Research Article

Gerald Plesniewicz* and Baurzhan Karabekov

Specifying temporal knowledge for workflows ontologies

DOI 10.1515/comp-2016-0020
Received May 20, 2016; accepted Jul 04, 2016

Abstract: A workflow is an automation of a process, in which participants (people or programs) are involved in activities for solving a set of tasks according to certain rules and constraints in order to attain a common goal. The concept of workflow appeared in business informatics. Currently the workflow techniques are used in many other fields such as medical informatics, bioinformatics, automation of scientific research, computer-aided design and manufacturing, etc. An ontology is a formal description (in terms of concepts, entities, their properties and relationships) of knowledge for solving a given class of problems. In particular, ontologies can be used in problems related to workflows. In this paper, we introduce a formalism that extends the language of the interval Allen’s logic, and show how this formalism can be applied to specify temporal knowledge in ontologies for workflows. For the extended Allen’s logic, we construct a deduction system based on the analytic tableaux method. We also show (by examples) how to apply the deduction method to query answering over ontologies written in the extended Allen’s logic.

Keywords: ontologies; workflows; temporal logics; Allen’s logic; logical deduction; analytic tableaux; query answering

1 Introduction

A workflow is an automation of a process, in which participants (people or programs) are involved in activities for solving a set of tasks according to certain rules and constraints in order to attain a common goal. Successful completion of the process depends on the correct synchronization and scheduling of the tasks. The concept of workflow appeared in business informatics. Currently, the workflow techniques are used in many other fields such as medical informatics, bioinformatics (in particular, genomics [1]), automation of scientific research [2], computer-aided design and manufacturing, etc. Workflows are applied to the problems of designing Web services [3]. The concept of workflow is the central in Process-Aware Information Systems (PAIS) [4].

An ontology is a description (in terms of concepts, entities, their attributes and relationships) of knowledge for solving a given problem or class of problems. In formal ontology, concepts are formal.

A formal concept with the name $C$ is represented by a triple $(U^C,E^C,\neg C)$, where $U^C$ is the universe of the concept (i.e. the set of possible names for denoting instances of the concept), $E^C$ is a subset of $U^C$ consisted of all names denoting instances of the concept, and $\neg C$ is an equivalence relation, so-called, co-referentiality relation. Co-referential names denote the same object of the modelled problem domain [5]. The pair $\text{Ext}^C = (E^C,\neg C)$ is the extension of the concept $C$.

We can use the language LSS from the system “Binary Model of Knowledge” for defining concept universes [5]. A typical LSS statement allows us to define the universe of the concept as a set of tuples containing surrogates $s$ (i.e. object identifiers) and expressions of the form $A:s,A:\{s_1,\ldots,s_k\}$ and $A:v$. Here $A$ is an attribute name, $s_i (1 \leq i \leq k)$ are surrogates, and $v$ is a member of a data type.

In workflow ontology, the main role is played by events, i.e. by concepts whose instances exist at temporal intervals. Thus, the tuples representing these instances should include the attributes $\text{Beg}$ and $\text{End}$ that mark the beginning and the end of the temporal intervals.

Example 1. Consider a simple example of a workflow that represents a business process whose aim is transportation of goods (this workflow is shown graphically in Figure 1). Each transportation has to be carried out by one of two firms “TransVan” or “AutoTrans”.

“TransVan” is used for a transportation if and only if some condition $p$ is satisfied. The business process starts with an order processing (task $A$). Then, the invoice is registered (task $D$) and the goods transportation is carried out...
Figure 1: Example of workflow scheme

(tasks B or C). Finally, the unloading of goods and registration of the payment is carried out (tasks E and F). The performance of each task from this scheme requires some time. So, certain temporal intervals are associated with the tasks. We denoted these intervals by the same letters as the tasks.

