

Construction and specification of complex events from sensor network data in farmland

Li Xiang, Wei Xiaohong, Chen Xin^{*}, Tang Xianglu, Xie Taomin, Jia Lu

(College of Information and Electrical Engineering, China Agricultural University, Beijing 100083, China)

Abstract: An important task of expert systems for precision farming is to recognizing composite situations in farmland to control intelligent devices to precision farming. In the existing expert systems, knowledges of composite situations are constructed to threshold inequation or production rules. However, threshold inequations could model only simple situations and production rules are complicated and lack readability. Complex event processing (CEP) is a helpful online pattern recognition technology. With CEP, composite situations can be detected according to pre-specified complex event patterns. In this paper, a novel model for complex events from sensor network in farmland is proposed, and a relevant event specification language is defined based on XML syntax. The model considers various influencing factors to judge and construct farmland complex events in a comprehensive manner. Especially, it supports mathematical, temporal and spatial logical relationships among constituent events of complex events. Spatial logic is necessary to model complex events from distributed sensors, but it is not supported by traditional complex event models. Data for 1 year from 27 sensors deployed in a 10 meters by 10 meters square farmland are collected. By analyzing the sensor data, five kinds of typical composite situations, i.e. rain, dry, irrigation, seepage and sensor damage, are constructed with the complex event model and specified with the specification language. Tests indicate false positives and omissions rate of our constructed complex events are significantly lower than threshold control, the readability is better than production rules and complexity of specification is significantly lower than the widely-used model in SASE+.

Keywords: complex event processing, precision farming, expert system, sensor network, internet of things

Citation: Li, X., X. H. Wei, X. Chen, X. L. Tang, T. M. Xie, and L. Jia. 2017. Construction and specification of complex events from sensor network data in farmland. *International Agricultural Engineering Journal*, 26(3): 269–282.

1 Introduction

With wide use of intelligent agricultural expert systems in precision farming, the concept of “intelligent” is discussed more frequently. A necessary ability of an intelligent agricultural expert system is that it can recognize and respond to more composite situations in time. Nowadays, with the development of Internet of Things technology, more sensors are deployed in farmland to sense mass of status of air, soil and plants. Composite situations should be more precisely recognized from several dimensionalities of data from the

mass of sensors deployed in different places, than from a single sensor. For example, rain is a composite situation. When it is rainy, a rain sensor in weather station can sense the rain and soil humidity sensor can sense the change of abnormal high humidity of soil. However, abnormal high humidity from a single sensor is not enough, for it may imply the sensor damage, instead of rain.

For this demand, traditional threshold control methods are not suitable, because only simple situations can be modeled. Many expert systems based on production inference were developed (Liu, 2007; Janssen et al., 2010). Although composite situations can be modeled in production rules, the specifications of these rules are complicated and lack of readabilities.

Complex event processing (CEP) is a helpful pattern recognition technology. Each composite situation can be

Received date: 2017-07-06 Accepted date: 2017-08-21

^{*} Corresponding author: Xin Chen, Associate Professor of College of Information and Electrical Engineering, China Agricultural University, 100083 P. R. China. Email: chxin@cau.edu.cn. Tel: +86-135-0100-5205.

modeled as a complex event, which describes necessary data from sensors and temporal and spatial relationships among these data for recognition of the composite situation. CEP algorithms are used to online recognize and detect composite situations from atomic data depending on the pre-specified complex event patterns in real-time. Precise construction and specification of complex events with an effective complex event model is the key to precise recognition of composite situations.

In the recent decade, complex event models, real-time event detection algorithms and applications in different areas are developed (Buchmann et al., 2009). In 1990s, CEP was used in database management systems, in which semantic graph (Chakravarthy et al., 1994), Petri net (Gatzia, 1992) and finite state automata (Gehani, 1992) were the three most widely used models to construct database complex events. The models have supported mathematical logic and simple temporal logic, but their structures were too abstract and their readabilities are poor for users. In the late 2000s, it was found that CEP is effective to process event flow, so the technology got paid attention again. SQL-like models were developed (Wu, 2006). Then a regular SQL-like model was defined in SASE+ system, special operators like Kleene closure are defined (Diao et al., 2007). The model of SASE+ is still one of the most popular model now. A sliding window was added to the model (Krämer, 2009). Another popular model is defined in Esper system (EsperTech, 2016). It is also a SQL-like model. In addition, a XML-based complex event model is proposed in (Zang et al., 2007). Some new complex logic operators to specify composite situations are defined in IBM Amit system (Magid et al., 2010). The models above can describe very composite situation and have suitable readabilities. However, these models have not considered spatial logic relationships among constituent events, so they are lengthy when specifying complex events from sensor network distributed deployed in farmland.

In recent years, CEP is viewed as one of the leading technologies of Internet of Things (Haller et al., 2008). Event models and process algorithms are widely studied (Fengjuan et al., 2013; Jun et al., 2014). However, the advanced technology is rarely used in agriculture. Precisely model and specification language of complex

events of internet of things in farmland are insufficient.

In this paper, a novel complex event model for farmland complex events is proposed and meanwhile a relevant event specification language is defined based on XML syntax. Five kinds of typical composite situations, i.e. rain, dry, irrigation, seepage and sensor damage, are constructed to complex events by analyzing data from actual sensor network in farmland and specified with the specification language. Tests indicate false positives and omissions rate of our constructed complex events are significantly lower than threshold control, the readability is better than production rules and complexity of specification is significantly lower than the widely-used model in SASE+.

2 Complex event processing

An event is variations of some states in the physical world. It is usually “complex”, i.e. several or dozens of states and variations of states need to be considered, so it is called complex event. Conversely, a variation of a single state is called an atomic event. A complex event could be specified in a pattern, which describes how the above-mentioned states and variations (i.e. atomic events) determine whether the complex event occurs and information of the occurrence. When a system is monitoring huge number of states in physical world, a number of atomic events can be detected and continually converged into an atomic event flow. From the flow, a complex event processing algorithm is used to detect complex events according to pre-specified complex event patterns.

There are two hot research topics in complex event processing area: 1) how to model complex events and 2) how to efficiently detect complex events from atomic event flow. This paper focuses on the first topic. A model of a complex event includes constituent atomic events and their relationships which can be described with mathematical logic, temporal logic and spatial logic functions. For example, $((E_1 \Delta E_2); E_3); (E_2 \Delta E_4)$ is a complex event, in which E_1, E_2, E_3, E_4 are atomic events and linked with two logical operators: “ Δ ” and “;”, respectively denoting mathematical logic function “and” and temporal logic function “sequence”. With “;”,

{{}}

2.1 Complex event model

A complex event model with spatial logic is proposed and a specification language to describe complex event models is defined. In the model, an event is formal represented as $E=(i, a)$, $a = \{d, t, r, p, o\}$. a is an attribute set, In that, $d=[t_0, t_1]$ and t denote time period and time point of the event occurrence; r and p denote spatial region and spatial position of it; o denotes other attributes of the event.

Figure 1 illustrates hierarchical structure of complex event model. A complex event consists of constituent events. Whether the complex event occurs or not is determined by constituent events. Attributes of the complex event is calculated with attributes of next level of events. Four sets determine occurrence and attributes of a complex event: ES , IS , FJ , and FC .

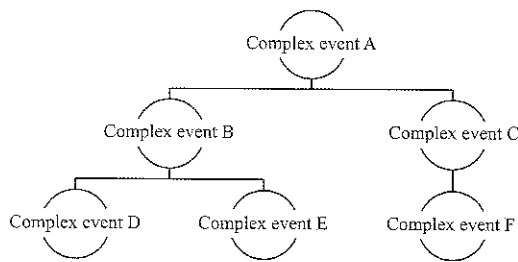


Figure 1 Hierarchical structure of complex event model

ES is constituent events set. For example, in Figure 2, $ES_A = \{B, C\}$.

