R^2CEDM: A Rete-Based RFID Complex Event Detection Method

Xiaoli Peng[1,2,3], Linjiang Zheng[2(✉)], and Ting Liao[1,3]

[1] School of Intelligent Manufacturing,
Sichuan University of Arts and Science, Sichuan 635000, China
[2] College of Computer Science,
Chongqing University, Chongqing 400030, China
zlj_cqu@cqu.edu.cn
[3] Institute of Dazhou Intelligent Manufacturing Technology,
Sichuan 635000, China

Abstract. In various RFID application scenarios, RFID generates real-time and inherent unreliable raw data continually, which contain valuable enterprises business event. How to detect valuable event from RFID raw data has becoming a key issue. A Rete-based RFID complex event detection method called R^2CEDM is proposed. Firstly, α_detecting net is established to detect all attribute of RFID primitive events in R^2CEDM. Then β_detecting net is used to assemble RFID events into RFID complex events with the business rules, and concerned RFID complex events is obtained. The comparative experiments show the proposed R^2CEDM can effectively improve the processing efficiency.

Keywords: Complex event detection · RFID · Rete · Rule match

1 Introduction

RFID system consists of tags, readers and management information systems. RFID is a non-contact automatic identification technology, which has some characteristic such as batch identification, fast mobile identification in comparison with the barcode. In recent years, RFID technology has been widely used in logistics, transportation, medical, agriculture, animal husbandry, special materials management, manufacturing lines and other fields [1]. In various application scenarios, RFID system will generate vast amounts of real-time raw data continually. Although these raw data contain valuable enterprises business event, they are difficult to be used directly by enterprises application system. There are also redundant, incorrect and other characteristics in the RFID raw data [1–3]. How to detect enterprises business logic events from RFID raw data, which are concerned by enterprises, has becoming a key issue that must be solved in RFID application system [4]. Event is a meaningful change in the system, or an occurrence of the interested content, such as a RFID reading process in the RFID system. In general, the RFID events can be classified into RFID primitive event and RFID complex event [5].

RFID event process is shown in Fig. 1 [6]. Through the data cleaning from event filter, the redundant and erroneous RFID primitive events have been removed. Then, RFID complex event can be detected through RFID event aggregation according to the business logic rules, and actively inform to the business system such as WMS, ERP. RFID complex event detection is the key for RFID event processing, which directly affects the application of RFID system.

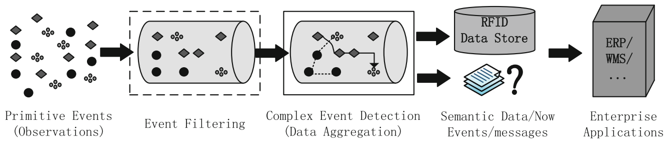

Fig. 1. RFID event process

RFID complex event detection is an important topic in RFID application system. Now, there are some popular RFID complex event detection methods such as finite automata-based methods [7], matching tree-based methods [8], directed graph-based methods [9] and Petri net-based methods [10]. In the actual application scenarios such as manufacturing process, these objects such as materials, products, personnel and location are composed of many logical attributes. In addition, these objects always will extend some other attributes at any time as needed. When determining the concerned complex events, the user must decide several attributes and make their range as the detection rules. However, it is difficult that how to directly determine which kinds of primitive events are interested in, then we need to detect the attributes of each primitive event to determine whether it meets the rules.

Focusing on multi-attribute detection scenarios for RFID primitive events, this paper proposes a Rete-based RFID complex event detection method called R^2CEDM, which expands attributes of primitive event. The aim of this method is to do as follows:

(1) To extend object attributes for primitive events and do complex events detection. Extending all their attributes of the primitive events, and then transferring them to do complex event detection.
(2) To create an event detection network. Firstly, finding out the primary component events with detecting the attributes of input events. Secondly, generating parent events continually from Rete network. Finally, outputting the target complex events.

2 RFID Complex Event Detection Model

The primitive event is defined as PE = <OID, RID, T> [2] in traditional studies, but the definition is unsatisfied in actual application environment and cannot be used directly in complex event detection. Take a manufacturing workshop for example, managers are interested in all events which batch number is “M20160923”. When the complex event detection system receives a primitive event <O1, R1, t> , it can’t determine whether the object O1 belongs to the batch of “M20160923”, unless we query all objects of the batch “M20160923” in the database.

