July-August 2020
ISSN: 0193-4120 Page No. 6351 - 6363

Detection and Processing of RFID Internet of Things Complex Events Combing Event Clustering Algorithm

Xing Zhang[1,*]
[1]Nanyang Medical College, Nanyang, China, 473000

Article Info
Volume 83
Page Number: 6351 - 6363
Publication Issue:
July-August 2020

Article History
Article Received: *25 April 2020*
Revised: *29 May 2020*
Accepted: *20 June 2020*
Publication: *10 August 2020*

Abstract

The development of radio frequency identification (RFID) technology has made it more convenient for the perception of the world. In this paper, based on the features of RFID data, data modeling is carried out with the events as the focus. Various rules and patterns are used in a multi-constraint environment to detect specific complex events effectively. Through the analysis and processing of RFID events, the detection of abstract underlying RFID events is used to monitor and process complex events efficiently. According to the event classification and detection of RFID data in the Cascadia system, the event detection method based on the application of the event parsing diagrams is presented, and an event clustering algorithm is put forward. Combined with the specific application instances, the RFID complex event detection model based on the application of the event parsing diagrams is explained and demonstrated. The experimental results suggest that the RFID complex event clustering algorithm proposed in this paper can meet the accuracy requirements relatively well and improve the overall efficiency of the RFID system.

Keywords: *Radio Frequency Identification (RFID), Complex Event Detection, Event Clustering Algorithm, Detection Algorithm;*

1. Introduction

Radio frequency identification (RFID) technology is an automatic identification and acquisition technology for non-contact and two-way wireless communication by means of radio frequency. Compared with traditional barcodes, it has high speed, real time, reusability, high penetrability, strong environmental adaptability, large data capacity, and many other advantages[1]. RFID technology is extensively applied in military, communications, transportation, medical care, service, and other industries. As a core supporting technology, RFID data processing has no longer been limited to the applications in the technical field. However, it has formed a broad economic value chain system. Studies on RFID data processing technology in academia have presented a surge in recent years, and the research contents have covered the underlying perception technology, the related application fields, security, policy, and issues in multiple other aspects[2]. For example, various types of popular smart mobile terminals, car navigation, and so on have the function of interconnection and mutual communication. In addition, location sharing can be implemented based on GPS, Beidou, Bluetooth, RFID, and other technologies. Industry analysts predict that wireless and mobile terminals will surpass desktop computers, and mobile computing technology emerges as the times require[3]. This technology can provide diversified services to meet the demand of more users and enterprises. The perception of physics can be combined with the development of politics, such as the GPS signals and the US E911 authorization, and the continuous development of technology in infrastructure and mobile phone-based positioning technology. These technologies have been extensively used in many application fields based on urban planning and indoor maps, as well as airports or MALL. However, there are relatively few studies on the modeling and processing of RFID spatial and temporal correlation data. In particular, data modeling for RFID has yet to be perfected[4-5]. The primary reason is that the data processing part of RFID is relatively complicated and that the reusability and scalability of the application software are quite low. The development trend at present is to provide a platform based on RFID applications to establish convenient connections between the physical world and the logical world[6].

At present, most RFID platforms mainly focus on the filtering and collection of RFID underlying data, such as the Weblogic RFID Edge Server, the Java System RFID Software, and so on[7-8]. RFID platform is an approach to implement the ALE (Application level events) standard[9]. The ALE proposed by the EPC (Electronic Product Code) Global Organization as the RFID standard also focuses on the underlying data without attaching the semantics of the EPC data. From the data at the ALE level, various types of RFID events can be extracted [10-11]. In the early development of RFID, the data were transmitted directly to the application program. The application program was responsible for interpreting the source data as the logical business data [10]. Based on this data-centric approach, the traditional database technology is used to establish the model of RFID data and save the data in the DBMS. This method is highly dependent on the DBMS, and the development of the event processing mechanism needs to expand the DBMS, which has relatively low adaptability[12]. Through the above analysis, it can be known that the RFID data processing technology focuses more on high-level information processing, including the custom functions such as EPC data conversion and so on. However, as a large number of implicit semantics is contained in the conversion from the original EPC data to the EPC business information, the detection of the underlying atomic events has become particularly important[13-15].

The existing studies on the detection of complex events in multiple constraint environments mainly focus on data detection. However, the capacity to establish models for the events with rich semantic space-time data is relatively weak. In this paper, the event model of RFID data processed by the Cascadia system is first referenced. The event clustering algorithm is combined to put forward three basic constraint environments and summarize the operations of six types of events in three classes. Subsequently, the definition of the event parsing tree, as well as the instances where the event parsing tree and the event parsing diagram are used for event detection, are provided. Based on the event parsing diagram, a complex event detection algorithm that contains various event operations in different constraint environments is put forward.

2. RFID event processing model and algorithm

The primary purpose of RFID event processing is to detect events, that is, to detect specified events according to the given rules and patterns. In the next section, two solutions are introduced: the data-centric approach and the event-centric approach. The formal event detection model is an essential foundation for the verification of the validity in event detection, which is also the theoretical guarantee for the implementation of the RFID technology.