IS is occurrence consumption strategies set, that determines which occurrences of constituent will be used or removed. Different consumption strategies can be chosen for different constituent events.

$$IS = (IS_C, IS_R, IS_W) \tag{1}$$

IS_C – chosen strategy of occurrences of constituent events used to assemble the complex event. Four common chosen strategies are: *recent* – choosing newest occurrences of constituent events; *chronicle* – choosing earliest occurrences of constituent events in the time window; *continuous* – choosing all occurrences of constituent events and assemble several occurrences of the complex event with different combinations; and *cumulative* – choosing all occurrences of constituent events and assemble only one occurrence.

IS_R – determines which occurrences will be removed

from the constituent event occurrences flow after the detection of the complex event. Usually only newest occurrences are usable to detect complex event, so unusable occurrences should be removed because of store space limitation. Four common remove strategies are: *reserve* – not removing any occurrences; *removeone* – removing all occurrences in IS_C ; *removeearly* – removing all occurrences in IS_C and removing all occurrences occurred early than them; and *removeall* – removing all occurrences in the time window.

IS_W – determines the length of the sliding window of the constituent event occurrences flow. New occurrences will be added to the window continually and occurrences out of the window will be removed. Two common window strategies are: *timewindow* – occurrences occurred in a period keeping in the windows; and *lengthwindow* – a certain number of occurrences keeping in the window. Here which occurrences can be remained in the window is determined by both IS_R strategy and IS_W strategy.

FJ is a function to judge whether the complex event occurs.

$$FJ(\{e.a \mid \forall e \in IS_U(ins(E_{i1}), ins(E_{i2}), \dots, ins(E_{in})) \}) = \begin{cases} \text{true, complex event occurs} \\ \text{false, complex event not occurs} \end{cases}, E_y \in ES \tag{2}$$

$$\begin{cases} FJ = FJ_{occr} \ \& \ FJ_{d,t} \ \& \ FJ_{r,p} \ \& \ FJ_o \\ FJ_{occr} \equiv CJ(e_{i1}, \dots, CJ_1(e_{j1}, \dots), \dots) \\ FJ_{d,t} \equiv TJ(e_{i1}.t, e_{i1}.d, \dots, TJ_1(e_{j1}.t, e_{j2}.d, \dots), \dots) \\ FJ_{r,p} \equiv SJ(e_{i1}.p, e_{i1}.r, \dots, SJ_1(e_{j1}.p, e_{j2}.r, \dots), \dots) \\ FJ_o \equiv OJ(e_{i1}.o, \dots, OJ_1(e_{j1}.o, \dots), \dots) \end{cases} \tag{3}$$

FJ_{occr} defines rules to judge whether necessary constituent events occur. $FJ_{d,t}$ judges whether temporal relationships among constituent events in IS_C meet requirements by calculating d and t of them. $FJ_{r,p}$ is the judgement function for spatial relationships with arguments r and p , and FJ_o is for other relationships with arguments o .







e_E – an occurrence of event E .

$ins(E)$ – occurrences flow of an event E .

$e_{E,x}$ – attribute x of e_E .

CJ , TJ , SJ and OJ – occurrence, temporal, spatial and other judgement operators. All defined operators are listed in Table 1.

Table 1 Defined occurrence (CJ), temporal (TJ), spatial (SJ) and other judgement operators (OJ)

Category	Operator	Implication	
CJ	$con(e_A, e_B, \dots)$ or $e_A \wedge e_B \wedge \dots$	All of events A, B, \dots occur	
	$dis(e_A, e_B, \dots)$ or $e_A \vee e_B \vee \dots$	One of events A, B, \dots occurs	
	$any(e_A, e_B, \dots, n)$	Any n events of A, B, \dots , occur	
	$not(e_A, e_B, \dots)$	None of events A, B, \dots occur	
	$count(e_A, n)$	Event A occurs for n times	
TJ	$sync(e_A, t, e_B, t, \dots, p)$	Events A, B, \dots occur in the same period p	
	$between(e_A, t, e_B, t, e_C, t)$	Event B occurs between A and C occur.	
	$sequence(e_A, t, e_B, t, \dots)$	Event A, B, \dots occur in the order	
	$after(e_A, t, e_B, t)$	An event B occurs after an event A	
		d of an event A is in d of and event B , i.e. $in_{align}(e_A, e_B) \equiv e_A.t_0 \geq e_B.t_0 \ \& \ e_A.t_1 \leq e_B.t_1$ Values of subscript <i>align</i> : <i>begin</i> - $e_A.t_0 = e_B.t_0$ <i>end</i> - $e_A.t_1 = e_B.t_1$ <i>both</i> - $e_A.t_0 = e_B.t_0 \ \& \ e_A.t_1 = e_B.t_1$ <i>none</i> - $e_A.t_0 > e_B.t_0 \ \& \ e_A.t_1 < e_B.t_1$	
		An event B occurs following an event A , i.e. $follow_{align}(e_A, e_B) \equiv e_A.t_0 < e_B.t_0 \ \& \ e_A.t_1 < e_B.t_1$ Values of subscript <i>align</i> : <i>separate</i> - $e_A.t_1 < e_B.t_0$ <i>overlap</i> - $e_A.t_1 > e_B.t_0$ <i>tangency</i> - $e_A.t_1 = e_B.t_0$	
	$follow(e_A, d, e_B, d)$		
	$same(e_A, p, e_B, p)$	Event A occurs in the same point with event B	
	$split(e_A, p, e_B, p)$	Event A occurs in the different point with event B	
	$inside(e_A, p, e_B, r)$	Event A occurs in the region of event B	
$outside(e_A, p, e_B, r)$	Event A occurs out of the region of event B		
SJ	$disjoint(e_A, r, e_B, r)$	Region of e_A is disjoint with region of e_B 	
	$circumscribe(e_A, r, e_B, r)$	Region of e_A circumscribes region of e_B 	
	$overlap(e_A, r, e_B, r)$	Region of e_A overlaps with region of e_B 	
	$inscribe(e_A, r, e_B, r)$	Region of e_B inscribes with region of e_A 	
	$contain(e_A, r, e_B, r)$	Region of e_A contains region of e_B 	
	$equal(e_A, r, e_B, r)$	Region of e_A is the same as region of e_B 	
	OJ	Other relationships to judge occurrence of the complex event with attributes of constituent events	

FC is a function to calculate attribute set a with attribute sets of constituent events.

$$e_E.a = FC(\{e.a \mid \forall e \in IS_U(ins(E_{i1}), ins(E_{i2}), \dots, ins(E_{in})))\}, E_{ij} \in ES) \quad (4)$$

$$\begin{cases} FC \equiv (FC_{d,t}, FC_{r,p}, FC_o) \\ FC_{d,t} \equiv TC(e_{i1}.t, e_{i1}.d, \dots, TC_1(e_{j1}.t, e_{j2}.d, \dots), \dots) \\ FC_{r,p} \equiv SC(e_{i1}.p, e_{i1}.r, \dots, SC_1(e_{j1}.p, e_{j2}.r, \dots), \dots) \\ FC_o \equiv OC(e_{i1}.o, \dots, OC_1(e_{j1}.o, \dots), \dots) \end{cases} \quad (5)$$

$FC_{d,t}$ is the function to calculate the time period (d) and time point (t) of an occurrence of the complex event. $FC_{r,p}$ is used to calculate the spatial region (r) and spatial point (p). FC_o is used to calculate other attributes (o). TC , SC and OC are respectively temporal, spatial and other calculation operators. All defined operators are listed in Table 2.