The structure of the instances of the concept Firm can be defined, for example, by the following statement of the language LSS from the system “Binary Model of Knowledge” (LSS is a language for specifying concept universes):

\[
\text{Firm [Name: String, Address: String, Owner: Person, Tariff: Integer]},
\]

where Person is the concept with the structure defined by the statement (for example)

\[
\text{Person [Name: String, Year-of-Birth: Integer, SSN: 3 Digit-2Digit-4Digit]}.
\]

Tuples from concept extensions define certain facts. For example, suppose that E_Firm contains the tuple

\[
\text{[Name: TransVan, Address: ‘‘Tverskaya str. 22, Moscow’’, Owner: #21]}
\]

with the surrogate #7, and E_Person contains the tuple

\[
\text{[Name: ‘‘A.P. Petrov’’, Year-of-Birth: 1960, SSN: 078-05-1120]}
\]

with the surrogate #21. Then we have the following facts:

- #7. Name=TransVan ("TransVan" is the name of the firm with surrogate #7);
- #7. Owner.Name=’A.P. Petrov’ (A.P. Petrov is the name of the owner of the form with the surrogate #7);

Binary relations such as “before”, “during” etc. can be observed between the events. For example, the event "goods transportation by the firm TransVan" occurs before the event "uploading of goods" in the considered workflow. Formally, this can be written as the following statement in the language LTS from the system “Binary Model of Knowledge”:

\[
\text{Transportation (Firm.Name=TransVan) BEFORE Unloading.}
\]

In this statement, the first term denotes the set of all x from the extension of the concept Transportation such that x.Firm.Name=TransVan.

We can consider these binary relations also as relations between temporal intervals of events. “Before” is one of basic relation of the interval Allen’s logic [6]. Other basic relations are “equals”, “during”, “finishes”, “meets”, “overlaps” and “starts”. An arbitrary sentence of Allen’s logic is interpreted as a binary relation between two temporal intervals that is a disjunction of several basic relations.

We can write the LTS statement that the invoice registration precedes the transportation of goods by TransVan if p is true:

\[
\text{IF P THEN InvoiceRegistration BEFORE Transportation (Firm = TransVan).}
\]

In the same way we can write the LTS statement

\[
\text{IF NOT P THEN InvoiceRegistration BEFORE Transportation (Firm = AutoTrans).}
\]

Suppose now that there are the following assumptions:

1. order processing lasts 1 time unit;
2. invoice registration lasts 2 time units;
3. transportation by TransVan lasts 50-60 time units;
4. transportation by AutoTrans lasts 45-60 time units;
5. payment registration lasts ≤ 1 time unit;
6. if a certain condition q is satisfied, then an unloading of goods and the payment registration begin at the same time, but after the payment registration the unloading lasts for at least 35 time units and no more than 40 time units;
7. if the condition of q is not satisfied, then a payment registration is delayed by 1 time unit and further an unloading is carried out 40 time units;
8. if p is satisfied, then q is not satisfied.

The assumptions (1) – (8) can be written in LTS as follows:

(1*) IF X IN OrderProcessing THEN X.End − X.Beg = 1,
(2*) IF X IN InvoiceRegistration THEN X.End − X.Beg = 1,
(3*) IF X IN Transportation (Firm.Name = TransVan) THEN
50 ≤ X.End − X.Beg ≤ 60,
(4*) IF X IN Transportation (Firm.Name = AutoTrans) THEN
50 ≤ X.End − X.Beg ≤ 60,
(5*) IF X IN PaymentRegistration THEN X.End − X.Beg = 1,
(6*) Transportation (Firm = TransVan)
BEFORE (2.Beg − 1.End)
InvoiceRegistration,
(7*) IF Q THEN UnloadingSTARTS (35 ≤ 2.End − 1.End ≤ 40)
PaymentRegistration,
(8*) IF P THEN NOT Q.

In (7*), 2.End is the shorthand for PaymentRegistration.END.

The expression STARTS(35 ≤ 2.End − 1.End ≤ 40) in (7*) represents the relation “starts” with the constraint that the first interval (for the event PaymentRegistration) starts the second interval (for the event Unloading) and the residue of the second interval has length from 35 to 45.

Thus, for the language LTS it is necessary to extend the interval Allen’s logic by including propositional logic and metric information (i.e. distances between time points).

In this paper, we construct such extension of Allen’s logic and formulate the inference rules that determine a method of deduction in the style of analytical tables. We show how the method of deduction can be used to query answering over workflow ontologies. For the extended Allen’s logic, we choose abstract syntax which agrees with mathematical notation adopted by Allen in his logic.