The data-centric approach is the earliest method that was adopted in RFID processing systems. The traditional database technology is used to establish the model of the RFID data, and the data are stored in the database. The event detection is supported on the basis of the DBMS. One of the representative systems is the Siemens RFID system. In the RFID system by Siemens, a temporal mode-oriented ER model is put forward. The model can be used not only to describe the features of RFID data, but also to express business logic, and support shard-based storage. The rule-based framework provides automatic RFID data filtering, conversion, and aggregation to generate high-level semantic data. The rules can be used to detect complex events based on DBMS.

The event-centric approach is a higher-level RFID event processing technology where complex events are directly handled with high efficiency through the establishment of the event model. There is some relevance of the event detection technology and data flow system technology in the active database to this technology. Among them, active event modeling and active complex event processing methods can provide a study basis for the event-centric RFID event processing technology. However, as the features and semantic complexity of the data stream are not taken into consideration in the processing technology of the active database, and the active database mainly supports static event analysis, it cannot be directly applied to RFID data processing. On the other hand, in the RFID data processing, the correlation and limitations between different events on the same event stream are mainly considered, which is different from the simple event filtering in the Publish / Subscribe system and relational algebra-based processing in the data stream processing. Hence, it is necessary to redesign a highly efficient, real-time, and incremental event processing mechanism that can be accomplished in memory.

The detection of the underlying complex events of RFID is a vital support link to support the detection and management of complex events at high levels. The event model of Cascadia is the core of its system and a bridge that connects the API and RFID infrastructure. In fact, the RFID data processed by the Cascadia system contains spatial and temporal data. According to the modeling of the location model and entity model, six types of event operation primitives in three classes can be abstracted from the event models. (1) with and without: stand for the adjacency relationship between entities; (2) inside and outside: stand for the inclusion relationship between entities; (3) near and far: stand for the relationship between the distance and range between entities, with fuzzy semantics.

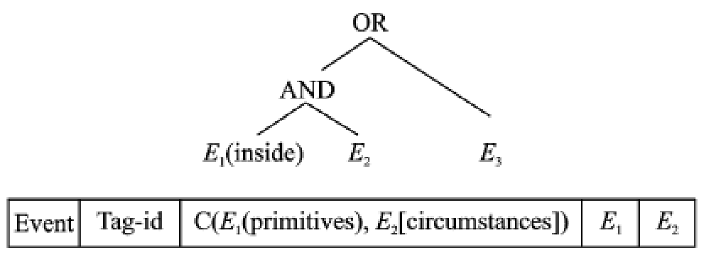
Through the analysis of the query semantics of the aforementioned typical event model, it can be seen that mainly the location and spatial relationship of the events are taken into consideration. All these operations have defined the spatial abstract position of the single-point event when the event occurs. However, in practical applications, entity tags can have mobile features. Hence, there are a huge number of queries that contain the queries of semantics such as “AND”, “OR”, “simultaneous”, and so on. In addition, the existing studies are not focused on further complex operations of complex events, especially in the case whether two complex events coincide or contain such complex semantics is determined. Moreover, for the features of RFID complex events detection mentioned earlier, very few models can implement comprehensive detection.

3. Event Analysis Diagram

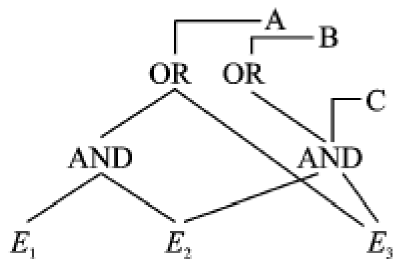
The grammatical rules of RFID event detection itself are not complicated, which is similar to the ECA rules in active databases. The rule defines the entire behavior of the event execution. In addition, whenever the detection system identifies an event, it starts to detect whether there are associated conditions. However, the semantic rules of RFID complex events are relatively abundant. Different from the method for detecting complex events in the Petri net model, in the application of the event parsing tree in the detection of complex events, common subexpressions of complex events can be merged so that multiple events can share common event nodes.

Definition 1: The event parsing tree is composed of leaf nodes, parent nodes (branch nodes), and edges. Among them, the leaf nodes stand for the atomic events, while the parent nodes are corresponding to the complex events. Other than the root nodes, each node has a pair of edges, which can be divided into an input edge and an output edge. The operations of events are propagated along the edges from the leaf nodes to the root nodes.

Figure 1 (a) shows an example of the structure of an event parsing tree and leaf nodes; Figure 1 (b) shows an event parsing diagram that is composed of an event parsing tree.



(a) Event parsing tree



(b) Structure of the event parsing diagram

Figure 1-2. Example of the complex event detection.

The construction of the event parsing tree complies with the following rules. It is assumed that the RFID rule set in the event detection process is $R=\{r_1,r_2,\cdots,r_n\}$, and its construction process is as the following:

(1) Rule 1: A subtree is established for each ruled event. That is, by defining the rules for each $r_i \in R$, a global event parsing tree is constructed on this basis, in which the leaf nodes of the global parsing tree stand for the atomic events, and the middle part nodes stand for the complex events. The entry edge and the exit edge connect the atomic events with the complex events, or the complex events with the complex events. The root node of T_i stands for the complex event part defined by the rule r_i.

(2) Rule 2: Due to the bottom-up principle, the propagation interval condition is limited. For each event parsing tree T_i, $E_i \in T_i$, the interval limit is propagated by E_i to all its child nodes. The time interval for the existence of complex events is always longer than that of its constituent events. Similarly, the upper nodes must also be present after the construction of the child nodes. Hence, in the detection algorithm, if there is a time sequence (such as Conjunction and Sequence) constraint limit operator, the constituent events are carried out in chronological order; otherwise, it is only required to accumulate the time period.