Table 2 Defined temporal (TC), spatial (SC) and other calculation operators (OC)

Category	Operator	Implication
TC	$nc(e_A, t, e_B, t, \dots)$	Calculate the time point of complex event E by numerical calculation of time points of constituent events A, B, \dots
	$continue(e_A, t, e_B, t, \dots)$	Time period (d) of complex event E is the minimum period that contains all time points (t) of constituent events A, B, \dots , i.e. $e_E.d = [\min_i e_i.t, \max_i e_i.t]$
	$intervalunion(e_A, d, e_B, d, \dots)$	The union of all time periods of constituent events A, B, \dots , i.e. $e_E.d = e_A.d \cup e_B.d \cup \dots$
	$interalintersection(e_A, d, e_B, d, \dots)$	The intersection of all time periods of constituent events A, B, \dots , i.e. $e_E.d = e_A.d \cap e_B.d \cap \dots$
	$interaldifference(e_A, d, e_B, d)$	The relative complement of d of e_B with respect to d of e_A . $e_E.d = e_A.d \setminus e_B.d$
	$interalcontinue(e_A, d, e_B, d)$	Time period (d) of complex event E is the minimum period that contains all time periods (d) of constituent events A, B, \dots , i.e. $e_E.d = [\min_i e_i.t_0, \max_i e_i.t_1]$
	$centroid(e_A, r)$	The centroid of the region (r).
	$polygon(e_A, p, e_B, p, \dots)$ or $polygon(e_A, r, e_B, r, \dots)$	Spatial area (r) of complex event E is the minimum polygon that covers all spatial points (p) or all regions of constituent events A, B, \dots
	$circle(e_A, p, e_B, p, \dots)$ or $circle(e_A, r, e_B, r, \dots)$	Spatial area (r) of complex event E is the minimum circle (in 2D space) or ball (in 3D space) that covers all spatial points (p) or all regions of constituent events A, B, \dots
	SC	$areaunion(e_A, r, e_B, r, \dots)$
$areaintersection(e_A, r, e_B, r, \dots)$		The intersection of all spatial of constituent events A, B, \dots , i.e. $e_E.r = e_A.r \cap e_B.r \cap \dots$
$areadifference(e_A, r, e_B, r)$		The relative complement of r of e_B with respect to r of e_A . $e_E.r = e_A.r \setminus e_B.r$
$areapolygon(e_A, r, e_B, r, \dots)$		Spatial area (r) of complex event E is the minimum polygon that contains all spatial areas (r) of constituent events A, B, \dots
$areacircle(e_A, r, e_B, r, \dots)$		Spatial area (r) of complex event E is the minimum circle (in 2D space) or ball (in 3D space) that contains all spatial areas (r) of constituent events A, B, \dots
OC	Other relationships to calculate a of the complex event with attributes of constituent events.	

2.1 Complex event specification language

A complex event specification language based on XML syntax is designed to describe complex events.

For atomic event, identification, sources and attributes

will be specified, and for complex event, identification, *ES*, *IS*, *FJ*, and *FC* will be specified, as Table 3.

For a constituent event, a judgement function and a calculation function, the syntax is as Table 4.

Table 3 Syntaxes for specifying atomic event and complex event

Atomic event	Complex event
<pre> <atom> // an atomic event <id>atomevent1</id>//identification <sources> <src>sensor1</src>//a source ... </source> <attributes>// attributes <attr id="o1" type="x" var="true false"> o1 </attr> //an attribute, type is d t r p o ... </attributes> <jfunc> //a judgement function ... </jfunc> </atom> </pre>	<pre> <ce> // a complex event <id>complexevent1</id>//identification <otherattrs> ... </otherattrs>//other attributes <eventset>//constituent event set ES <event>...</event>//a constituent event and its occurrence consumption strategies ... </eventset> <judgefunction logic="conjunction disjunction"> //judgement functions ... </judgefunction> <calculationfunction>//calculation functions ... </calculationfunction > </ce> </pre>

Table 4 Syntaxes for specifying a constituent event, a judgement function and a calculation function

Constituent event	
<pre> <event> <evn>...</evn>//id of the constituent event <chosenstrategy>recent chronicle continuous cumulative</chosenstrategy>//IS_C, with 4 options <removestrategy>reserve removeone removeearly removeall </removestrategy>//IS_R, with 4 options <timewindow>time1</timewindow>//time window <lengthwindow>10</lengthwindow>//length window </event> </pre>	
Judgement function	Calculation function
<pre> <judgefunction logic="conjunction disjunction"> <variables>//variables for judgement <var id="v1">//a variable <evn>e1</evn>//constituent event "e1" <occ>1</occ>// occurrence serial <attr>o1</ attr >// an attribute "o1" </var> ... </variables> <jfunc>f1(\$v1,\$v2,...)</jfunc> <jfunc>f2(\$v1,\$v2,...)</jfunc> ... </judgefunction> </pre>	<pre> <calculationfunction> <variables>//variables for calculation <var id="v1">//a variable <evn>e1</evn>//constituent event "e1" <occ>1</occ>// occurrence serial <attr>o1</ attr >// an attribute "o1" </var> ... </variables> <cfunc attr="d r p ..."> f(\$v1,\$v2,...) </cfunc> </calculationfunction> </pre>

3 Construction and specification of complex events in farmland

3.1 Data

Data are collected from a 10×10 m² farmland in Shangzhuang, Beijing. The farmland is empty, no plants, except for grass. Nine monitoring points are assigned in the farmland, illustrated in Figure 2a. In each monitoring point, three sensors are deployed to sense soil temperature and moisture respectively in 10 cm, 20 cm

and 40 cm deep, illustrated in Figure 2b. One weather station is deployed in north of the farmland to sense air temperature, humidity, rain, wind direction, etc. Frequencies of data sensing of soil sensors and weather station are the same - once per hour. The data was continued to collect in near one whole year, from May 2015 to May 2016. The soil moisture data sensed by sensors in the 10 cm in nine monitoring points and rainfall data sensed by the weather station from May 2015 to May 2016 are illustrated in Figure 3.

