2018 International Conference on Mechanical, Electronic and Information Technology (ICMEIT 2018)
ISBN: 978-1-60595-548-3

Research on Complex Event Detection over Streams Supporting Interval Temporal Logic and Regular Expression Pattern Matching

Jia-wei ZHANG*, Hu-sheng LIAO, Hong-yu GAO and Chang SU

Faculty of Information Technology, Beijing University of Technology,
No.100 Pingle Yuan, Chaoyang, Beijing, China

*Corresponding author

Keywords: Complex event processing, Interval temporal logic, Regular expression.

Abstract. To improve the detective ability of complex event processing (CEP), we combine interval temporal logic (ITL) and regular expression in pattern description of event streams constraint. We propose different timed automata to support varied ITL. The pattern described by regular expression improves the simplicity of the pattern description. Meanwhile, we have implemented above technique in a CEP system to test performance. The experimental evaluation shows that the pattern can be efficiently and accurately matched.

Introduction

The technology of complex event processing treats the data produced in the system or the data come out from an external system as an event. When events are captured, the generated event stream will be matched and filtered by the pattern, leaving out the meaningful events. When an indicator is reached, a feedback will be made at once. The technology is widely used in the field of typical streaming data such as Internet of Things information monitoring [1,2,3], medical institution [4], financial information monitoring [5]. In addition, the technology has also made some achievement. However, most complex event processing system focus on discovering the point-Based events pattern, but they have insufficient support for interval-based events pattern detect.

The start time and the finish time of an event may not appear at a same time point, which will lead to the time duration between two points. A temporal constraint on two events who have time durations can be more useful in realistic requirements. For example, in the event of an accidental fire case, the temperature of an event monitored by a fire monitor is an interval-based event. This event may be accompanied by temperature rise or drop, and other events such as auto-sprinkler working can occur. The traditional temporal constraint of point-based events cannot accurately describe the complex pattern. Therefore, it is necessary to discuss the temporal relationship between interval-based events.

Allen [6] proposed that there are 13 kinds of temporal logic relations exist between two interval-based events, those relations are called interval temporal logic (ITL). The introduction of the ITL into complex event processing techniques can enrich the pattern matching of events so as to obtain a more accurate event pattern. Based on this, Patel et al. [7] introduced ITL in event pattern mining to find the temporal relationship between events in event stream. Li et al. [8] introduced interval temporal logic into complex event processing techniques in 2011, which can detect events that conform to the exact temporal relationship patterns. Liu et al. [9] also added it into complex event processing techniques. However, these two technologies have determined the start and finish time of the event when the event reached. That limits its detection capability, as some relationship of events can be determined when the event is not over.

Wang et al. [10] described the temporal relationship between point-based events by regular expression. Because, regular expression not only can describe the relationship between any two different events which come from different sources, you can also use the closure to describe the occurrence of multiple occurrences of the same event source easily.

If combine regular expression and ITL to constrain event patterns, we can not only make up the lack of description of temporal logic relations between events, but also can describe the pattern conveniently, which can enhance the complex events detection ability.

Contributions:

a. For the interval-based events, we proposed a pattern matching technique which was based on regular expression, this technique enhanced the ability to detect complex events.

b. According to the matched regular expression under different ITL, a pattern matching method based on automata is proposed.

c. We have implement above technique in a complex event processing system. The experimental evaluation shows that this method can efficiently and accurately perform ITL pattern matching base on regular expression.