(3) Rule 3: After multiple event parsing trees have been formed, for each T_i formed, the common subtree parts are combined to form a new event parsing tree. In this way, multiple detections of branch events of the same composition can be avoided so as to improve the spatial and temporal utilization.

According to different constraint environmental events, event operation primitives are executed in three basic constraint environments in time as the following: ① Conjunction: The detection of simultaneous or nearly simultaneous events is taken as a complex event. ② Duration: The process of continuing an event over a period of time, which has the cumulative effect of events. ③ Sequence: The sequence of events at the time of occurrence, which has no cumulative effect of events. The Cascadia system can detect uncertain RFID events through a probabilistic event extractor. However, no semantic analysis and detection algorithms are provided in the operation of the events under different constraint environments for complex events. Hence, through the improvement of the three basic models of the Cascadia system, the event parsing tree is used to represent complex RFID events and further implement the algorithm of complex RFID detection based on the event parsing diagram.

The Conjunction constraint environment falls in the category of aperiodic operation. The monitoring events and termination events are different events, while for the Duration and Sequence operation, they are the same event. Based on the different semantics, the three constraint environments can be further divided into the following operations:

(1) Conjunction constraint environment

①AND operation: The conjunction of events E_1 and E_2 is denoted as $E_1E_2\Delta$. When E_1 occurs and E_2 occurs, no matter in what order the events occur, $E_1E_2\Delta$ always occurs. E_1 and E_2 can be taken as the initial event and the termination event. The complex events can be formally expressed as:

$$E_1E_2\Delta = \exists t\ ,$$

$$t'\left(t_1 \le t \le t_2 \wedge t_1 \le t' \le t_2 \wedge \left(E_1\left[t_1,t\right] \wedge E_2\left[t',t_2\right] \vee \left(E_1\left[t',t_2\right] \wedge E_2\left[t_1,t\right]\right)\right)\right) \tag{1}$$

②OR operation: the extraction of events E_1 and E_2, and the complex events is denoted as $E_1E_2\Delta$. When E_1 occurs or E_2 occurs, $E_1E_2\Delta$ occurs. Similarly, E_1 and E_2 can be taken as the initial event and the termination event. The complex events can be formally expressed as:

$$E_1E_2\Delta = E_1\left[t_1,t_2\right] \vee E_2\left[t_1,t_2\right] \tag{2}$$

③ NOT operation: It is assumed that there are events E_1, E_2 and E_3. In the closed time interval composed of the end time of E_1 and the initial time of E_3, the NOT event occurs when it is detected that the event E_2 has not occurred. Hence, it can be expressed as:

$$\left(\neg\left(E_2\right)\left[E_1,E_3\right],\left[t_1,t_2\right]\right) = \left(E_1,t_1\right) \wedge \left(E_3,t_2\right) \wedge \neg\left(E_2,\left[t_1,t_2\right]\right)$$

(3)

It should be noted that the semantic expansion of Conjunction has become different from the conjunction semantics represented by the general active rules. Hence, it can perform more comprehensive event detection on RFID data reads in real time for a large number of events.

(2) Sequence constraint environment

In addition to the above three operations, the Sequence constraint environment also includes two special operations:

① The non-cumulative non-periodic event operation is denoted as A, which indicates that the two events E_1 and E_2 occur in sequence. When E_2 occurs, E_1 has already occurred, and the Sequence event occurs. This implies that the end time value t of E_1 is less than the initial time value t' of E_2. E_1 is the initial event of the sequence event and E_2 is the end event. For example, $\left(E_1,E_2,E_3,A\right)$ indicates that a complex event occurs only if each occurrence time of E_2 falls within the time interval formed by the end time of E_1 and the initial time of E_3. Its complex events can be formally expressed as

$$E_1E_2A = \exists t\ ,\ \ t'\left(t_1 \le t < t' \le t_2 \wedge \left(E_1,\left[t_1,t\right] \wedge \left(E_2,\left[t',t_2\right]\right)\right)\right) \tag{4}$$

② The operation of non-cumulative periodic events is denoted as P, which indicates that the events occur periodically. Its complex events can be formally expressed as

$$\left(E_1,T,E_3,P\right) = \exists t' < t \exists i \in Z^+\left(t = t' + T_i \wedge \left(E_1,t'\right) \wedge \neg As\left(E_3,\left[t'+1,t\right]\right)\right) \tag{5}$$

The complex event can be expressed as $\left(E_2,T,E_1E_3,P\right)$, and T is a periodic expression. Periodic events occur once every other period T within the closed time interval formed by the initial time of E_1 and the end time of E_3. E_1 is an initial event, E_2 is a monitoring event, and E_3 is an end event.