In most scenarios, the complex event detection mainly focuses on those events, which are about a type of objects or happen in a region or period, rather than an individual, which is uniquely labeled by an OID. Moreover, the biggest drawback about focusing on the individual will generate a huge number of complex events, which are not all interested by the actual application. Therefore, in order to be more suitable for RFID complex event detection system in real scenario, this paper proposes expanding primitive event to expansion primitive event.

Definition 1: Expansion primitive event. EPE = <TypeID, O, R, L, T> , where TypeID is an identification of the expansion primitive event. O, R, L and T are defined as follows.

Definition 2: Monitor object. O = {OID, a1, a2, …, an}, where O denotes a finite attribute set of RFID tag object, and its content can be increased or decreased on demand. OID denotes the object's unique identification (OID as a default tag data), and a1, a2, …, an denote the other attributes of the monitor object. OID is the key of this data field, expressed as O.ID = OID.

Definition 3: RFID reader. R = {RID, ReadPointID, a1, a2, …, an}, where Rdenotes a finite attribute set of RFID reader, and its content can be increased or decreased on demand, RID denotes a unique identification of the reader, ReadPointID is used to associate the incident locations of events, and a1, a2, …, an express other attributes of the RFID reader. RID is the key of this data field, expressed as R.ID = RID.

Definition 4: Incident location of event. L = {ReadPointID, BusinessLocationID, a1, a2, …, an}, where L denotes a finite attribute set of incident location of event, and its content can be increased or decreased on demand. ReadPointID denotes the identification of physical reading position. BusinessLocationID denotes the location identification specified by the upper system. And a1, a2, …, an denotes other attributes to incident location of event. ReadPointID is the key to incident location of event, expressed as L.ID = ReadPointID.

Definition 5: Event time. T = {Timestamp, a1, a2, …, an}, where T denotes a finite attribute set of event time, and its content can be increased or decreased on demand, Timestamp is defined in primitive event PE, and a1, a2, …, an denotes other attributes of event time. Timestamp is the key to event time, expressed as T.ID = Timestamp.

In the definition of PE, the location of fixed RFID reader is treated as the incident location of event, but actually, this has the following three questions:

(1) There may have multiple readers at the same location.
(2) Without being specified explicitly, the reader located between two areas cannot judge where the object with tag enters into.
(3) The reader is moving in control scene, so it may read the tag in many locations not just one.

Therefore, we propose to define L separately. In practical applications, to solve the above problems is to read the "location tag". When the reader's location changes, the reader can read the location tag, which is previously fixed on a specified object, to update the associated relationship between RFID reader and incident location of event.

We can do complex event detection after expanding PE to EPE for the processing object. The R^2CEDM model is shown in Fig. 2.

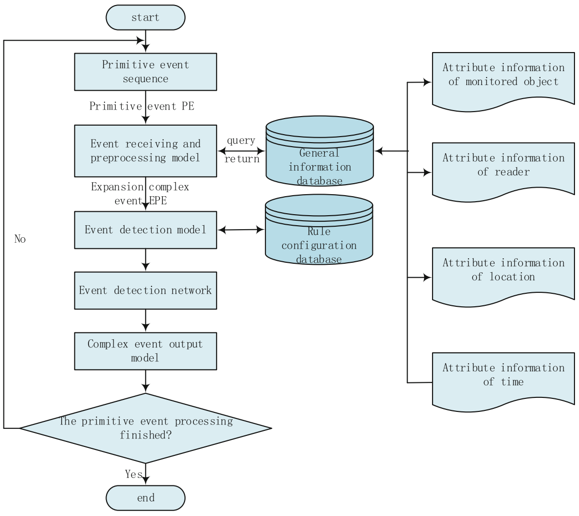
start
Primitive event sequence
Primitive event PE
Event receiving and preprocessing model
query
return
General information database
Expansion complex event EPE
Event detection model
Rule configuration database
Event detection network
Complex event output model
The primitive event processing finished?
No
Yes
end
Attribute information of monitored object
Attribute information of reader
Attribute information of location
Attribute information of time

Fig. 2. RFID complex event detection model

3 R^2CEDM

The complex event consists of component events according to certain rules, so it is suitable to Rete algorithm, which is widely used for rule-based system. The main idea of Rete algorithm is as following: Firstly, to do routine detecting in α net. Secondly, to do detection and combination to these data arriving β net. Finally, to output rules which are triggered. EPE is also need to detect attributes and judge primary component event where it belongs at first, do combination to triggered primary component event with rules, and generate triggered target complex event at last. Consequently, we propose a Rete-based RFID complex event detection method.