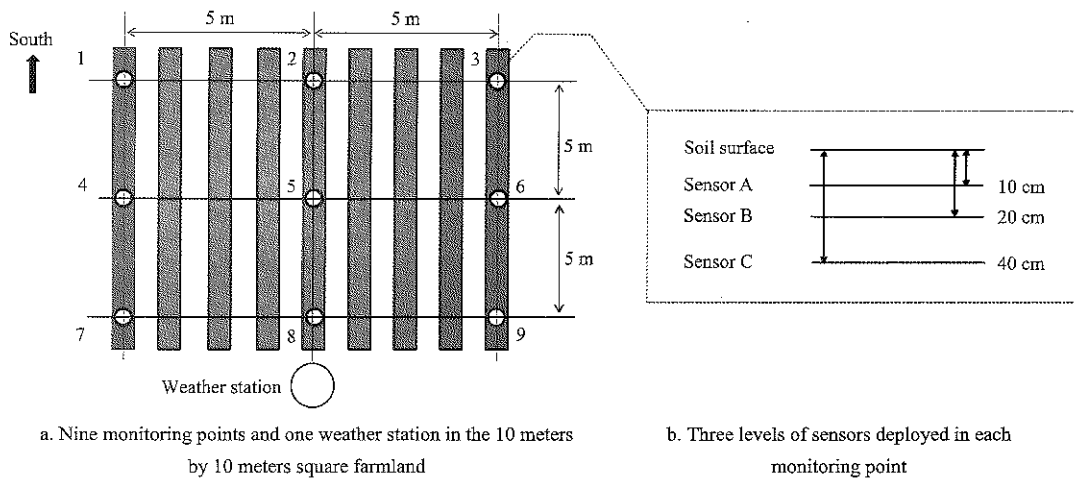


Figure 2 Setup of soil temperature and moisture data collection

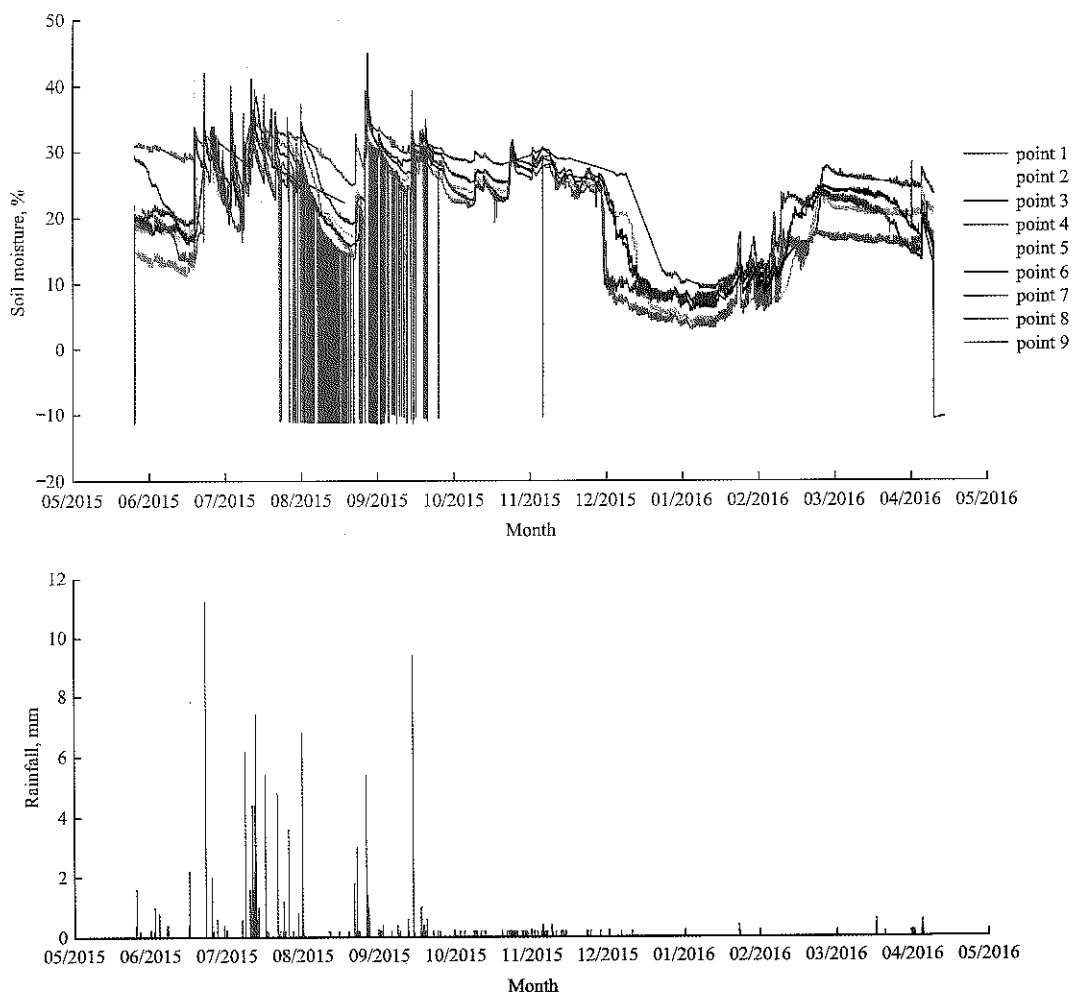


Figure 3 Soil moisture data sensed by sensors in 10 cm at 9 monitoring points and rainfall data sensed by the weather station from May 2015 to May 2016

3.2 Atomic events

For single sensor, five kinds of sensor data can be considered. The basic atomic event is “update data event”, denoted as u , which occurs when the sensor updates a piece of data.

“Higher event” and “lower event”, respectively denoted as h and l , which are defined by data from single

sensor at only one time point, imply the value of the sensor is respectively higher and lower than a threshold. For example, soil moisture is impossible higher than 100%, so “higher than 100%” is a higher event.

The other two kinds of atomic events are more or less “complex”, because they involve data from single sensor at two time points. In order to reduce the complexity of

complex events, the two kinds of frequently-used “complex” events are regarded as atomic events. They are “increase event” and “decrease event”, denoted as “*F*” and “*D*”. “increase event” implies that the increase of the sensor value from the previous time point to the latter time point is higher than a threshold. “decrease event” implies that the decrease is higher than a threshold. For example, “increase 20% between two time points” is an increase event.

Thresholds for atomic events are determined by statistics. Considering mathematical expectations (*mean*) and standard deviations (*std*) of all data from a sensor, we suggest that thresholds of higher event and lower event should be respectively $mean+3 \cdot std$ and $mean-3 \cdot std$, and thresholds of increase event and decrease event should be $1 \cdot std$. For example, for 10cm sensor at point 1, *mean* and *std* of all soil moisture data are respectively 20.12% and 8.11%. Thus, the suggested thresholds of higher event and lower event are respectively $mean+3 \cdot std=44.45\%$ and $mean-3 \cdot std=-4.21\%$. Because $mean-3 \cdot std < 0\%$, the threshold of lower event should be 0%. The thresholds of

increase event and decrease event should be $1 \cdot std=8.11\%$.

An example of higher event is illustrated in Appendix I.

3.3 Complex events

“Sensor damage event”

The first kind of considered complex events is “sensor damage”. Only after filtering “sensor damage” events, other complex events can be recognized correctly.

Figure 4 illustrates the data of the 10 cm sensors at seven points. Four sensor damage events occur at 7th point. From the figure, obvious characteristic of a sensor damage event can be found. Firstly, each sensor damage event can be detected with a higher event (h_i), a lower event (l_i), an increase event (i_i) or a decrease event (d_i), but it is not enough. In the broken blue circle in Figure 4, increase and decrease events of 10 cm sensors at all points, which imply there is a rain or irrigation event, but not sensor damage event. Thus, it implies a sensor damage event that h_i , l_i , i_i or d_i occurs at only one sensor while those atomic events have not occurred at other sensors at the same time point.

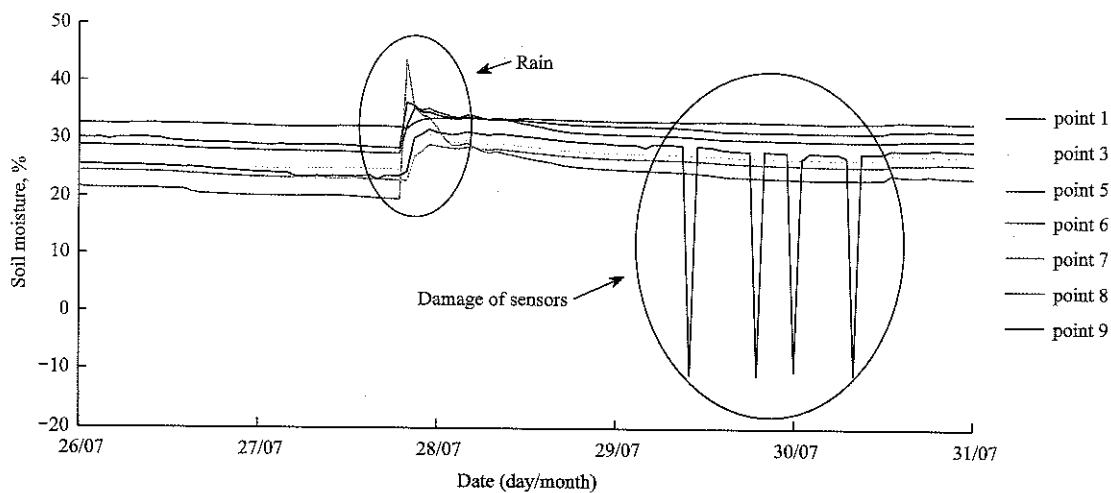


Figure 4 Four sensor damage events of the 10 cm sensor at 9th point

To construct the complex event of sensor damage, above-mentioned four kinds atomic events are constituent events. Only the newest one occurrence of each atomic event needs to be considered, so their *IS_C* strategies are *recent* and *IS_w* strategies are *lengthwindow* with one length window. After detection of the complex event, all occurrences can be removed from the flow, so *IS_R* strategies are *removeall*. *FJ* functions of the complex event are as follow:

$$\begin{cases} FJ_{occ} = dis(h_{i1}, l_{i1}, i_{i1}, d_{i1}) \wedge not(h_{i2}, h_{i3}, \dots, l_{i2}, l_{i3}, \dots, \\ \quad i_{i2}, i_{i3}, \dots, d_{i2}, d_{i3}, \dots) \\ FJ_{d,t} = sync(u_{i1}, t, u_{i2}, t, u_{i3}, t, \dots, 2) \\ FJ_{r,p} = inside(u_{i1}, p, u_{i2}, r) \wedge inside(u_{i1}, p, u_{i3}, r) \wedge \dots \end{cases} \quad (6)$$

