Wetter, and N. Tatbul. Rip: run-based intra-query parallelism for scalable complex event processing. In *DEBS*, pages 3–14, 2013.

[7] W. Brendel, A. Fern, and S. Todorovic. Probabilistic Event Logic for Interval-based Event Recognition. In *CVPR*, pages 3329–3336, 2011.

[8] L. Brenna, A. J. Demers, J. Gehrke, M. Hong, J. Ossher, B. Panda, M. Riedewald, M. Thatte, and W. M. White. Cayuga: a high-performance event processing engine. In *SIGMOD*, pages 1100–1102, 2007.

[9] G. Cugola and A. Margara. Complex event processing with t-rex. *J. Syst. Softw.*, 85(8):1709–1728, aug 2012.

[10] G. Cugola and A. Margara. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys*, 44(3):15:1–15:62, 2012.

[11] C. Dousson and P. L. Maigat. Chronicle recognition improvement using temporal focusing and hierarchisation. In *IJCAI*, pages 324–329, 2007.

[12] O. Etzion and P. Niblett. *Event Processing in Action*. Manning Publications Co., 2010.

[13] I. Flouris, N. Giatrakos, A. Deligiannakis, M. Garofalakis, M. Kamp, and M. Mock. Issues in complex event processing: Status and prospects in the big data era. *Journal of Systems and Software*, 127:217 – 236, 2017.

[14] I. Flouris, V. Manikaki, N. Giatrakos, A. Deligiannakis, M. Garofalakis, M. Mock, S. Bothe, I. Skarbovsky, F. Fournier, M. Stajcer, T. Krizan, J. Yom-Tov, and T. Curin. Ferari: A prototype for complex event processing over streaming multi-cloud platforms. In *SIGMOD*, pages 2093–2096, 2016.

[15] M. Garofalakis, J. Gehrke, and R. Rastogi. *Data Stream Management: Processing High-Speed Data Streams*. Springer, 2016.

[16] N. Giatrakos, A. Deligiannakis, M. Garofalakis, I. Sharfman, and A. Schuster. Distributed geometric query monitoring using prediction models. *ACM Trans. Database Syst.*, 39(2):16:1–16:42, 2014.

[17] T. Heinze, Y. Ji, Y. Pan, F. J. Grueneberger, Z. Jerzak, and C. Fetzer. Elastic complex event processing under varying query load. In *BD3@ VLDB*, 2013.

[18] M. Hirzel. Partition and compose: parallel complex event processing. In *DEBS*, pages 191–200, 2012.

[19] D. Keren, G. Sagy, A. Abboud, D. Ben-David, A. Schuster, I. Sharfman, and A. Deligiannakis. Geometric monitoring of heterogeneous streams. *IEEE TKDE*, 26(8):1890–1903, 2014.

[20] R. Mayer, C. Mayer, M. A. Tariq, and K. Rothermel. Graphcep: Real-time data analytics using parallel complex event and graph processing. In *DEBS*, pages 309–316, 2016.

[21] Y. Mei and S. Madden. Zstream: a cost-based query processor for adaptively detecting composite events. In *SIGMOD*, pages 193–206, 2009.

[22] E. Michelioudakis, A. Skarlatidis, G. Paliouras, and A. Artikis. OSL$\alpha$ : Online structure learning using background knowledge axiomatization. In *ECML PKDD*, pages 232–247, 2016.

[23] C. Molinaro, V. Moscato, A. Picariello, A. Pugliese, A. Rullo, and V. S. Subrahmanian. PADUA: Parallel Architecture to Detect Unexplained Activities. *ACM (TOIT)*, 14(1):3:1–3:28, 2014.

[24] B. Mozafari, K. Zeng, L. D'Antoni, and C. Zaniolo. High-performance complex event processing over hierarchical data. *ACM Trans. Database Syst.*, 38(4):21:1–21:39, 2013.

[25] K. Patroumpas, E. Alevizos, A. Artikis, M. Vodas, N. Pelekis, and Y. Theodoridis. Online event recognition from moving vessel trajectories. *GeoInformatica*, 21(2):389–427, 2017.

[26] N. P. Schultz-Møller, M. Migliavacca, and P. Pietzuch. Distributed complex event processing with query rewriting. In *DEBS*, pages 4:1–4:12, 2009.

[27] I. Sharfman, A. Schuster, and D. Keren. A geometric approach to monitoring threshold functions over distributed data streams. In *SIGMOD*, pages 301–312, 2006.

[28] U. Srivastava, K. Munagala, and J. Widom. Operator placement for in-network stream query processing. In *PODS*, pages 250–258, 2005.

[29] S. Wasserkrug, A. Gal, O. Etzion, and Y. Turchin. Efficient processing of uncertain events in rule-based systems. *IEEE TKDE*, 24(1):45–58, 2012.

[30] L. Woods, J. Teubner, and G. Alonso. Complex event detection at wire speed with fpgas. *Proc. VLDB Endow.*, 3(1-2):660–669, 2010.

[31] N. Zacheilas, V. Kalogeraki, N. Zygouras, N. Panagiotou, and D. Gunopulos. Elastic complex event processing exploiting prediction. In *Big Data*, pages 213–222, 2015.

[32] H. Zhang, Y. Diao, and N. Immerman. Recognizing patterns in streams with imprecise timestamps. *Inf. Syst.*, 38(8):1187–1211, 2013.

[33] H. Zhang, Y. Diao, and N. Immerman. On complexity and optimization of expensive queries in complex event processing. In *SIGMOD*, pages 217–228, 2014.