(3) Duration constraint environment

Similarly, in addition to the three operations, including Conjunction, there are also two special operations included in the Duration:

① The accumulated aperiodic event operation can be expressed as A*, and the aperiodic operator can express the occurrence of an aperiodic event within a closed period of time. Cumulative aperiodic events can be expressed as E in $\left(E_1,E_2,E_3,A*\right)$. All occurrences of E_2 are accumulated in the closed interval formed by E_1 and E_3 until the event E_3 occurs, then event E occurs. The formal definition of the A * operator is as the following:

$$\left(E_1,E_2,E_3,A*\right)=\left(E_2,A*\right)\wedge\exists t<t_1\left(E_1\wedge\neg As\left(E_3,[t+1,t_2]\right)\right) \tag{6}$$

② The accumulated period event operation can be expressed as P *, and the accumulated period event can be expressed as: E_1E_3P*. Different from P, P * only occurs once when E_3 occurs. Its complex events can be formally expressed as

$$\left(E_1,T,E_3,P*\right)=\exists t'<t\exists i\in Z^+\left(t=t'+n_i\wedge\left(E_1,t'\right)\wedge\neg As\left(E_3,[t'+1,t]\right)\right) \tag{7}$$

It can be seen from the above operation semantics that the significant difference between Duration and Sequence is the cumulative effect of events. The Conjunction constraint environment does not have periodic and sequential features. Hence, it does not have a cumulative effect.

The first event of 2 complex events is defined as left-leaf node events, and the second event of complex events is taken as the right-leaf node events. To facilitate the description in the subsequent event detection algorithm, we refer to the two events of AND and OR operations as the left part event (LPE) and the right part event (RPE). The middle part leaf events for Not, A, P, A *, and P * operations are referred to as middle part events (MPE).

The detection of complex events is a process of identifying the occurrence of complex events. In addition, various parameters of the events are collected and recorded during this process. As the detection algorithm of complex events in the event parsing diagram model is based on the event parsing tree, each complex event is represented by an event parsing tree, the leaf nodes of the event parsing tree are atomic events, and the middle nodes of the event parsing tree are event operators. The event parsing trees with common sub-expressions are merged to form an event parsing diagram. After the user defines the rules, the system constructs an event parsing diagram for each event expression. When an event parsing diagram is established, not only the event expression in the rule definition but also the event expression of the user should be read. The event parsing tree can implement event sharing node through the combination of common sub-expressions to avoid multiple detections of the same event, reduce data calculation, and save storage space.

4. RFID complex event detection based on event clustering algorithm

The above analysis suggests that the event parsing diagram has the robust expressive capacity and description specification, which can solve the problem of timing parameter processing properly and make up for the defects of repeated storage and detection of public event subexpressions in Petri nets. Based on the advantage of using the event parsing diagram model to detect events, a complex event detection algorithm that can match multiple operations in three constraint environments is put forward combined with the event clustering algorithm in this paper. The basic idea of the event clustering algorithm is as the following: When an atomic event occurs, the leaf node representing the event is first read, and then the leaf node passes the operation to the parent node along the edge and stores it in the corresponding event list. At this point, complex event detection is carried out. If a complex event occurs, the event entity tag is stored in the event list. A flag is set at the branch node to mark whether the complex event is a system-defined event. Where the flag is empty, it indicates that the complex event is only a member event of the system-defined event (which forms a branch node); otherwise, the occurrence of complex events is notified to the system. In this paper, the event parsing diagram model is used to provide complex event detection algorithms with multiple matching operations in three constraint environments.

(1) Complex event detection in the Conjunction environment

Firstly, the basic idea and character description of the event detection algorithm in the Conjunction environment are provided, and then its parameter transfer algorithm and complex event detection algorithm are given, respectively. The algorithm adopts the bottom-up approach from the leaf nodes to the root node. If an atomic event triggers a complex event, the node of the corresponding complex event will be marked, and the occurrence of the complex event will be reported to the system. Otherwise, no complex event will occur, and the node will not be marked.

In Algorithm 1, the processing of LPE, RPE, or LPE, RPE, and MPE under the three operations in the Con-junction environment are taken into full consideration, and event detection can be executed upon the occurrence of each event.

```
Algorithm 1 Con_Oper_Detec(node, parameter_list)
Input: node;
Output: parameter_list;/* List of the related parameter entity tag units
*/
begin
case branch_node:/* Carry on according to rules 1 and 2 and the corresponding event operations
*/
AND:
  if node is LPE, then
  if the list of $E_2 \neq o$, then
for each $e_2$ that meets ($t\_s(e_2) \le t\_s(e_1) \wedge (t\_e(e_2) \le t\_e(e_1)$) do in $E_2$
          Save ( < $e_2$, $e_1$ > , [$t\_s(e_2)$, $t\_e(e_1)$]) to this node;
        Add the newly added entity tag of e to the list
    of $E_1$;
call Detect_Compo(node, parameter_list);
if node is RPE, then
if the list of $E_1$ is $\neq o$, then
            for each $e_1$ that meets
            ($t\_s(e_1) \le t\_s(e_2) \wedge (t\_e(e_1) \le t\_e(e_2)$)do in $E_1$
            Save ( < $e_1$, $e_2$ > , [$t\_s(e_1)$, $t\_e(e_2)$]) to this node;
            Add the newly added entity tag of $e_2$ to the
            list of $E_2$;
call Detect_Compo(node, parameter_list);
OR:
for the occurrence of any event do, the event entity tag that occurs will be stored to this node do
call Detect_Compo(node, parameter_list);
NOT:
if node is LPE, then
Add the newly added entity tag of $e_1$ to the list of $E_1$;
if node is MPE, then
if the list of $E_1 \neq o$ and ($t\_e$(the earliest end time in $E_1$) < $t\_s(e_2)$), then
Add the newly added entity tag of $e_2$ to the list of $E_2$;
if node is RPE, then
if the list of $E_1 \neq o$, the
For each $e_1$ in the list of $E_1$ that meets (t_s ($e_2$) > t_e ($e_1$)) do
          For each $e_2$ in the list of $E_2$ that meets (t_e
      ($e_2$) < t_e (e1)∧t_s ($e_3$) < t_s ($e_2$)) do
          Save ( < $e_1$, $e_2$, $e_3$ > , [t_e ($e_1$), t_s ($e_3$)]) to
      this node;
call Detect_Compo(node, parameter_list);
end.
```

```
Algorithm 2 Detect_Compo(node, parameter_list)
Input: occur event;
Output: complex event;
begin
if the mark of this node is checked, if it is a system-defined event, then
          Notify the system of the occurrence of
          atomic or complex events;
          / * Output complex events here * /
if the parent node, that is, node_parent $\neq o$, then
    Based on rule 2, the child node and its parameters are copied to the parent node node_praent;
    call Con_Oper_Detec(node_praent, parameter_list);
end.
```

