A Novel Method for Formally Detecting RFID Event Using Petri Nets

Jinan Sun\textsuperscript{1}, Yu Huang\textsuperscript{2,3}, Xin Gao\textsuperscript{1}, Shikun Zhang\textsuperscript{2,3}, Lifu Wang\textsuperscript{2,3}, Chongyi Yuan\textsuperscript{1,3}

\textsuperscript{1}School of Electronics Engineering & Computer Science, Peking University, 100871, Beijing, China
\textsuperscript{2}National Engineering Research Center for Software Engineering, 100871, Peking University, Beijing, China
\textsuperscript{3}Key Laboratory of High Confidence Software Technologies of Ministry of Education, Peking University, Beijing, China

{sjn, hy}@pku.edu.cn

\textbf{Abstract}—Radio Frequency Identification (RFID) provides fast collection of large volume of data and can be used to identify physical objects with unique IDs. In order to provide semantically meaningful data to different applications, RFID data need to be processed to discover user-defined complex events. We propose a Petri net-based method for the detection of complex events in RFID. A model named ED-net is introduced to specify semantics of complex events, which is also taken as the basis for the implementation of an event detector. Formal model ED-net is an extension of ordinary Petri net, providing user-defined types, functions and expressions, which are suitable for the precise description of attributes and constraints of RFID complex events, with non-temporal, temporal and parameterized constraints. Through modeling all the events to be detected in one ED-net model, we avoid multiple detections of common sub-events of different complex events. Through the experimental evaluation, we verify the efficiency of our detection method. This paper is sponsored by the National Natural Science Foundation of China under Grant No. 60803014 & the National Research Foundation for Doctoral Program of Higher Education of China under Grant No.20080011017. Yu Huang is corresponding author.

\textbf{Keywords}—RFID; Petri Net; Event Detecting

I. INTRODUCTION

Radio Frequency Identification (RFID) technology uses radio-frequency waves to automatically identify objects, which makes it possible to create a physically linked world where every object is automatically numbered, identified, cataloged, and tracked in real time. An RFID system generally consists of four parts: RFID tags, readers, middleware and application software. An RFID tag is uniquely identified by a worldwide unique ID, stored in its memory and defined by the electronic product code (EPC) standard [1]. Readers are capable of reading the information stored in RFID tags. RFID middleware systems are typically deployed between the readers and the applications in order to correct captured readings and provide clean and meaningful data to applications.

The RFID middleware plays the primary role in RFID data management. However, the characteristics of RFID data pose many challenges in the research of RFID middleware and complex event processing [2]. Firstly, the data generated from an RFID application are simple, and each RFID observation is of some form (epc, reader, timestamp); secondly, RFID data are temporal, dynamic and in large volume. They are generated dynamically and automatically, and must be processed in real time; thirdly, RFID data are inaccurate and have implicit semantics. Erroneous readings, such as missed or duplicate readings, have to be semantically processed. The information carried by RFID data also need to be analyzed to support advanced applications. Such information is often related to business knowledge and specific applications. Traditional event systems do not well support temporal characteristics of RFID events. Therefore, some novel and effective methods are needed for complex event processing in RFID technology.

The concept of complex event originated from active database. Lots of work has been done on complex event detection, both in active database and RFID applications. The work of [6] introduced a general purpose event monitoring system Cayuga, in which queries over event systems are expressed by Cayuga Event Language and implemented by Cayuga automaton. Petri nets are used for the modeling and detection of composite events for active object-oriented database system SAMOS [10]. The Snoop system [7] and EVE system [8] use graph-based approaches, and the work of [9] proposed EPS, in which a subscription tree used to process events.

Researchers also investigated specification languages and detecting methods for RFID complex events. The work of [11] proposed an event-oriented approach to process RFID data by defining various constructors, including non-temporal and temporal constructors, to express the relationship among complex events. A tree-based algorithm [13] was advanced to improve existing algorithm RCEDA in [11], decreasing the time complexity of RFID complex event detection. SASE[12] executes complex event queries over real-time streams of RFID data, and a complex event language was proposed, supporting negation operating in sequences, parameterized predicates and sliding windows. But some operator-nested complex events were not referred in this language. Zhu [14] aimed at proposing a formal descriptive language \textit{QDDC}_{\textit{ct}} for complex events, supporting quantitative complex events.