Preliminaries Definition

Interval Temporal Logic

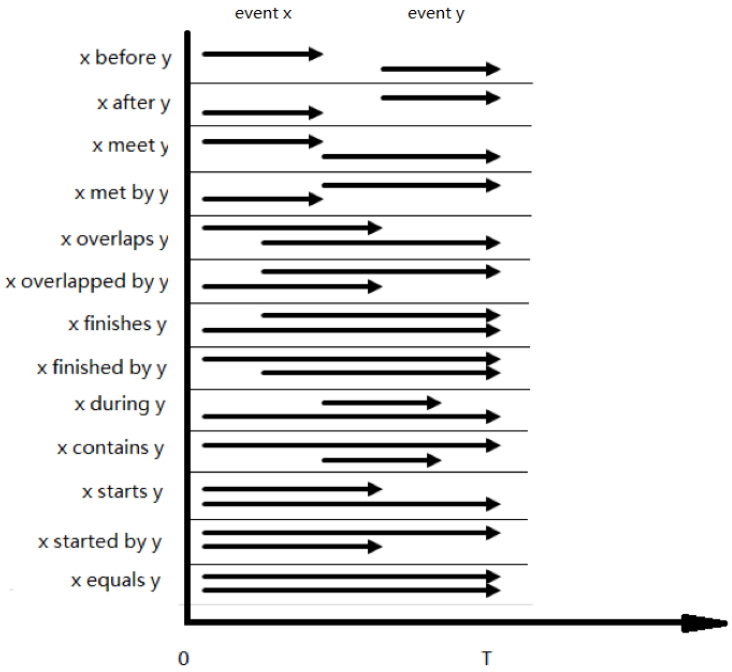


Figure 1. ITL between two during-base events.

For any two interval-based events x and y, there may be a variety of temporal relationships. According to Allen's proposed temporal relationship model, this paper considers a total of 13 species. As shown in Fig. 1, according to the start time and finish time of each event, there are 6 species ITL which are symmetric. Therefore, it will be adequate to only take 7 species ITL between two events into consideration. Those are before, meet, during, finished by, contain, start and equal.

Multi-source Events of the Regular Expression

In complex event processing, regular expressions are often used to describe the temporal relationships which must be satisfied by several consecutive events. Following are the regular expression definitions adopted in this paper:

<rexp> → <rp> by <type>

<rp> → <rp> [',' <rp>] | <rp> ['|' <rp>] | <rp> '' | '(' <rp> ')' | ID*

<type> → before | meet | overlap | finished by | contains | start | equal

Among them, ID represents an event source, rexp is the regular expression, type is a type of ITL between two events. For each pattern, as a result of the match, an interval-based event is generated. The specific definition is as follows:

a. For the connection operation '*x , y*', the result event represents event x and event y conform the ITL, and the result event's start and finish time are the same as y.

b. For the disjunction operation '*x | y*', the resulting event has the same start and finish time as the first-arrived event.

c. For the Kleene operation '*x**', when each event x arrives, there will be several events formed by the '*x*' event connection with the same start and finish time as the last '*x*'.

For example, a fire accident detection event can be defined as '*temperatureRise, auto-sprayStart, temperatureDec* by overlap*', which means that "If a fire accident occurred, the temperature will rise and the auto-sprinkler will work when the temperature rising over a dangerous threshold value. And then, the temperature will be decreased or not."

Timed Automata

In this paper, timed automata (*TA*) is used to detect events. *TA* = <*K, s, Z, X, E, F*>; where *K* is the state set, $s \in K$ is the initial state; *Z* is the input symbol set; X is the clock set; F is the reception state set; E is a state transfer function of the form *(s, e, x, δ, s')*; it indicates that the current state is at s, and if the input event e occurred and the clock constraint δ is satisfied, the clock x is reset and the state *s* transform to state *s'*.

TA and TA-Net

For each regular expression with different ITL, this paper uses different time automata to implement the complex event detection. For the temporal relationship before and meet, the entire regular expression match is done by a time automata. For the remaining five temporal relationships, the expression is converted and the results are expanded by distributive laws. Then, the expression is converted to a series of sub-expressions linked by "|"; for each operator in each sub-expression of each temporal relationship, a time automata is constructed; these automata are combined into a TA-Net to complete the entire complex event detection.

Before-TA and Meet-TA

In the timed automata for interval-based events, each interval-based event labeled a is treated as two inputs, namely the event start character '*A*' and the event finish character '*/A*', which appear chronologically. As a result, it is not difficult to get a DFA using traditional automata construction methods. For example, the automata shown in Fig. 2 can be obtained for expression '*(A,B*)**' , where there is a clock T; state 0 is the initial state, the clock is reset within brackets [] and the input symbol and clock constraints is signed in transfer edge.