Algorithm description: Complex event detection in the context of Conjunction is a recursive call process, which includes two sub-algorithms: Algorithm Con_Oper_Detec and Algorithm Detect_Compo.

July-August 2020
ISSN: 0193-4120 Page No. 6351 - 6363

Among them, the case statement, for loop statement, and if judgment statement are all terminable. When an atomic event occurs, the activated leaf node first determines whether it is an event required by the system. If so, a signal will be sent to the system. Subsequently, the presence of a parent node is checked. If it does not exist, the algorithm terminates; if there is a parent node, the algorithm Con_Oper_Detec is called. The operation is determined, the detected complex event is saved in the entity tag list of this node, and the algorithm Detect_Compo is called. The whole process is judged from the bottom up, and the algorithm is terminated when the condition is no longer met, or there is no parent node. In the algorithm, the number of nodes in the entire event parsing diagram is n. All the complex events detected from bottom up are composed of binary operators, and the time complexity of the detection process is $O(\log_2 n)$. However, for other complex event detection composed of binary operators, the time complexity of the process is $O(\log_2 n)$.

(2) Complex event detection in the Sequence environment

It is required that the sequence constraint environment should be stronger in the sequence of the event composition than Conjunction. Based on the operations involved, the basic idea of the event detection algorithm in the Sequence environment is as the following. If the occurrence of an atomic event leads to a complex event, the node of the corresponding complex event will be marked, and the occurrence of the complex event will be reported to the system. In the algorithm, the nodes at LPE, RPE, or LPE, RPE, and MPE under the five operations in the Sequence environment are considered, and event detection can be performed for the occurrence of each event.

```
Algorithm 3 Seq_Oper_Detec(node, parameter_list)
Input: node;
Output: parameter_list;
begin
case branch_node:/* Carry on according to rules 1 and 2 and the corresponding event operations
*/
AND:
  if node is LPE, then
  if the list of $E_2 \neq o$ and (($E_2$head) ≤t_s ($e_1$) ∧ (t_e ($E_2$head) ≤t_e ($e_1$)), then
              Save ( < $E_2$head, $e_1$ > , [t_s ($E_2$head), t_e ($e_1$)]) to this node;
              Delete $e_1$ from the list of headers of $E_2$;
    call Seq_Detect_Compo(node, parameter_list)
    else, add the newly added entity tag of $e_1$ to the list of $E_1$;
  if node is RPE, then
  if the list of $E_1 \neq o$ and (t_s ($E_1$head) ≤t_s ($e_2$) ∧ (t_e ($E_1$head) ≤t_e ($e_2$)), then
            Save ( < $E_1$head, e2 > , [t_s (head), t_e ($e_2$)]) to this node;
            Remove $e_2$, the head node of $E_1$, from the list;
    call Seq_Detect_Compo(node, parameter_list);
else, add the newly added entity tag of $e_2$ to the list of $E_2$;
OR:
  for any event that has occurred do
    Save the entity tag of the event that has occurred to this node;
    call Seq_Detect_Compo(node, parameter_list);
NOT:
  if node is LPE, then
    Add the newly added entity tag of $e_1$ to the list of $E_1$;
  if node is MPE, then
    if the list of $E_2 \neq o$ and (t_e ($e_1$) < t_s ($e_2$)), then
      Add the newly added entity tag of $e_2$ to the list of $E_2$;
  if node is RPE, then
    if the list of $E_1 \neq o$ and (t_s ($e_2$) > t_e (head)), then
  if the list of $E_2 \neq o$, then
    for each $e_2$ in the list of $E_2$ meets (t_e ($e_2$) < t_e ($e_1$) ∧t_s ($e_3$) < t_s ($e_2$)) do
        Save ( < $E_1$ head, $e_2$, $e_3$ >, [t_e ( $E_1$ head), t_s ($e_3$)]) to this node;
        Clear the list of $E_1$ and $E_2$;
    call Seq_Detect_Compo(node, parameter_list);
  else
  Save ( < $E_1$ head, $e_2$, $e_3$ > , [t_e ( $E_1$ head), t_s ($e_3$)]) to this node;
     Remove the header node of $E_1$ from the list;
  call Seq_Detect_Compo(node, parameter_list);
A:
 if node is LPE, then
  Acquire the eid of node;
  Create a storage unit, save the time expression of eid and the corresponding $E_2$;
  Add the newly added entity tag of $e_1$ to the list of $E_1$;
 if node is MPE, then
  Acquire the eid of node;
  if the list of $E_1 \neq o$ and the eid of $e_1$ is consistent with $e_2$, then
  Create the storage unit, save the time expression of eid and the corresponding $E_2$;
  Save ( < $e_1$, $e_2$ > , [t_s ($e_2$), t_e ($e_2$)]) to this node;
  call Seq_Detect_Compo(node, parameter_list);
 if node is RPE, then
  if the list of $E_1 \neq o$, then
   if ($t\_e(e_1) < t\_s(e_3)$), then
   Delete the header node of $E_1$;
P:
 if node is LPE, then
  Acquire the eid of node;
  Create a storage unit, save the time expression of eid and the corresponding $E_2$;
  Add the newly added entity tag of $e_1$ to the list of $E_1$;
 if node is MPE, then
  Acquire eid of node;
  if the list of $E_1 \neq o$ and the eid of $e_1$ is consistent with $e_2$, then
   Create a storage unit, save the time expression of eid and the corresponding $E_2$;
   Add the newly added entity tag of $e_2$ to the list of $E_2$;
 if node is RPE, then
  if the list of $E_1 \neq o$ and (t_e ($e_1$) < t_s ($e_3$)), then
  Acquire eid of $e_1$;
   Add the eid in $E_2$ and $e_1$ that is consistent to the temporary list temp $E_2$;
   Delete the list header of $E_1$;
  if temp $E_2$ is not empty, then
   Save ( < $eq_1$, temp $E_2$, $e_3$ > , [t_s (the earliest initial moment of temp $E_2$), t_e (the last termination moment of temp $E_2$)]) to this node;
   Delete temp $E_2$;
   call Seq_Detect_Compo(node, parameter_list);
end.
```

```
Algorithm 4 Seq_Detect_Compo(node, parameter_list)
Input: occur event;
Output: complex event;
begin
 if the mark of this node is checked, if it is a complex event defined by the system, then
  Notify the system of the occurrence of the complex events; / * Output complex events * /
  if the parent node, node_parent $\neq o$, then
  Copy the node and its parameters to the parent node node_praent;
  call Seq_Oper_Detec(node_praent, parameter_list);
end.
```