Most of existing researches lack formal semantics to describe complex events, which may bring ambiguity and confusion in expressing and understanding complex events in RFID. Attributes (start time and end time) of a complex event are not defined strictly. Another problem is that existing detection methods are mainly developed for a single complex event, instead of a set of complex events. While in an RFID application, we often have more than one complex event to be detected, and these events share common sub-events. For
complex events \( E = (E_e, E_a) \cap E_c \) and \( E_a = E_c \cup (E_e, E_a) \), they both contain a sub-event \( E_e \). The detection of event \( E_e \) will be processed twice when the detections for \( E_a \) and \( E_c \) are carried on separately. This is time-wasting and should be avoided in real-time detection for complex events in RFID.

In this paper, we propose a Petri net-based method named ED-net for the description of complex events in RFID, which is convenient for describing temporal and parameterized constraints with locality property, token-flow mechanism and combinability. Moreover, ED-net has formal semantics which can guide us to design better software.

II. THE MODEL OF RFID EVENTS

An RFID event can be either a primitive event or a complex event. A primitive event is an RFID observation, which occurs at a point of time when an RFID tag is read by an RFID reader. A complex event is an RFID observation sequence, consisting of a set of primitive events and having special semantics. The time of a complex event can be considered as the point of time it is detected, or the entire time interval. The first case is often used in active database, while it will cause logical problems [3]. Therefore, we define the time of a complex event to be a time interval. In RFID event detection, both the temporal distance between two events and the interval of a single event are critical [5].

We introduce several signs and functions which will be used later. An event type is represented by \( E \), and an event instance is represented by \( e \). \( t, t\_begin(e) \) returns the start time of an event instance \( e \); \( t, t\_end(e) \) returns the end time of an event instance \( e \); \( interval(e) \) calculates the interval of an event instance \( e \), and \( interval(e) = t\_end(e) - t\_begin(e) \); \( dist(e_1, e_2) \) calculates the distance between two event instances \( e_1 \) and \( e_2 \), and \( dist(e_1, e_2) = t\_end(e_2) - t\_end(e_1) \). The above functions are available for all the RFID events. Besides of these functions, users could define specific functions to distinguish primitive events. For example, \( group(reader) \) is used to represent to which group the reader of a primitive event belongs. Readers in the same group often have the same function; \( type(epc) \) is used to get the type of an object with tag \( epc \).

Temporal constraints need to be taken into account when describing RFID events. The work of [11] discussed series temporal complex event constructors. Constructors for complex events in [11] are classified into two categories: non-temporal complex event constructors and temporal complex event constructors. Non-temporal complex event constructors include OR(\( \lor \)), AND(\( \land \)), and NOT(\( \neg \)). \( E_1 \lor E_2 \) occurs when either \( E_1 \) or \( E_2 \) occurs. \( E_1 \land E_2 \) occurs when both \( E_1 \) and \( E_2 \) occur. \( \neg E \) is usually combined with a temporal constraint, and it occurs when no instance of \( E \) occurs during a specific time interval.

Temporal complex event constructors specify temporal constraints of complex events, including the occurrence order of sub-events, distance constraint and interval constraint. These constructors can be nested in arbitrary order to describe various complex events. However, parameters are not included to express constraints on complex events. For example, operator \( SEQ+ \) only requires one or more occurrences of an event, but the exact number of the occurrences is not specified. This might cause inaccurate description of complex events. Suppose we have a sequence of event instances: \( (e, e', e', e') \), where the superscript represents occurrence time of event \( e \). According to the semantics of \( SEQ^+ \) operator, we can tell that complex event \( SEQ^+(E) \) occurs more than once, \( (e, e'), (e', e'), (e', e', e') \) could all be seen as the corresponding sets of events for complex event \( SEQ^+(E) \). Therefore, it is inconvenient to describe complex events with exact aggregation number, such as complex event \( E \) occurs when event \( E_i \) has occurred ten times. In the following section, we will show how to model parameterized constraints with ED-net.

III. THE MODEL OF ED-NET

A. The static structure of ED-Net

Definition (ED-net) An ED-net (Event Detection Net) is a tuple \( N = (\Sigma, P, T, A, C, G, B, E) \), and \( \Sigma \) is a finite set of non-empty types, called color sets; \( P \) is a finite set of non-empty places; \( T \) is a finite set of non-empty transitions; \( A \) is an arc set, \( A \subseteq P \times T \cup T \times P \); \( C \) is a color function; \( G: P \rightarrow \Sigma \); \( G \) is a guard function, \( G: T \rightarrow \{Expr\} \), where \( Expr \) is a boolean expression; \( B \) is a body function, \( B: T \rightarrow \{Stat\} \), where \( Stat \) is a group of assignment operations; \( E \) is an arc function, \( E: A \rightarrow \{AExp\} \), where \( AExp \) is an expression whose value is a multi-set. The form of \( AExp \) could be: \( AExp = m^c \| n^v \| m^c + AExp \lor n^v + AExp \), where \( m \) and \( n \) are positive integers, \( c \) is a constant value of a specific color, and \( v \) is a variable. When \( m \) or \( n \) equals one, we just omit it in the expression. The sign ‘\( \lor \)’ means addition of two multi-sets, as introduced in [15].