As shown in Fig. 2, the clock T is reset when '*/A*' is input; the subsequent transition condition '*T>0*' indicates the '*before*' temporal constraint and '*T==0*' indicates the '*meet*' temporal constraint. So, we can get two time automata, respectively describing the pattern (Fig. 2 (a)) at '*before*' temporal and the pattern at '*meet*' temporal (Fig. 2 (b)).

(a) Before-TA (b) Meet-TA

Figure 2. Before-TA and Meet-TA of (A,B*)*.

TA-Net

There is no meaningful multiple events connection in the same event source for the other five ITL except before and meet. Therefore, for the closure operator, it only needs to check whether there is an eligible event and does not need to consider whether it is repeated or not. Hence, the Kleene operation '*R**' in the regular expression can be converted to '*R?*' Then, all the "|" operations in the expression are expanded according to the distributive laws. A series of sub-expressions that are joined by the "|" operation can be generated. There is only the "," operator included in each sub-expression.

For example, when the expression is '*(A, B*)| (C, D, E)*', it will be converted to the expression '*A, B| A| C ,D, E*'. Correspondingly, it generates as shown in Fig. 3, a TA-Net. The circle in the figure is Entry-Node. It indicates that if a start symbol or an end symbol of this type of event arrived, it will send it to the next node; The box indicates a timed automata that outputs according to the internal logic of the automata.

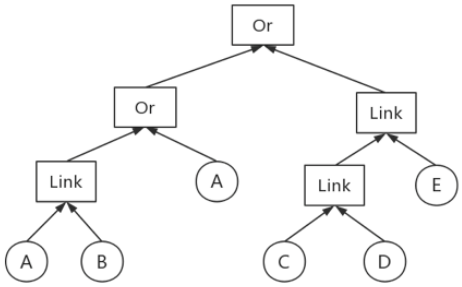
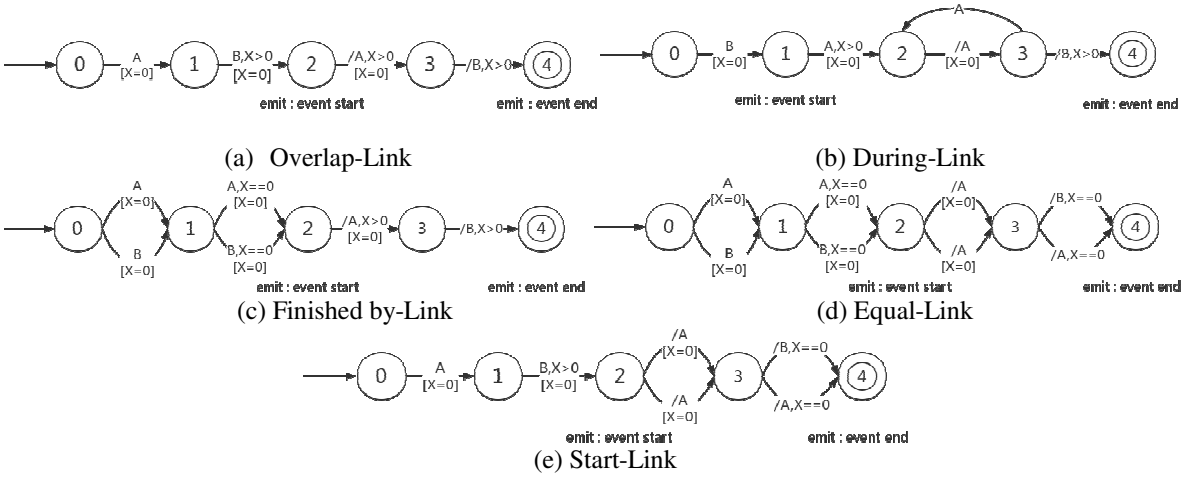


Figure 3. TA-Net of (A, B*)|(C, D, E).

The Type of TA in TA-Net

The relationship of the various ITL describes the connection semantics. Considering the semantics of 'overlap' and other five kinds of ITL, we design five kinds of timed automata with different structures respectively for them.

The five kinds of timed automata match the connection relation between a event stream and b event stream, and all state transitions occur when the start symbol '*A*' and '*B*' or the end symbol '*/A*' and '*/B*' of the a or b event occur. The structure shown in Fig. 4.