Here $i1$ is the identification of the sensor which h_{i1} , l_{i1} , i_{i1} or d_{i1} occurs, and $i2, i3, \dots$ are identifications of other sensors. $FJ_{d,t}$ limits that the occurrence time of the update events of all considered sensors should be in two update periods of sensors. $FJ_{r,p}$ limits that position of other

considered sensors should be in the region of event i_1 .

Considering FC of the complex event, attributes $t, d, r,$ and p are equal to the corresponding attributes of h_{i1}, l_{i1}, i_{i1} or d_{i1} .

As an example, the specification of the sensor damage event with our specification language is illustrated in Appendix II.

Another kind of sensor damage event should be also considered: a sensor has not updated data for a long time. For the complex event, more than one occurrence states of update event of one sensor are detected. FJ function is:

$$FJ_{occr} = \text{count}(\text{not}(u_i), 3) \quad (7)$$

implying update event of sensor i has not occurs for three data update periods. Attributes $t, d, r,$ and p are equal to the corresponding attributes of the last occurrence of u_i . Considering consumption strategies (IS), three occurrences of u_i are used, so length of sliding window (IS_W) is 3.

“Rain event”

Figure 5 illustrates two rain events on July 23 and 28. The weather station has a rain sensor, so rain can be detected with higher events (h_{ws}) and increase events (i_{ws}) of the sensor. It is not enough, because of the errors or damage of the sensor. When it is rainy, higher events (h_i)

and increase events (i_i) of soil sensors in 10 cm occur which can be used to assist detection of the rain. Thus, FJ functions of rain event are:

$$\begin{cases} FJ_{occr} = \text{dis}(h_{ws}, i_{ws}) \wedge \text{any}(h_1, h_2, \dots, i_1, i_2, \dots, 3) \\ FJ_{d,t} = \text{sync}(h_{ws}, t, i_{ws}, t, e_{i1}, t, e_{i2}, t, e_{i3}, t, 2) \end{cases} \quad (8)$$

Here, $e_{i1}, e_{i2},$ and e_{i3} are three actual occurrences of $h_1, h_2, \dots, i_1, i_2, \dots$. It is rainy when h_{ws} or i_{ws} occurs and any three higher or increase events of soil sensors occur. All of these events should occur in two data update periods.

Considering that FC functions of a rain event (re), time attributes t and d of it are equal to the corresponding attributes of h_{ws} or i_{ws} . The region (r) of it should contain all region of constituent events and the position (p) should be the centroid of r . Thus, region and the position of the complex event is:

$$\begin{cases} re.r = \text{polygon}(h_{ws}, r, i_{ws}, r, e_{i1}, r, e_{i2}, r, e_{i3}, r) \\ re.p = \text{centroid}(re.r) \end{cases} \quad (9)$$

Considering consumption strategies (IS), only the newest one occurrence of each atomic event needs to be considered, so their IS_C strategies are *recent* and length of the windows of all constituent events are 1. After detection of the complex event, all occurrences can be removed from the flow, so IS_R strategies are *removeall*.

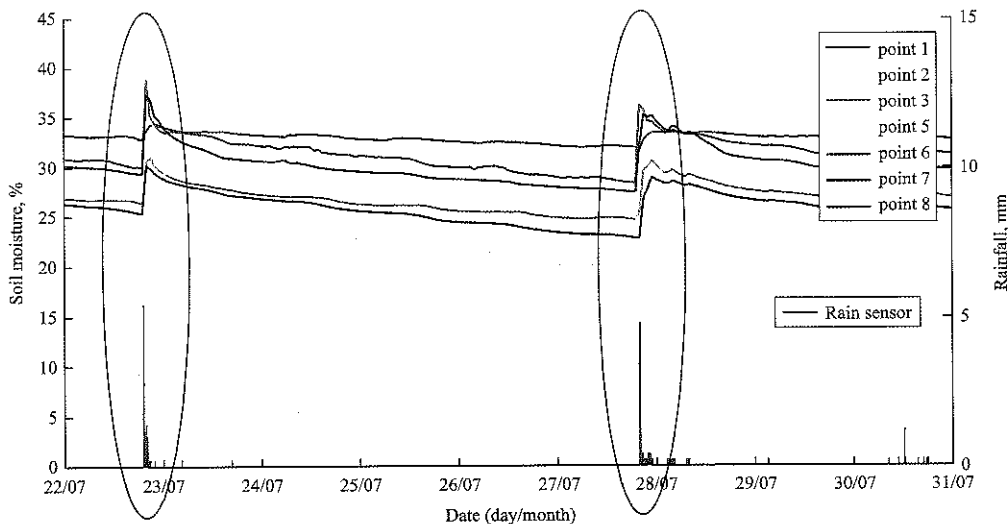


Figure 5 Two rain events on July 23 and July 28

“Irrigation event”

Figure 6 illustrates an irrigation event on November 7. When irrigation, higher events (h_i) or increase events (i_i) in 10 cm, 20 cm and even 40 cm. To ensure effect of irrigation, h_i or i_i in 20 cm are chosen. Meanwhile, to distinguish it from rain events, irrigation events usually

occur in it is not rainy. Thus, FJ functions of the irrigation event are:

$$\begin{cases} FJ_{occr} = \text{any}(h_1, h_2, \dots, i_1, i_2, \dots, 3) \\ FJ_{d,t} = \text{sync}(e_{i1}, t, e_{i2}, t, e_{i3}, t, 1) \\ \wedge \text{between}(\text{min}(e_{i1}, t, e_{i2}, t, e_{i3}, t), \\ \text{not}(re), \text{max}(e_{i1}, t, e_{i2}, t, e_{i3}, t)) \end{cases} \quad (10)$$

Here, re is a rain event, and e_{i1} , e_{i2} , and e_{i3} are 3 actual occurrences of $h_1, h_2, \dots, i_1, i_2, \dots$. An irrigation event occurs when any three higher or increase events of 20 cm

soil sensors occur, all of these events of soil sensors should occur in the same data update period, and it is not rainy in the period.