Fig. 1 shows an ED-net model and its instance. The model describes a complex event of the conjunction of two primitive events, with the constraint that readers of the two primitive events are in the same group.

![ED-net model for conjunction](image)

B. Dynamic behavior of ED-net

The state of an ED-net is represented by a marking that records the number and color of tokens in each place. A
marking is defined as a function $M: P \rightarrow \{C(p)|p \in P\}$, where $C(p)_{MS}$ is the multi-set over the color set of place $p$. Take the instance of ED-net in Fig. 1 for example, its current state can be represented as a marking $M = (1^\infty(epc_a, r_1, 2350, 2350), 1^\infty(epc_b, r_1, 2200, 2200), \emptyset), M(p_1) = \{(epc_a, r_1, 2350, 2350)\}, M(p_2) = \{(epc_b, r_1, 2200, 2200)\}$ and $M(p_3) = \emptyset$ as place $p_3$ has no token.

The procedure of taking a step in an ED-net is as follows: First, check if a transition could be fired under current marking according to the firing rules introduced later. More than one transition may be enabled in one step, we randomly choose one to occur. Second, do the assignment operations according to the body of the transition, calculating the start and end time of the complex event. For example, for transition $t$ in Fig. 1, value $(2200, 2350)$ will be assigned to variable $ce$. Third, remove tokens from the input places of transition $t$ and add tokens to the output places of transition $t$. The number and color of tokens to be consumed and produced are determined by corresponding arc expressions. According to the arc expression in Fig. 1, tokens $(epc_a, r_1, 2350, 2350)$ and $(epc_b, r_1, 2200, 2200)$ are removed from $p_1$ and $p_2$ respectively and token $(2200, 2350)$ is added to $p_3$. Thus, the new marking of the ED-net is $(\emptyset, \emptyset, 1^\infty(2200, 2350))$.

Firing conditions for a transition in P/T net [16] are also available in ED-net, which means that for a transition to be fired, there must be enough tokens in all of its input places. Besides of this, guard expression of the transition in an ED-net must be satisfied. We summarize the process of deciding whether a transition $t$ could occur under marking $M$ as follows: Firstly, check if all the input places of transition $t$ are marked under marking $M$. We can see that $p_1$ and $p_2$ are both marked in Fig. 1; Secondly, if step 1 is satisfied, find a binding for variables appearing on all the input arcs of transition $t$. In Fig. 1, transition $t$ has two input arcs $p_1t$ and $p_2t$, and two variables $ae_1$ and $ae_2$ appear in the arc expressions. Under current marking, they could only be bound with $(epc_a, r_1, 2350, 2350)$ and $(epc_b, r_1, 2200, 2200)$ respectively. Multiple binding results could be found when tokens in input places are more than needed; Thirdly, Evaluate guard expression of transition $t$ under a binding established in step 2. Only when the result is true, can transition $t$ be fired. Otherwise, go to step 2 and choose another binding for corresponding variables. In case the guard expression cannot be satisfied under all the possible bindings, transition $t$ cannot be fired.

C. ED-net models for complex events

Non-temporal complex events include conjunction complex event, disjunction complex event and negative complex event. The ED-net model for a conjunction complex event of two primitive events has been shown in Fig. 1. A negative event should be associated with a time-constrained event; otherwise it is meaningless to discuss the occurrence of an event during an endless interval. We will introduce the modeling of negative event in the model for an interval-constrained complex event (WITHIN).

A model for the disjunction of two primitive events is shown in Fig. 2. For convenience, we mark the name of the event under its corresponding place in the graphical representation of the model. Once one of places $p_1$ and $p_2$ is marked, which means that one sub-event is detected, transition $t_1$ or $t_2$ will be fired, and place $p_1$ will be marked with a token of color $CE$ whose value is obtained from the corresponding primitive event. The model is also available for the disjunction of complex events; we only need to change colors of places $p_1$ and $p_2$ and variables $ae_1, ae_2$ to be $CE$.