Algorithm description: In the algorithms Seq_Oper_Detec and Seq_De-tect_Compo, the case statement, for loop statement and if judgment statement are all terminable, and the process is a recursive call process. When an atomic event occurs, the active leaf node first determines whether it is an event required by the system. If so, a signal is sent to the system. Subsequently, the presence of a parent node is checked. If no parent node exists, the algorithm is terminated; if there is a parent node, the algorithm Seq_Oper_De-tec is called to determine which kind of complex operator it is and save the monitored complex event to the entity tag list of this node. The algorithm Seq_Detect_Compo is called. The whole process is judged from the bottom up, and the algorithm is terminated when the condition is no longer met or there is no parent node. It is assumed that the number of nodes in the entire event parsing diagram is n. The bottom-up monitoring complex events are all composed of ternary operators, and the time complexity of the monitoring process is $O(\log_3 n)$; while for all the other complex event monitoring processes composed of binary operators, the time complexity is $O(\log_2 n)$. As $O(\log_3 n)$ is less than $O(\log_2 n)$, the time complexity of the entire algorithm is $O(\log_2 n)$.

(3) Complex event detection in the Duration environment

Similar to the execution process in the Sequence constraint environment, the algorithm is a process from the leaf node to the root node. If an atomic event triggers a complex event, the node of the corresponding complex event will be marked, and the occurrence of the complex event will be reported to the system; otherwise, no complex event will occur, and the node will not be marked. The basic idea and analysis of the algorithm are similar to those of algorithms 3 and 4. Hence, they are omitted in this paper.

5. Experiment and analysis

The complex event detection based on the Petri net model only matches the atomic events that arrive in chronological order. The spatial and temporal relationship of matching basic events is not considered in the filtering process of the traditional tree or diagram-based event detection. For example, if the left part event of a complex event does not occur, then the right part event may also be filtered out. In this case, it can not only lead to a waste of system overhead but also have a certain impact on the response time of the system. Hence, the limitation of the existing methods in practical applications is not ignorable. In the following section, the RFID event detection method of the event clustering algorithm used in this paper for application examples is described and compared with the Pe-tri net model method.

In the logistics real-time location tracking, it is necessary to record cargo handling operations. In order to express the business logic that “a batch of goods is loaded on a pallet”, a complex event E (Item (n) primitives Pallet) can be defined. Among them, primitives are the operation primitives of the RFID event processing model in the Cascadia system. Item and Pallet are two EPC event types, which indicate that the RFID reader has identified the tag of the goods and the pallet; n stands for the parameters of the goods.

In order to verify the practical function of the algorithm, the effectiveness of the detection algorithm is demonstrated from two aspects, that is, execution efficiency and validity. The experimental environment simulation environment is configured as the following: The algorithm is implemented in C ++. As the real logistics environment is not yet available, the experiment is completed in the simulation environment.