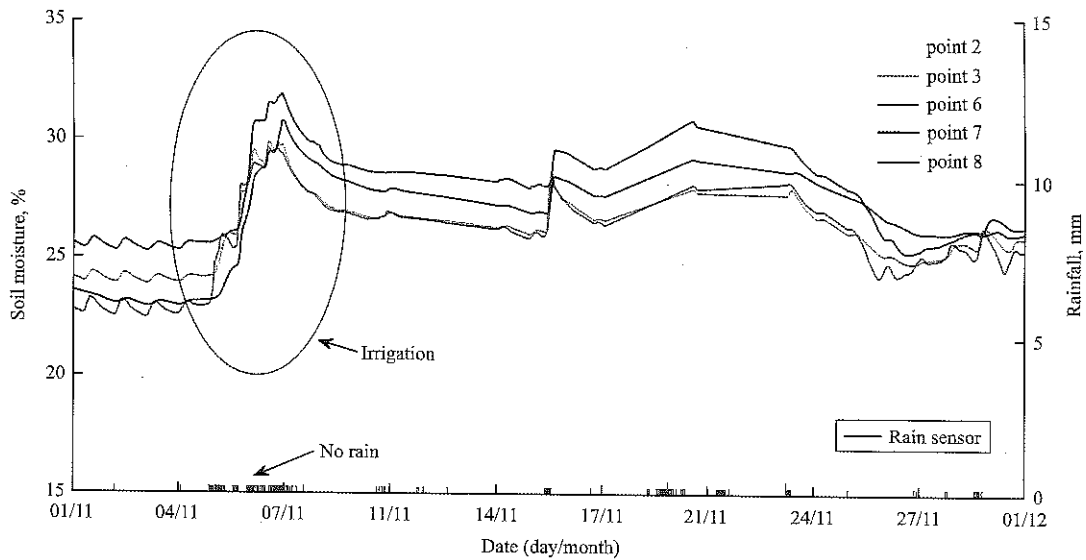


Figure 6 An irrigation event on Nov 07

Considering FC functions of an irrigation event (ie), time attributes t and d are equal to the corresponding attributes of the first constituent event h_i or i_i . Its region (r) contains all regions of constituent events and its position (p) is the centroid of its region (r), i.e.

$$\begin{cases} ie.r = polygon(e_{i1}.r, e_{i2}.r, e_{i3}.r) \\ ie.p = centroid(re.r) \end{cases} \quad (11)$$

IS_C, IS_w and IS_R strategies are the same as rain event.

“Dry event”

Figure 7 illustrates some dry events in December 2015. When soil is dry, lower events (l_i) of more than

three nearby sensors occur for three data update periods at the same time. FJ functions of dry event are:

$$\begin{cases} FJ_{occ} = any(count(l_1, 3), count(l_2, 3), \dots, 3) \\ FJ_{d,t} = sync(l_{i1}^1, t, l_{i2}^1, t, l_{i3}^1, t, 1) \\ FJ_{r,p} = overlap(l_{i1}^1, r, l_{i2}^1, r) \wedge overlap(l_{i1}^1, r, l_{i3}^1, r) \\ \quad \wedge overlap(l_{i2}^1, r, l_{i3}^1, r) \end{cases} \quad (12)$$

Here, $FJ_{d,t}$ describes that the first occurrence of the three constituent events, denoted as $l_{i1}^1, l_{i2}^1, \text{ and } l_{i3}^1$, should occur in one data update period, $FJ_{r,p}$ describes sensors $i1, i2, i3$ are neighbors.

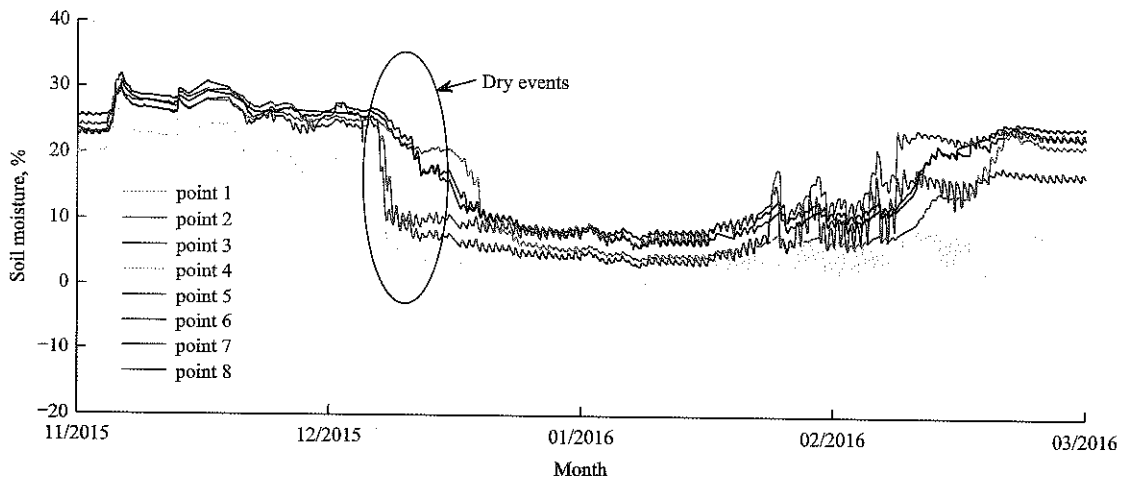


Figure 7 Dry events in December 2015

Considering FC functions of a dry event (de), time attributes t and d are equal to the corresponding attributes

of the first constituent event l_i . The region (r) of it contains all regions of constituent events and position (p)

of it is the centroid of the region (r) of ie , i.e.

$$\begin{cases} re.r = polygon(i_1^1.r, i_1^2.r, i_1^3.r) \\ re.p = centroid(re.r) \end{cases} \quad (13)$$

Considering consumption strategies (IS), all occurrences of constituent events in the window should be used to detect the complex event, so IS_C is *cumulative*. All occurrences in the window should be remained to detect other dry events, so IS_R is *reserve*. Meanwhile, three occurrences of each constituent events are needed, so length of the sliding windows of them are 3.

“Seepage event”

Seepage from a point to nearby points in the farmland usually implies leak of the irrigation device at the point. An example the dynamic process is illustrated in Figure 8. When a seepage event occurs, an increase event (i_{center}) and a series of higher events (h_{center}) occur at the nearest position from the leak point and then increase events and higher events occur at nearby positions followed. Thus, FJ functions are:

$$\begin{cases} FJ_{occr} = any(h_1, h_2, h_3, \dots, 3) \\ FJ_{d,t} = after(h_{i1}.t, i_{i2}.t) \wedge after(h_{i1}.t, i_{i3}.t) \\ FJ_{r,p} = inside(h_{i2}.p, h_{i1}.r) \wedge inside(h_{i3}.p, h_{i1}.r) \end{cases} \quad (14)$$

Here, when any three higher events occur, the detection starts. Three actual occurrences are denoted as h_{i1} , h_{i2} and h_{i3} . $FJ_{d,t}$ limits that higher events of sensor $i2$ and $i3$ will follow an occurrence of higher event of sensor $i1$, meanwhile $FJ_{r,p}$ limits that $i2$ and $i3$ is in the region of $i1$, implying sensor $i1$ is the nearest from the leak device and $i2$ and $i3$ are nearby. Compared with irrigation event, in irrigation event more than three higher events occur in the same data update period, but in seepage event, h_{i2} and h_{i3} occurs later than h_{i1} .

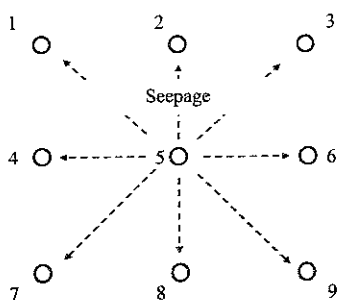


Figure 8 An example of seepage

Considering FC functions, attributes t , d , r , and p are equal to the corresponding attributes of h_{i1} of the nearest

sensor.

Considering consumption strategies (IS), all occurrences of constituent events in the window should be used to detect the complex event, so IS_C is *cumulative*. All occurrences in the window should be remained to detect other dry events, so IS_R is *reserve*. Meanwhile, IS_W is time window, and the seepage speed determines length of the sliding window.