Temporal complex events refer to events with time-constraint, including the occurrence order of sub-events, distance of two events and the interval of an event. The work in [11] has listed a group of temporal complex event constructors covering most of the detecting situations. However, parameterized constraints are not included in the description of complex events. For example, constructors $SEQ+$ and $TSEQ+$ both express the periodic occurrence of one or more occurrences of an event, but neither specifies the exact number of occurrences. The imprecise description may cause unexpected results. Also, parameters are necessary and important in the specification of a complex event.

\begin{figure}[h]
\centering
\includegraphics[width=0.8\textwidth]{fig2.png}
\caption{ED-net model for disjunction}
\end{figure}

Sequential constructor specifies that the occurrence of one event is after another. The semantics of "after" varies according to different understandings. When a user say event $E_2$ occurs after event $E_1$, all of these situations could be distinguished by the guard expression defined in the ED-net model. An ED-net Model for a sequence complex event $E_1; E_2$, and the semantics of "after" corresponds to the third situation, as the guard expression of transition $t$ is $ce_{end} < ce_{begin}$.

\begin{figure}[h]
\centering
\includegraphics[width=0.8\textwidth]{fig3.png}
\caption{ED-net model for aggregation}
\end{figure}

The modeling of distance-constrained sequential complex event $TSEQ(E_1; E_2; t_1, t_2)$ is similar to the model of sequential complex event, by changing the guard expression of transition $t$ to be $(ce_{end} < ce_{begin}) \land (ce_{end} < ce_{end} - ce_{begin} < t_1) \land (ce_{end} - ce_{end} < t_2)$, which indicates that the distance of $E_1$ and $E_2$ is bounded by $t_1$ and $t_2$. 
Fig. 3 shows an ED-net model for an aggregation complex event \( E \), which occurs when its sub-event \( E_1 \) has occurred ten times. Places \( p_1 \) and \( p_3 \) corresponds to events \( E \) and \( E_1 \) respectively. As soon as \( E_1 \) occurs, place \( p_1 \) is marked. Then transition \( t_1 \) can be fired and a token \( e \) is produced in place \( p_1 \). The number of tokens in \( p_4 \) indicates the occurrence frequency of event \( E_1 \). When the number reaches 10, transition \( t_2 \) can be fired and ten tokens in \( p_2 \) are consumed according to the arc expression \( 10^e \) on arc \( p_4 \).

We define the interval of an aggregation complex event to be from the earliest start time to the latest end time of all the sub-events. Instead of collecting all the sub-events and finding out the smallest and greatest values, we update the interval of complex event \( E \) each time a sub-event arrives, and when all the sub-events are detected, the latest value is the interval for \( E \). In Fig. 3, transition \( t_1 \) updates the interval of complex event \( E \) according to the time of past occurrences of \( E_1 \) (from place \( p_3 \) and the time of \( E_1 \)'s new arrival(from place \( p_1 \)). On the first arrival of \( E_1 \), transition \( t_1 \) will consume a token of \((\infty, 0)\) from place \( p_1 \), which is assigned on creation of the model. After the firing of \( t_1 \), token in \( p_1 \) will be updated to \( (\text{ce1.begin}, \text{ce1.end}) \), which is exactly the time of the first occurrence of \( E_1 \):

\[
\text{ce1.begin} = \min\{\text{ce1.begin}, \text{ce1.begin}\} = \min(\text{ce1.begin}, \infty) = \text{ce1.begin}
\]

\[
\text{ce1.end} = \max\{\text{ce1.end}, \text{ce1.end}\} = \max(\text{ce1.end}, 0) = \text{ce1.end}
\]

After ten arrivals of \( E_1 \), transition \( t_2 \) occurs and place \( p_2 \) has a token whose color indicates the time of complex event \( E \).

**IV. DETECTING COMPLEX EVENTS WITH ED-NET MODELS**

In this section, we will introduce how to detect complex events based on ED-net. It is assumed that the RFID data have been filtered before the detection process and the inputs of the event detector is clean. The filtering step could eliminate redundant data [17] [18]. The detection process takes filtered RFID data as input, and sends signals to applications when an event is detected. As introduced in previous section, each event corresponds to a place in an ED-net, which is marked with a token when the event occurs. The ED-net model behaves dynamically by taking steps, and once an event place is marked after one step, associated actions defined in an application will be taken, such as sending an alarm or updating database.

![ED-net model for E and E'](image)

The detection process could be divided into two parts: constructing an ED-net model for all the events to be detected and detecting complex events based on the ED-net model which takes RFID data as inputs. To improve the efficiency of the detection method, we combine ED-net models for different complex events and a step-by-step detection mode is applied. Details of the two parts are given below.

**A. Constructing ED-net models**

In the work of [11] detection is carried on separately. The graph-based computation model merges common sub-graphs, thereby avoiding multiple detections for sub-events. However, common sub-events in different complex events are not combined together. Therefore, they will be detected multiple times in separate detection processes. Instead of modeling all complex events separately, we combine the ED-net models for all the events to be detected in one ED-net model. Note that the final ED-net model may be composed of several independent parts which share no common sub-events.