(a) Overlap-Link (b) During-Link

(c) Finished by-Link (d) Equal-Link

(e) Start-Link

Figure 4. TA in deferent ITL.

In the time automata mentioned above, a continuous result event will be generated for the input event that satisfies the matching relationship. For example, for an Overlap-Link time automata, starting at state 0, after experiencing '*A*' resets '*X*', encountering '*B*' and '*X > 0*' sends the start of the result event. When met after '*/A*' and '*/B'*, overlap relationship holds; Thus, the end of the result event sent out.

Interval-based Events Detection Processing

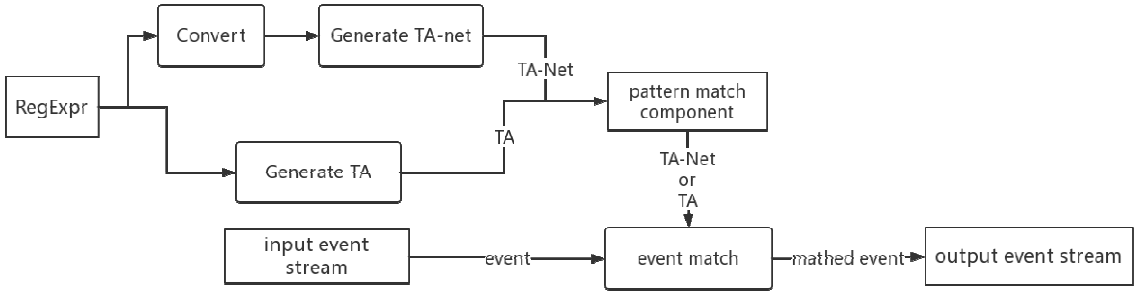


Figure 5. Pattern matching process.

Identify ITL

When entering a regular expression, the system will generate a TA-Net or a TA according to different species of ITL. If the TA was generated, the system will pass all the event streams into the automata according to the arrival time. Otherwise, the system synchronously passes the event stream to each entry-Node. After multi-layer filtering, matching stream of events is obtained.

TA-Net Pattern Matching

Because the rules of the generation of the Before-TA and the Meet-TA have little change with the rules of the original regular expression to the automata, this paper mainly introduces the matching method of the TA-Net.

This part mainly consists of 3 parts algorithms: expression conversion algorithm, TA-Net generation algorithm and matching event output algorithm.

Expression Conversion Algorithm. This algorithm intends to convert the '*' operator in the expression to '?' operator. The conversion process is done using the operator stack and character stack. After getting the corresponding expression, the "ε" character is combined with the subsequent expression according to the distributive laws.

TA-Net Generation Algorithm. Identification network may include three kinds of nodes, entry-Node, timed automata, Output-Node. In the expression, 'ID' corresponds to the entry-Node, which filters out the ID-conforming events. The timed automata is corresponded to the "," operation, and the Output-Node sends the result to the outside.

The TA-Net Constructed from bottom to top. Timed automata are generated based on the current operator type. After getting the final timed automata, we construct the output node and set it as the target node of the last automata.

Matching Event Output Algorithm. After getting the input symbol, the automata will transfer the state and make corresponding transfer action. When the state transferred cannot be occurred, if the event has been sent to the next node start character, then transfer to the trap state. After reaching the trap state, if any event termination symbol is received, an event cancellation message is sent to the next node, otherwise, all automata in the recognition network are reset. If no events are sent to the next node at this time, all automatic machines in the identification network are reset. If the event is canceled, the "canceled message" will be output.

Evaluation

In this paper, we use two different approaches to deal with regular expression under different ITL. To verify the correctness and performance of the solution, the algorithm was implemented and several sets of experiments were designed for testing.

Experimental Setup

A four-node cluster environment with one node for event detection agents, one node for CEP server, and two nodes for stream data processing. One of the nodes is 64G memory, and the remaining three nodes are 32G memory.

Experiment Design

The experiment tested the following three groups of queries:

Table 1. Event detection expression case.