4 Comparison and discussion

4.1 Comparison with threshold inequation

In this section, the complex events with widely-used threshold controls are compared. In threshold control for devices in farmland, an action is triggered by some threshold inequations, i.e. a situation that the value or the change becomes more or lower than a threshold, similar with our atomic events. However, it is too simple to model composite situations, leading to higher rates of false positives and omissions than complex event processing.

Figure 9 illustrates four composite situations should be detected, including two sensor damages, one rain and one no rain. With the constructions of complex events in farmland, the three complex events will be correctly detected and the one no rain situation is correctly ignored.

However, with threshold control, if rains are detected only by using rain sensor in weather station, we can set the threshold as 0.2 mm. From the data in Figure 9, two rains will be detected, in that the first one is false positive. If composite situations are detected only by using a single soil sensor, 4 kinds of composite situations cannot be distinguished: sensor damage, rain, irrigation and seepage. Assuming the change of moisture from a sensor over 5% is threshold to detect a rain, from the data in Figure 9, three rains are detected, but only the second is correct, and the other two detections are actually sensor damages.

In fact, our complex event model is compatible with threshold control, because the 4 kinds of atomic events are similar with 4 kinds of over-threshold situations. Based on that, more composite situations can be constructed with mathematical logic, temporal logic and spatial logic in our model. Thus, our complex event model and constructions of several complex events are

more effective for describe composite situations in farmland.

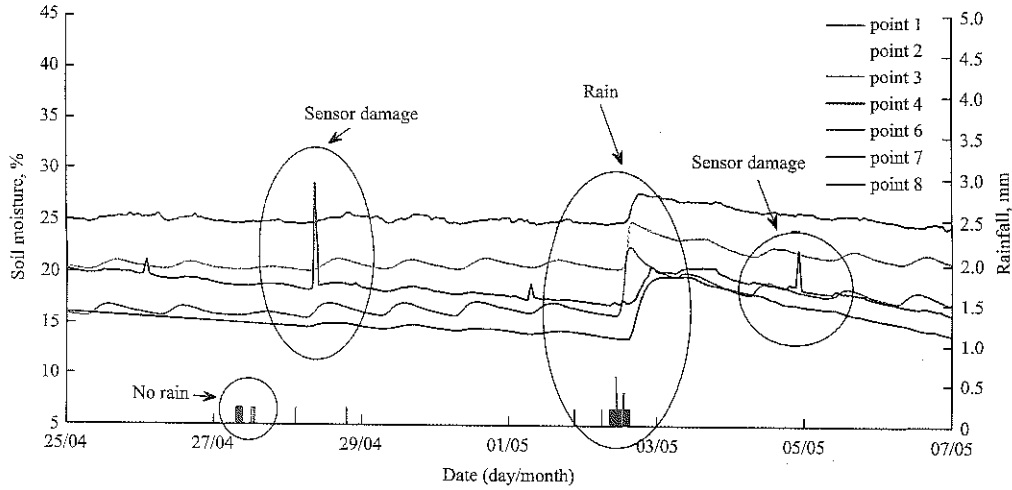


Figure 9 Four composite situations: 1) no rain on Apr. 27, 2) sensor damage on Apr. 28, 3) rain on May 02, and 4) another sensor damage on May 05

4.2 Comparison with production rules

/*Our Model*/

Rain: {ES: { $h_{ws}, i_{ws}, h_1, h_2, \dots, i_1, i_2, \dots$ }

IS: $IS_U = recent, IS_C = removeall, IS_W = lengthwindow(1)$

FJ: $\begin{cases} FJ_{occur} = dis(h_{ws}, i_{ws}) \wedge any(h_1, h_2, \dots, i_1, i_2, \dots, 3) \\ FJ_{d,t} = sync(h_{ws}.t, i_{ws}.t, e_{i1}.t, e_{i2}.t, e_{i3}.t, 2) \end{cases}$

FC: $\begin{cases} re.t = h_{ws}.t \text{ or } i_{ws}.t, re.d = h_{ws}.d \text{ or } i_{ws}.d \\ re.r = polygon(h_{ws}.r, i_{ws}.r, e_{i1}.r, e_{i2}.r, e_{i3}.r) \\ re.p = centroid(re.r) \end{cases}$

$$\begin{cases} R_1 : P_1 \vee P_2 \rightarrow P_3 \\ R_2 : P_4 \wedge P_5 \wedge P_6 \rightarrow P_{22} \\ R_3 : P_4 \wedge P_5 \wedge P_7 \rightarrow P_{23} \\ \dots \\ R_{817} : P_{19} \wedge P_{20} \wedge P_{21} \rightarrow P_{837} \\ R_{818} : (P_{22} \vee \dots \vee P_{837}) \rightarrow P_{838} \\ R_{819} : P_3 \wedge P_{838} \rightarrow P_{839} \\ R_{820} : P_{840} \wedge P_{841} \dots \wedge P_{849} \rightarrow P_{850} \\ R_{821} : P_{839} \wedge P_{850} \rightarrow P_{851} \end{cases} \quad (17)$$

(16)

Assertions: $\begin{cases} P_1 : h_{ws}.occur = true \\ P_2 : i_{ws}.occur = true \\ P_3 : h_{ws} \text{ or } i_{ws} \text{ occur} \\ P_4 : h_1.occur = true \\ P_5 : h_2.occur = true \\ \dots \\ P_{21} : i_9.occur = true \\ P_{22} : h_1, h_2, h_3 \text{ occur} \\ P_{23} : h_1, h_2, h_4 \text{ occur} \\ \dots \\ P_{837} : i_7, i_8, i_9 \text{ occur} \\ P_{838} : m_1 = true \\ P_{839} : m_2 = true \\ P_{840} : |h_{ws}.t - i_{ws}.t| \leq 2 \\ P_{841} : |h_{ws}.t - e_{i1}.t| \leq 2 \\ \dots \\ P_{849} : |e_{i2}.t - e_{i3}.t| \leq 2 \\ P_{850} : m_3 = true \\ P_{851} : re.occur \end{cases}$

Rain complex event is constructed with our model and production rules respectively in (16) and (17). From the comparison, readability of production rules is poor. Too many temporary symbols, assertions and rules are defined and there are so many complicated chain triggering relationships. It is difficult for users to check and eliminate loops, conflicts, redundancies, implications and even correctness of them. Conversely, construction with our model is far better readable.

4.3 Comparison with SASE+, Esper and Zang models

/*SASE+ Model*/

PATTERN: $\{(SCAN\ h_{ws} \vee SCAN\ i_{ws}) \wedge (SCAN * h_1[], SCAN * i_1[], SCAN * h_2[], SCAN * i_2[], \dots)\}$,

WHERE:

$$\begin{cases} \{h_1.length + i_1.length + h_2.length + i_2.length + \dots \geq 3, \\ |h_1.t - h_2.t| \leq 2, |h_1.t - h_3.t| \leq 2, |h_1.t - h_4.t| \leq 2, \dots \\ |h_2.t - h_3.t| \leq 2, |h_2.t - h_4.t| \leq 2, |h_2.t - h_5.t| \leq 2, \dots \\ |h_3.t - h_4.t| \leq 2, |h_3.t - h_5.t| \leq 2, |h_3.t - h_6.t| \leq 2, \dots \\ \dots \} \end{cases} \quad (18)$$

WITHIN: 1

RETURN:

$$\{h_{ws}.t, i_{ws}.t, \\ h_{ws}.d, i_{ws}.d, \\ r = f_1(h_{ws}.r, h_{ws}.r, h_{i1}.r, h_{i2}.r, h_{i3}.r) \\ p = f_2(r)\}$$

Rain complex event is constructed with the most popular SASE+ model in (18). From the comparison, the construction with SASE+ is far more complex and has poor readability than that with our model.

Firstly, SASE+ model does not support *any* semantics, so we have to define a inequation with lengths of occurrences sets of all constituent events. It is not intuitive. Conversely, in our model, the relationship is constructed as *any*($h_1, h_2, \dots, i_1, i_2, \dots, 3$), which is more easily understood.