**Algorithm 1: Constructing ED-Net model**

**Input:** complex events set \( S \)  
**Output:** ED-Net model \( N \)

```
foreach complex event \( e \) in \( S \) do
    if \( e \) has not been modeled in \( N \) then
        Get \( e \)'s sub-events set \( C_e \)
        foreach sub-event \( c \) in \( C_e \) do
            if \( c \) has been modeled in \( N \) then
                Add a place \( p' \) as a copy of place \( p \)
            else
                Build \( c \)'s ED-net model;
                Extend \( N \) with \( c \)'s model;
            end
        end
        Build the ED-net model for event \( e \) (taking its sub-events as inputs);
        Extend \( N \);
    end
end
return \( S_n \)
```

Main steps of constructing an ED-net model for a set of complex events are given in Algorithm 1. A complex event is often composed of several sub-events, which may also contain sub-events. The constructing of the complex event takes the places of its sub-events as inputs. Suppose we have complex events \( E=(E_1 \land E_2) \) and \( E'=(E_1 \lor E_2) \), assuming \( E_1, E_2 \) and \( E_3 \) are all primitive events. Fig. 4 shows the sketch of ED-net model for \( E \) and \( E' \). The caption under place is the name of its corresponding event. The shadowed place \( p_j \) corresponds to event \( E \), and it is also a part of ED-net model for \( E' \).

Different events often share common sub-events. Therefore, an event may be used in more than one detection model. In this case, we duplicate the place for the common sub-event. Event \( E \) participates in both complex events \( E_1 \) and \( E_2 \). Once an event \( E \) is detected in place \( p_1 \), the information will be duplicated to places \( p_2 \) and \( p_3 \), which will be used in the modeling of \( E_1 \) and \( E_2 \) respectively. In this way, event \( E \) only needs to be detected once, instead of twice in the separate detection for \( E_1 \) and \( E_2 \).
B. Detecting complex events with ED-net model

Detection for complex events behaves as Algorithm 2 lists. The detector works as long as RFID data are received. The marking of the ED-net model is a global variable, and whenever a primitive event is detected, the marking is updated. To state clearly, we use notations $\dagger$ and $\vdash$ to represent sets of input and output places of transition $t$ respectively.

Algorithm 2: Detect complex events based on ED-net


1. Current marking of $N$; Enabled transition set $Te = \emptyset$;
2. Repeat
   3. foreach $t$ in $T$ are marked do
      4. Find $b$ for variables related to $t$
      5. until $G(t)$ is true under binding $b$
      6. Add transition $t$ to $Te$;
   7. foreach $t$ in $Te$ do
      8. Fire $t$;
      9. Evaluate output arc expressions of $t$
      10. foreach place $p$ in $t \dagger$ do
        11. $M'(p) = M(p) - E(p_i)$;
        12. endforeach
      13. foreach place $p$ in $t \vdash$ do
        14. $M'(p) = M(p) + E(t_j)$;
        15. if $p$ is a complex event
        16. Send $M$ to applications;
        17. endforeach
      18. foreach place $p$ in $P - (t \dagger \cup t \vdash)$ do
        19. $M'(p) = M(p)$;
        20. endforeach
      21. Change current marking $M$ to $M'$;
      22. Remove transition $t$ from $Te$;
   23. until no RFID data are received;

V. CONCLUSION

We have proposed a method for the detection of complex events in RFID, based on a formal model named ED-net. ED-net offers unified ways for describing both temporal and parameterized constraints of events and defining rules for calculating attributes of complex events. Color sets and functions are used to specify attributes and constraints of events. Constraints of a complex event are expressed by the structure of corresponding ED-net model and guard expressions associated with transitions. Attributes of a complex event can be obtained from those of its sub-events, and the calculation rule is specified in the bodies of transitions. Our detection method describes all complex events to be detected in one model; common sub-events are detected only once, instead of multiple times in independent detecting processes for these complex events, thereby improving efficiency of our methods.

We implement the prototype of ED-net with Java language in RFID test environment and compare it with traditional non-ED-net algorithm such as RCEDA[11]. We produce the RFID complex events with 3 sub-events combined about 1000 per seconds, and the average latency of detecting complex events has been efficiently reduced in our experiments. In our experiment, the average CPU occupation time maintained an acceptable level with the average latency down in detecting.

Our future work is to promote our detection method in an RFID event detector and apply it in specific applications.

REFERENCES