3.1 R^2CEDM Network

R^2CEDM network consists of two stages, attribute detection and event logic relationship detection. Attribute detection, called α net, detects attributes of O, R, L and T in EPE to determine whether it matches the attribute constraint of the primary component events of target complex event. Event logic relationship detection, called β net, detects to determine whether it matches the logic relationship between primary component events of target complex event for the EPE from α net.

Let's take the event rules of SASE language [11] described as Table 1 for example to illustrate and show the structure of R^{2}CEDM network shown in Fig. 3.

Table 1. Example of SASE described target complex event

	Event_Description	Remarks
Rule 1	**EVENT** ANY(2;A,B,C) **WHERE** (A.O.model = B.O.model)∪(B.O.model = C.O.model)∪(A.O.model = C.O.model)	ANY(2;A,B,C) represents the complex event that any two occur in events A,B and C.
Rule 2	**EVENT** OR(A,B,C) **WHERE** A.L.BLid='assembly line'∪B	
Rule 3	**EVENT** SEQ(AND(A,B),C) **WHERE** (A.O.size > B.size)∩C.O.model='TX125' **WITHIN** 10 min	
Rule 4	**EVENT** NObserve(AND(A,B);3)	
Rule 5	**EVENT** TNObserve(A;6;t_{begin}, t_{end})	

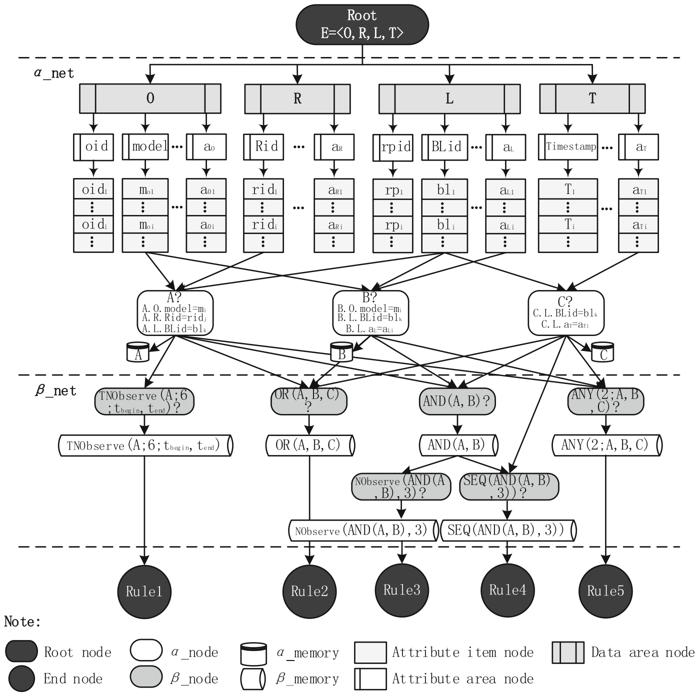

Fig. 3. The example of R^{2}CEDM network

Because there need to record and transfer intermediate results of detection in β net, we define an event_token structure to illustrate R^2CEDM algorithm.

Definition 6: Event_token. It is an event flag to record and transfer intermediate results in α_memory and β_memory. Event_token = (Component_event_List, Event_Description), where Event_Description denotes complex event expression with SASE language [11], Component_event_List denotes recording the entering component events and their attributes in WHERE clause of Event_Description and discarding attribute values which are unconcerned.

3.2 The Steps of R^2CEDM Algorithm

As shown in Fig. 4, the steps of R^2CEDM algorithm are as following:

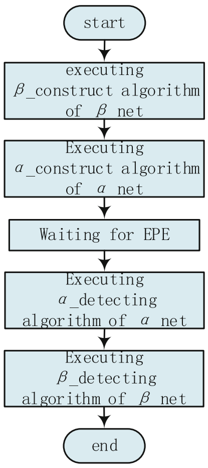
start
executing β_construct algorithm of β net
Executing α_construct algorithm of α net
Waiting for EPE
Executing α_detecting algorithm of α net
Executing β_detecting algorithm of β net
end

Fig. 4. R^2CEDM algorithm flow chart

(1) To execute β net to construct β_construct: querying rule configuration database, constructing β net, and generating β_node.
(2) To execute α net to construct α_construct: constructing α net according to attributes of O, R, L and T in information database, generatingarea_node, attribute_-node and value_node, establishing connection between value_node and corresponding α_node.
(3) To execute α_Detecting algorithm of α net: waiting the input of EPE, calculating primary complex events triggered by EPE.
(4) To execute β_Detecting algorithm of β net: waiting event_token output by α_node, calculating whether there are rules to be triggered.