By reference to the EPCISO18000-6 standard, the number of tags should not exceed 1000, which can be selected within this range according to different experimental requirements. The efficiency result adopts the concept of Speed-up, that is, the ratio between the implementation of the event detection algorithm and the implementation of the event-free detection algorithm is introduced, which is used as the evaluation parameter value. Where the value of this parameter is not greater than 1, it indicates that the efficiency is higher than the original method of the system. The smaller the value, the higher the efficiency. In order to facilitate the testing and evaluate the efficiency of the algorithm more accurately, the execution cost of operations such as database update is not counted.

Example 1. In the context of Conjunction constraints, when a complex event E (Item near Pallet) is detected, the event may indicate that loading is about to start or the unloading has just started. If it is necessary to make a further judgment and consider whether to enter the loading and unloading state, it can be determined directly through the detection of the complex event E (Item near Pallet) E (Item inside Pallet) or (Item outside Pallet). In the Petri net model, as the entity tag represented by Token involves multiple state changes in preparation for loading and unloading, it is necessary to detect all the complex events. The application of RFID events requires the detection of the complex event operation in a Conjunction constraint environment only once.

Example 2. In a Sequence-constraint environment, physical cargo tags are detected according to the chronological and spatial distance relationship. When a complex event E (Item near warehouse) E (Item inside Pallet) E (Item far warehouse) Δ is detected, the event indicates the process where an item enters the warehouse and is loaded and unloaded until it leaves the warehouse. On the other hand, in the Petri net model, the Token has to go through at least four state changes, and the modeling overhead is considerable. It is assumed that multiple trucks repeat the task of loading and unloading. Let the above process be represented by the event E_1. As the event parsing diagram can be used to further detect new complex events from the detected complex events through new event operations, then E_1P stands for the complete periodic execution process of the event for the loading and unloading of cargo.

Example 3. In the Sequence constraint environment, the entire process is recorded in the RFID complex event detection. However, it does not have a cumulative effect. The quantity of goods handled cannot be determined. In the environment of Duration constraint, when a complex event E (Item outside Pallet E) Δ is detected, assuming that E_2 stands for the loading process, then $E_2P*(n)$ stands for the complete loading event, in which n stands for the number of loads. When a complex event E (Item with Pallet) E (Item without Pallet) ∇ is detected, it indicates that some cargo may have been missed during the loading process. For the record of such non-cyclical accidental event, it is assumed that E_3 stands for the missing process, then $E_3A*(n)$ stands for the complete missing events, in which n stands for the number of missing. On the other hand, in the Petri net model, the incremental detection of complex events is described by the position of the mark in the Petri net. For the periodic or cumulative events, it is necessary to calculate the transition guard function through Token. If the status holds, a transition is triggered and the location node is marked. Only when the last location node in the sequence is marked, it is determined that a complex event occurs. Hence, its logical relationship is complicated, which will lead to low detection efficiency.

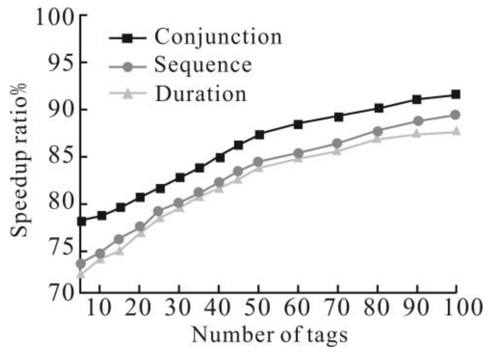


Figure 3. Comparison results of execution efficiency under three non-cumulative constraints.

Figure 2 shows that the speedup ratios in the three constraint environments of non-cumulative conditions are all less than 1, which suggests that the introduction of an event detection algorithm mechanism based on the event parsing tree has a significant effect on the efficiency of the system in completing the detection of complex events. In the three constraint environments, the execution effect of the Conjunction constraint environment is relatively less efficient. The reason is that when multiple events occur at the same time, the algorithm itself determines the underlying RFID data processing mode. In order to avoid excessive reads and missing reads, it is a better option to adopt the two modes of Sequence and Duration. In these two environments, the execution efficiency is basically the same.

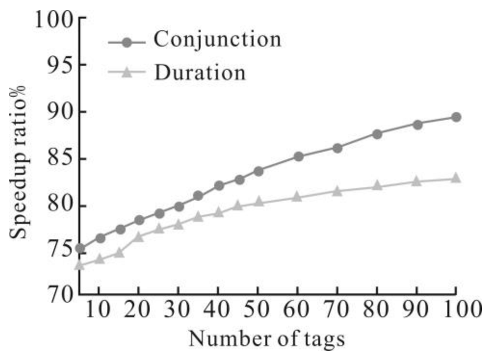


Figure 4. Comparison of execution efficiency under two cumulative constraints.

Figure 3 shows that under the cumulative conditions, the execution efficiency of the two constraint environments of Conjunction and Duration is also relatively ideal. At the beginning of the design of the detection algorithm, the cumulative situation is taken into full consideration. Hence, in the constraint environment of Duration, the execution efficiency is optimal in multiple modes.

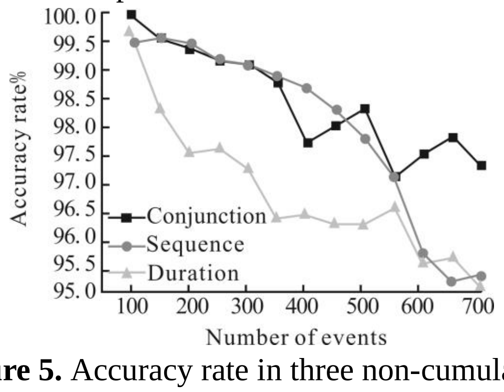


Figure 5. Accuracy rate in three non-cumulative constraint environments.