Secondly, because in SASE+ model, temporal logic relationships have to be described in mathematical equations inequation. The values of i and j in the inequation $|h_{i,t} - h_{j,t}| \leq 2$ are all permutations of 1 to N , if there are N atomic events. Conversely, in our model, the relationship is constructed with a simple temporal operator *sync*($h_{ws}.t, i_{ws}.t, e_{i1}.t, e_{i2}.t, e_{i3}.t, 2$). Details of the semantics of *sync* are processed by the CEP algorithm, and do not need explicit specification.

Thirdly, SASE+ model cannot support spatial logic, so the functions (f_1 and f_2) to calculate p and r of the rain event should be defined explicitly. f_1 is a function to find a minimum polygon that covers regions of all constituent events. f_2 is a function to find the centroid of the irregular polygon. Both of them involve too difficult geometries and their formulas will be very complicated to general users. Conversely, in our model, details are also hidden, and only two operators *polygon* and *centroid* are needed.

The model with three popular complex event models is compared: SASE+ (2007), Esper (2016) and Zang (2007). The results are listed in Table 5. All models support mathematical logic and temporal logic. However, besides our model, others cannot support spatial logic and personalized occurrence consumption strategies for different constituent events. Meanwhile, readability of our model and language is better than the models of SASE+ and Esper.

Table 5 Comparison with other complex event models. “s” denotes supported, and “u” denotes unsupported

Functions	This paper	SASE+	Esper	Zang
Mathematical logic	s	s	s	s
Temporal logic	s	s (partial)	s (partial)	s
Spatial logic	s	u	u	u
Personalized occurrence consumption strategies	s	u	u	u
Custom attributes	s	s (partial)	s	s
Readability	good	not too bad	bad	good

5 Conclusion

A novel model is proposed for complex events from sensor network distributed deployed in farmland. A relevant XML-based specification language is also defined. The model is suitable to construct farmland complex events because various influencing factors of judgement and construction are considered in a comprehensive manner. Especially, the model supports to describe mathematical logic, temporal logic and spatial logic relationships among constituent events. Spatial logic is not supported by other traditional complex event models. By analyzing data from actual farmland sensor network, five kinds of atomic events and five kinds of common complex events i.e. sensor damage, rain, irrigation, dry and seepage are constructed with the model and specified with the language. Tests indicate that our event model and language are more effective to construct farmland complex events and have better readability than threshold control, production rules and complex event models in other CEP algorithms. In the future, more complex events can be constructed with the event model.

Acknowledgements

This research is financially supported by the National Natural Science Foundation of China (No. 61601471) and Beijing Natural Science Foundation (No. 4164090).

[References]

- [1] Buchmann, A., and B. Koldehofe. 2009. Complex event processing. *IT-Information Technology Methoden und innovative Anwendungen der Informatik und Informationstechnik*, 51(5): 241–242.
- [2] Chakravarthy, S., and D. Mishra. 1994. Snoop: an expressive event specification language for active databases. *Data & Knowledge Engineering*, 14(1): 1–26.

- [3] Diao, Y., N. Immerman, and D. Gyllstrom. 2007. SASE+: an agile language for kleene closure over event streams. UMass Technical Report.
- [4] EsperTech. 2016. Esper: event series analysis and complex event processing for Java. EsperTech Inc. <http://www.espertech.com/products/esper.php>. (accessed in July, 2017)
- [5] Gatziau, S., and K. R. Dittrich. 1992. SAMOS: an active object-oriented database system. *IEEE Data Engineering Bulletin*, 15(1-4): 23–26.
- [6] Gehani, N. H., H. V. Jagadish, and O. Shmueli. 1992. Event specification in an active object-oriented database. *ACM SIGMOD Record*, 21(2): 81–90.
- [7] Haller, S., S. Karnouskos, and C. Schroth. 2008. The internet of things in an enterprise context. *September, 2008: Future Internet Symposium*. Springer, Berlin, Heidelberg.
- [8] Janssen, J. A. E. B., M. S. Krol, R. M. J. Schielen, A. Y. Hoekstra, and J. L. de Kok. 2010. Assessment of uncertainties in expert knowledge, illustrated in fuzzy rule-based models. *Ecological Modelling*, 221(9): 1245–1251.
- [9] Jun, C., and C. Chi. 2014. Design of complex event-processing IDS in internet of things. *January, 2014: Measuring Technology and Mechatronics Automation (ICMTMA), 2014 Sixth International Conference of IEEE*.
- [10] Krämer, J., and B. Seeger. 2009. Semantics and implementation of continuous sliding window queries over data streams. *ACM Transactions on Database Systems (TODS)*, 34(1): 4.
- [11] Liu, Y. 2007. Research on some key technologies of agricultural IOT data processing based on the process of agricultural production. PhD Diss. Beijing: Beijing University of Posts and Telecommunications.
- [12] Magid, Y., G. Sharon, S. Arcushin, I. Ben-Harrush, and E. Rabinovich. 2010. Industry experience with the ibm active middleware technology (amit) complex event processing engine. *July, 2010: Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems*.
- [13] Wu, E., Y. Diao, and S. Rizvi. 2006. High-performance complex event processing over streams. June, 2006: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data.
- [14] Wang, F. J., X. M. Zhang, Y. H. Wang, and K. N. Cao. 2013. The research on complex event processing method of internet of things. *January, 2013: Measuring Technology and Mechatronics Automation (ICMTMA), IEEE Fifth International Conference*.
- [15] Zang, C., and Y. Fan. 2007. Complex event processing of real time enterprises based on smart items. *Chinese Journal of Mechanical Engineering*, 43(2): 22–32. (In Chinese with English abstract)

Appendix I

A higher atomic event

```

<atom>
  <id>h1</id>
  <sources>
    <src>sensor1</src>
  </sources>
  <attributes>
    <attr id="dx" type="d_x" var="false">5</attr>
    <attr id="dy" type="d_y" var="false">5</attr>
    <attr id="px" type="d_x" var="false">0</attr>
    <attr id="py" type="d_y" var="false">0</attr>
    <attr id="t" type="t" var="true" />
    <attr id="p" type="p" var="true" />
    <attr id="value" type="o" var="true" />
  </attributes>
  <jfunc>value>44.45%</jfunc>
</atom>

```

Appendix II

Sensor damage complex event

```

<ce>
  <id>sensordamage</id>
  <otherattrs />
  <eventset>
    <event>
      <evn>h1</evn>
      <chosenstrategy>recent </chosenstrategy>
      <removestrategy>removeall</removestrategy>
      <lengthwindow>1</lengthwindow>
    </event>
    ...
  </eventset>
  <judgefunction logic="conjunction">
    <variables>
      <var id="h1">
        <evn>h1|h2|...</evn>
        <occ>1</occ>
      </var>
      <var id="u1t">
        <evn>u1|u2|...</evn>
        <occ>1</occ>
        <attr>t</attr>
      </var>
      ...
    </variables>
    <jfunc>con(dis(h1,11),not(h2,12,h3,13...)</jfunc>
    <jfunc>sync(u1t,u2t,u3t,...,2)</jfunc>
    <jfunc>inside(u1p,u2r)</jfunc>
    <jfunc>inside(u1p,u3r)</jfunc>
    ...
  </judgefunction>
  <calculationfunction>
    <variables>
      <var id="h1d">
        <evn>h1</evn>
        <occ>1</occ>
        <attr>d</attr>
      </var>
      ...
    </variables>
    <cfunc attr="d">h1d</cfunc>
    <cfunc attr="t">h1t</cfunc>
    <cfunc attr="p">h1p</cfunc>
    <cfunc attr="r">h1r</cfunc>
  </calculationfunction>
</ce>

```