The difference between Rete algorithm and traditional RFID complex event detection is as following: Every input processed by Rete is statement and has a single attribute that has a unique value. However, RFID events need to detect every attribute of each data area, compare every value to determine which type of event it belongs to. Based on the comparison above, it is important for R^2CEDM to optimize α_Detecting.

3.3 β_Construct Algorithm

β_construct algorithm works as follows: Taking out the rule from rule database one by one, constructing β net, α_node and α_memory according to EVENT clause, WHERE clause and WITHIN clause of the rules, and determining the event_token structures of α_memory and β_memory.

3.4 α_Construct Algorithm

3.4.1 Algorithm Description

Process description: Creating αttribute_node and vαlue_node according to the attribute which appears in the event_token's WHERE clause of α_node, and establishing a connection between α_node and vαlue_node.

Initialization:

(1) Generating root_node and 4 αrea_node of O, R, L and T respectively.
(2) Constructing Hash mapping between root_node and αrea_node.
(3) Using constraint to represent a comparison condition of WHERE clause.

3.4.2 Algorithm Steps

Step 1: To take out the next unprocessed _node $\alpha_i (1 \leq i \leq 4)$. If $\alpha_i = \text{null}$, it represents all α_node have been disposed, then go to step 6. Else, go to step 2

Step 2: To take out the next unprocessed $\text{constraint}_j (1 \leq j \leq \alpha_i$.compare_condition_count) in α_i. If $\text{constraint}_j = \text{null}$, it represents all attribute constraints of α_i have been disposed, i++, then go to step 1. Else, go to step 3

Step 3: To take out an unprocessed attribute constraint constraint_j in α_i. If the attribute of constraint_j is not in α net which has constructed, we need to construct a new αttribute_node required by constraint_j. Construct Hash mapping between αttribute_node and corresponding αrea_node, and let αttribute_node.α_relate_set = $\{\alpha_i\}$. Else, let αttribute_node.α_relate_set = αttribute_node.α_ relate_set $\cup \{\alpha_i\}$. Then, go to step 4

Step 4: If the attribute value required by constraint_j is not in α net, we need to construct a vαlue_node for the attribute value, connect α_i, construct Hash mapping between attribute_node and its vαlue_node, let value_node.α_relate_set = $\{\alpha_i\}$. Else, let value_node.α_relate_set = value_node.α_relate_set $\cup \{\alpha_i\}$. Then, go to step 5

Step 5: j++, go to step 2

Step 6: For all attribute_node, sorting the values of attribute_node.value_node_count descending, if the value of attribute_node.value_node_count is equal to attribute_node, sorting descending according to the value |attribute_node.α_relate_set|, and outputting Sequence_attribute_node to end α_construct

3.5 α_Detecting Algorithm

α_Detecting in R^2CEDM algorithm is based on bidirectional reasoning of production system: Firstly, reasoning from the attribute values of EPE to α_node. Secondly, selecting appropriate time to do verification from the alternative targets α_node set (Set_target_α), whose range has been narrowed, to the attribute values of EPE to find the triggered primary complex event quickly. This α_Detecting algorithm avoids comparing each attribute of EPE or every rule.

So, there are two problems in α_Detecting: One is how to greatly reduce the number of target α_node by less comparison times. The other is when reverse reasoning starts.

For the first problem, we detect an attribute value of EPE in α net. To choose a corresponding αttribut_node to do Hash mapping for an attribute value of EPE to see if it can hit the corresponding of the αttribut_node. Assuming the hit rate of each value_node is roughly equal. If we compare the attribute value of this αttribut_node in EPE at first, the greater the number of value_node, the greater the probability of being hit, and the more α_node related by value_node which is not hit being ruled out at the same time. Even if not being hit, it also can directly rule out all α_node related by this αttribut_node. So the target scope is narrowed quickly. If we compare those αttribut_node whose quantity is small in value_node at first, not only the smaller the hit rate, but also the less the irrelevant α_node being ruled out. The target scope is narrowed slowly. By the same token, it also can narrow the scope quickly to compare this attribute item at first whose |attribute_node.α_relate_set| is greater. So α_detecting compares those αttribut_node that the quantity of αrewaa_node is greater and the value of |attribute_node.α_relate_set| is bigger at first. In addition, the ID value of a data area can uniquely determine all the other attribute values of this data area, therefore, if the ID attribute value of an EPE is hit, the EPE's other attributes can no longer be detected. This can greatly avoid unnecessary comparison and reduce detection times.