As shown in Figure 4, the results of the accuracy of detecting complex events under three non-cumulative constraints are presented. In order to verify the effect of the proposed algorithm, the number of tag events is increased by nearly an order of magnitude in the experiment. From the experimental results, it can be seen that the introduction of the RFID event detection algorithm plays a significant role in ensuring the validity of the final complex event detection of the system, which is higher than 95.1%.

The accuracy and non-cumulative conditions in the accumulated Conjunction and Duration constraint environments are similar, and their efficiency is also obtained at the cost of the event space. Hence, such cases are omitted in this paper.

According to the description above, it can be known that the RFID complex event detection method based on the event clustering algorithm can selectively process basic events while detecting complex events at the same time, where valuable complex events can be quickly generated and used repeatedly to ensure logical validity, consistency, and high efficiency. Hence, it can be applied to reducing the response delay of the RFID complex event detection systems.

6. Conclusions

The logistics management at the airports requires complex event processing capacity with high efficiency in a multi-context awareness environment. In this paper, based on the features of RFID data processing in a multiple constraint environment, data modeling is carried out with the events as the focus. Combined with the event clustering algorithm, the abstract underlying RFID events during the analysis and processing of RFID events are effectively monitored and processed in this paper. The definition of the formal event operation in the event detection is provided. Based on the event parsing tree and the event analysis diagram, various algorithms for event detection in different constraint environments are put forward. The theoretical analysis and experimental verification are performed on the event detection algorithm. However, the event modeling method adopted in this paper is based on deterministic events. In the practical applications of RFID technology, due to the presence of the data on the missing reads, excessive reads, dirty reads, and so on, other uncertain events will be generated. In the future, an in-depth study should be further conducted on the event detection model, where the real environment should be used to process complex events of the active and passive RFID tags.

References

[1] Wang J, Wang T, Cheng L, et al. An efficient complex event processing algorithm based on INFA-HTS for out-of-order RFID event streams[J]. Ksii Transactions on Internet & Information Systems, 2016, 10(9):4307-4325.

[2] Zhang, Cunji, Yao. Sensors, Vol. 15, Pages 30165-30186: Abnormal condition monitoring of workpieces based on rfid for wisdom manufacturing workshops[J]. Sensors, 2015, 15(12):30165-86.

[3] Catarinucci L, Colella R , Tarricone L . Design of Passive RFID Sensor tags enhanced by a novel logical communication procedure over LLRP[J]. Journal of Communications Software and Systems, 2017, 13(2):120-128.

[4] Zimmerman J, Fisher M . Avoidant/restrictive food intake disorder (ARFID)[J]. Current problems in pediatric and adolescent health care, 2017, 47(4):95-103.

[5] Ibrahimy M I, Motakabber S M A . Bridge scour monitoring by coupling factor between reader and tag antennas of RFID system[J]. International Journal of GEOMATE, 2015, 8(2):1328-1332.

[6] Lazaro, Antonio, Ramos, Angel, Girbau, David signal processing techniques for chipless uwb rfid thermal threshold detector detection[J]. ieee antennas & wireless propagation letters, 2016, 15(3):618-621.

[7] Jung H . A memory efficient anti-collision protocol to identify memoryless RFID tags[J]. Journal of Information Processing Systems, 2015, 11(1):95-103.

[8] Bashri M S R , Muhammad I I , Motakabber S M A . Design and development of wideband patch antenna for uhf rfid metal mountable Tag[J]. international journal of electronics letters, 2015, 5(1):82-98.

[9] Bartsch C , Weiss M , Kipper S . Multiple song features are related to paternal effort in common nightingales Evolutionary ecology and behaviour[J]. Bmc Evolutionary Biology, 2015, 15(1):115-121.

[10] Wickramasinghe A, Ranasinghe D C . Recognition of falls using dense sensing in an ambient assisted living environment[J]. Pervasive and Mobile Computing, 2016, 34(2):14-24.

[11] Albert Y Huang, Guillaume Joerger, Remi Salmon. A robust and non-obtrusive automatic event tracking system for operating room management to improve patient care[J]. Surgical Endoscopy, 2015, 30(8):3638-3645.

[12] M Cooney, M Lieberman, T Guimond. Clinical and psychological features of children and adolescents diagnosed with avoidant/restrictive food intake disorder in a pediatric tertiary care eating disorder program: a descriptive study.[J]. Journal of Adolescent Health, 2018, 6(2):7-15.

[13] Ankitha K, Ashutha K . Smart Shopping Cart Using Embedded System and Wireless Modules[J]. Recent Patents on Computer Science, 2016, 9(3):1-10.

[14] Ursula Kaupert, Kay Thurley, Katja Frei. Spatial cognition in a virtual reality home-cage extension for freely moving rodents[J]. journal of neurophysiology, 2017, 117(4):1736-1748.

[15] Zhao Y , Gao Z, Ma Y, et al. Pattern Matching Performance Analysis Based on Linear Models with Information Backscattered from RFID Tags[J]. International Journal of Wireless Information Networks, 2017, 25(11):1-7.