Case Name	Regular Expression	Type
Q1.1	A	single-source
Q2.1	A,B	multi-source
Q3.1	(A|B)*,C*	multi-source
Q3.2	(A,B)*,C*	multi-source
Q3.3	A,B,C*	multi-source
Q3.4	A*,B*,C,D	multi-source
Q3.5	A,B,C,D	multi-source
Q3.6	A|B|C|D	multi-source
Q3.7	(A|B),C,D	multi-source
Q3.8	(A|B)*,(C|D)	multi-source

Q1.1 is a single-source event detection case. Q2.1 is a single TA test case.Q3.1-Q3.8 for multiple event source matching case.

Performance Evaluation

The case Q3.2, Q3.5 and Q3.8 in different ITL's throughput capacity test results and the response time results are shown in Fig. 6 and Fig. 7, respectively.

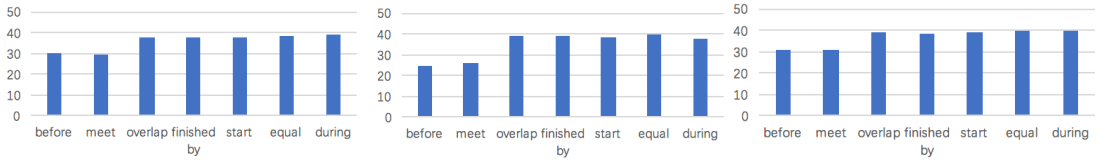


Figure 6. Throughput capacity(k/s)

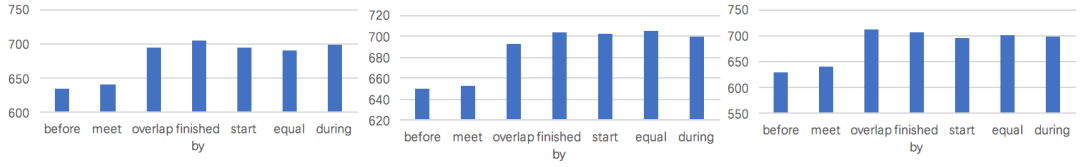


Figure 7. Response time(ms)

Conclusion and Outlook

In this paper, we expand the complex event processing technology by combining regular expression and ITL. The technique we proposed can support pattern matching for interval-based events. In order to ensure correctness of the matching result and efficiency of the processing, different processing methods are adopted for the pattern with different ITL. Our evaluation shows that the matching is accurate and it can handle huge amounts of data. In future work, we will focus on TA-Net research to improve performance by increasing data transfer rates.

References

[1] Jing Xin, Zhang Jing, Li Junhuai. Overview of complex event processing technology and its application in logistics Internet of Things[J]. Computer Applications, 2013, 33(7):2026-2030.

[2] Bruns R, Dunkel J, Masbruch H, et al. Intelligent M2M: Complex event processing for machine-to-machine communication[J]. Expert Systems with Applications, 2015, 42(3):1235-1246.

[3] Strack C, Seidemann M, Seeger B, et al. Complex event processing for reactive security monitoring in virtualized computer systems[C]//ACM International Conference on Distributed Event-Based Systems. ACM, 2015:22-33

[4] Kim M J, Yu Y S, Min B H. Development of Bigdata Application System Using Complex Event Processing Technology for Medical Institution[J]. 2016, 14(2):99.

[5] Adi A, Botzer D, Nechushtai G, et al. Complex Event Processing for Financial Services[C]// Services Computing Workshops, 2006. SCW '06. IEEE. IEEE, 2006:7-12.

[6] Allen J F. Maintaining Knowledge about Temporal[J]. Readings in Qualitative Reasoning About Physical Systems, 1990, 26(11):361-372.

[7] Patel D, Hsu W, Lee M L. Mining relationships among interval-based events for classification[C]//ACM SIGMOD International Conference on Management of Data. ACM, 2008:393-404.

[8] Li M, Mani M, Lin T, et al. Constraint-aware complex event pattern detection over streams[C]// International Conference on Database Systems for Advanced Applications. Springer-Verlag, 2010:199-215.

[9] Liu Y, Zhang H, Wang Y. RFID Complex Event Processing: Applications in Real-Time Locating System[J]. International Journal of Intelligence Science, 2012, 2(4):160-165.

[10]Wang Yi-xiong, Liao Hu-sheng, Kong Xiang-xuan. CEStream: A Complex Event Stream Processing Language[J]. Computer Science, 2017, 44(4):140-143.