For the second problem, the paper proposes that we should set different inputs to different rules and amend the opportunity according to the actual situation. It is not always high efficiency to set a fixed bidirectional reasoning conversion opportunity. So we use configurable parameters δ in α_Detecting algorithm to set when the event detection process satisfies the Eq. (1), we can start backward reasoning. In Eq. (1), α_node.wait_fulfil_constraint denotes the number of constraint to be compared in α_node of detection process, and EPE.wait_check_attribute_node denotes the number of attributes to be compared in EPE of detection process. That is to say, when the number of attributes to be compared in default rules is less than the number of input attributes in EPE, we can start backward reasoning.

$$\sum_{m=1}^{|\mathrm{Set\_target\_}\alpha|} \alpha\mathrm{\_node}_m\mathrm{.wait\_fulfil\_constrait} < \frac{1}{\delta} * \mathrm{EPE.wait\_check\_attribute\_node}(\delta \geq 1) \quad (1)$$

α_Detecting Algorithm description is as following.

Process description: according to the sequence of attribute nodes in Sequence_attribute_node ,detecting the attributes of EPE in the sequence to narrow the target scope, when the Eq. (1) is satisfied, reversely detecting attributes of EPE putting the target α_node as the starting point to make those α_node active.

Input: every attributes of EPE.

Output: Set_target_α, whose event_token is output by the α_node to corresponding β_node.

Initialization:

(1) To let α_net.α_node_count denote the total number of α_node in α net. EPE.wait_check_attribute_node = |Sequence_attribute_node|, where |Sequence_attribute_node| denotes the number of elements in the attribute of the queue.

(2) To let $\mathrm{Set\_target\_}\alpha = \sum_{j=1}^{\alpha\mathrm{\_net.}\alpha\mathrm{\_node\_count}} \{\alpha_j\}$ denote the set of alternative target α_node, where |Set_target_α| denotes the number of elements of alternative target α_node.

(3) To let α_node.constraint_count denote the number of attribute constraint condition of α_node.

(4) To let SEQ_wait_check_attribute_node denote the attribute_node queue to be compared and SEQ_wait_check_attribute_node = Sequence_attribute_node.

(5) To generate the ID attribute queue ID_node_list according to the sequence of the ID attribute in each data area of O, R, L, T in SEQ_wait_check_attribute_node.

3.6 β_Detecting Algorithm

Because β_Detecting algorithm is not the emphasis of R^{2}CEDM, we simply expound the work process as follows: To β-node whose input comes from α_node, to detect the constraints of EVENT clause, WHERE clause and WITHIN clause of β-node. To record the input of the satisfied constraint. If the event_token in β-node is activated, outputting down to trigger the father β-node to do the same detection, and traveling all β-node which has received input until the target complex event is output.

4 Experiment

R^{2}CEDM is for multi-attribute detection scenes of RFID primitive event, and optimize algorithm for event detection at the primary complex events stage of attribute detection. So one of the core of R^{2}CEDM is α_detecting algorithm. Generally, the multi-attribute procession is not considered by the complex event processing mechanism. Therefore, we treat each EPE to be detected as a simple primitive event in the experiment. In this condition, we do the performance test for optimized α_detecting algorithm, and compare it with non- optimized α_detecting algorithm. Experimental platform are as follows: Inter (R) Core (TM) 2 T5500 1.66GHZ, 1.5 GB RAM. Windows XP. JRE1.6.0 compiler environment.

To set there are 200 attribute items at most in the data areas of O and R of EPE, 80 and 20 attribute items in the data areas of L and T of EPE respectively. 60 primitive complex events are averagely divided into 3 groups A, B and C, that each contains 4, 3, 2 attribute constraints and has 20 events respectively. Group A is divided equally into 5 parts, that each contains 4 to 0 attribute constraints and has 4 events respectively. Group B is divided equally into 4 parts, that each contains 3 to 0 attribute constraints and has 5 events respectively. Group C is divided into 3 parts, and each has 5, 5 and 10 events. To set each attribute has the range of 100 discrete attribute values, meanwhile to randomly set the constraint value of primary complex events.

As shown in Fig. 5, the optimized α_detecting algorithm has certain advantage comparing to non-optimized α_detecting algorithm, and with the increase in the number of expansion primitive events, the advantage of optimized α_detecting algorithm in efficiency increases gradually. The experiment result shows that α_detecting algorithm has the feasibility, and has improved the computation efficiency.

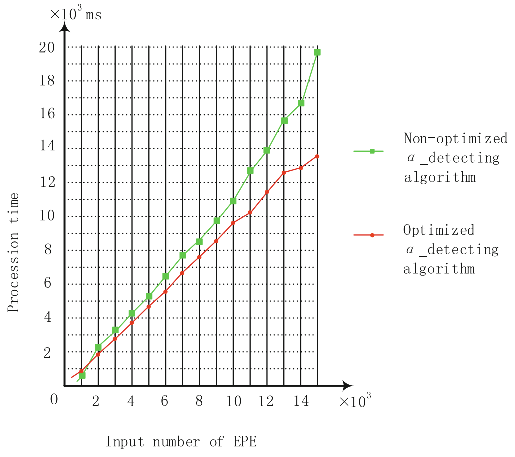

Fig. 5. The comparison of optimized and non-optimized α_detecting algorithm

5 Summary

With the wide application of the RFID technology in various industries, the efficiency of RFID complex event processing will be a key issue. Taking manufacturing enterprise environment as the background, this paper proposes a Rete-based RFID complex event detection method called R^{2}CEDM, using expansion primitive event and Rete algorithm to do complex detection. The core of the R^{2}CEDM is α_Detecting. On the attribute detection stage of R^{2}CEDM, we use the method of rapid narrowing the scope of the target nodes and bidirectional reasoning to improve the efficiency. Experiment shows that the optimized algorithm have improved the processing efficiency and reduced the processing time.

Acknowledgments. This work was supported by the National Science Foundation of China (61203135), National High-tech R&D Program of China (2015AA015308), China Post-doctoral Science Foundation (2014T70852), Chongqing Postdoctoral Science Foundation Project (Xm201305), Sichuan provincial education department (15ZB0327) and the general project of Sichuan University of Arts and Science (2015TP002Y).

References

1. Barenji, R.V., Barenji, A.V., Hashemipour, M.: A multi-agent RFID-enabled distributed control system for a flexible manufacturing shop. Int. J. Adv. Manuf. Technol. **71**, 1773–1791 (2014)
2. Yao, Z.L., Zhang, H., Yong, L.W.: RFID complex event processing: applications in real-time locating system. J. Int. J. Intell. Sci. **2**, 160–165 (2012)
3. Cugola, G., Margara, A.: Processing flows of information: from data stream to complex event processing. In: ACM International Conference on Distributed Event-Based Systems, pp. 359–360. ACM Press, New York (2011)
4. Yao, W., Chu, C.H., Li, Z.: Leveraging complex event processing for smart hospitals using RFID. J. Netw. Comput. Appl. **34**, 799–810 (2011)
5. Bok, K.S., Yeo, M.H., Lee, B.Y., et al.: Efficient complex event processing over RFID streams. Int. J. Distrib. Sens. Netw. **2012**(1550–1329), 53–62 (2012)
6. Wang, F.S., Liu, S.R., Liu, P.Y.: Complex RFID event processing. Int. J. Very Large Data Bases **18**, 913–931 (2009)
7. Nie, Y., Li, Z., Chen, Q.: Complex event processing over unreliable RFID data streams. In: Du, X., Fan, W., Wang, J., Peng, Z., Sharaf, Mohamed A. (eds.) APWeb 2011. LNCS, vol. 6612, pp. 278–289. Springer, Heidelberg (2011). doi:10.1007/978-3-642-20291-9_29
8. Yeh, M.K., Jiang, J.R., Huang, S.T.: Four-ary query tree splitting with parallel responses for RFID tag anti-collision. Int. J. AD Hoc Ubiquit. Comput. **16**, 193–205 (2014)
9. Lange, S., Donges, J.F., Volkholz, J., et al.: Local difference measures between complex networks for dynamical system model evaluation. Multi. Sci. **10**(6), e0129413–e0129413 (2015)
10. Wang, F., Liu, S., Liu, P., Bai, Y.: Bridging physical and virtual worlds: complex event processing for RFID data streams. In: Ioannidis, Y., Scholl, Marc H., Schmidt, Joachim W., Matthes, F., Hatzopoulos, M., Boehm, K., Kemper, A., Grust, T., Boehm, C. (eds.) EDBT 2006. LNCS, vol. 3896, pp. 588–607. Springer, Heidelberg (2006). doi:10.1007/11687238_36
11. Wu, E., Diao, Y., Rizvi, S.: High-performance complex event processing over streams. In: ACM SIGMOD, pp. 407–418. ACM Press, Chicago (2006)