Remark. Various logical languages are used for workflow modeling. In particular, the following logics are applied: propositional logics [8], logic CTL (Concurrent Transaction Logic) [9], linear temporal logic [10]. However, these logic do not allow to represent metric information in the workflow ontologies.

Table 1: Basic relations of Allen’s interval logic.

<table>
<thead>
<tr>
<th>Interval relation</th>
<th>Illustration</th>
<th>Inequalities and equalities for intervals ends</th>
</tr>
</thead>
<tbody>
<tr>
<td>A b B</td>
<td>==A==</td>
<td>A− ≤ B−</td>
</tr>
<tr>
<td>A m B</td>
<td>=A=</td>
<td>A− = B−</td>
</tr>
<tr>
<td>A o B</td>
<td>==A==</td>
<td>A− &lt; B−, B− &lt; A−</td>
</tr>
<tr>
<td>A d B</td>
<td>==B==</td>
<td>A− &lt; B+, A+ &lt; B−</td>
</tr>
<tr>
<td>A s B</td>
<td>==A==</td>
<td>A− = B−, A− &lt; B+</td>
</tr>
<tr>
<td>A f B</td>
<td>==B==</td>
<td>B&lt; − A−, A+ = B+</td>
</tr>
<tr>
<td>A e B</td>
<td>==A==</td>
<td>A− = B−, A+ = B+</td>
</tr>
</tbody>
</table>

2 Metric Boolean extension of Allen’s interval logic

In Allen’s interval logic (let us denote it by LA), there are 7 basic relations between intervals: e (equals), b (before), m (meets), o (overlaps), s (starts), d (during). The inverted relations are marked by asterisks: b∗ (after), m∗ (met-by), o∗ (overlapped-by), s∗ (finished-by), d∗ (contains). So, A a B ⇔ A α A, where α is basic relation and A, B are temporal intervals. Let us denote by A− and A+ the beginning and the end of a temporal interval. We consider the integers as time points. Thus, A = [A−, A+] = {x ∈ Integer | A− ≤ x ≤ A+}. Table 1 shows the basic relations of Allen’s interval logics.

A sentence (formula) of LA is an expression of the form A ω B, where ω is any subset of the set Ω = {e, b, m, o, f, s, d, b∗, m∗, o∗, f∗, s∗, d∗} and A, B are interval names. If ω = {α}, then we have a primitive sentence, and instead of A{α}B we write A α B. If ω = {α1, α2, ..., αn} then we write A α1 α2 ... αn B instead of A{α1, α2, ..., αn}B. The formula A ω B is interpreted as a disjunction of the primitive formulas A α B with a ∈ ω.

Boolean extension LA+ of Allen’s interval logic LA has the following syntax: (a) propositional variables and LA formulas are LA+ formulas; (b) ∼ φ, (φ ∧ ψ), (φ ∨ ψ) are LA+ formulas if φ and ψ are LA+ formulas. We also introduce φ → ψ as the abbreviation of ∼ φ ∨ ψ.

The semantics of LA+ formulas is obtained by combining the semantics of LA formulas with the semantics...
of Boolean connectives: “¬φ” = “¬φ”, “φ ∧ ψ” = “φ” ∧ “ψ” and “φ ∨ ψ” = “φ” ∨ “ψ”.

In metrical Boolean extension μLA+ of Allen’s interval logic LA, primitive formulas have the form A α (A) B, where α ∈ Ω and λ is a constraint which is a conjunction of inequivalences and equalities with time points A−, A+, B, B+.

For example, the primitive formula

A o(b−A−)=3; A−A− ≥ 1) B

has the constraint b−A−=3; A−A− ≥ 1 and it is interpreted as follows: “A o(b−A−=3; A−A− ≥ 1) B” = 1 if and only if (B−A−=3) ∧ (A−A− ≥ 1) ∧ (B−B+ > 0).

An interval ontology is a finite set of μLA+ formulas.

Example 2. Here are two examples of interval ontologies:

O1 = (p → A b(b−A− ≥ 3) B, Bb(2 ≤ B−A− ≤ 4) C, A o(b−D−) ≥ 3) D, p ∧ Do(b−C−) ≥ 3) C)

O2 = (|A| = 1, AmD, |D| = 2, p → Db(b−D− = 1) C, ~ p → D b(c−D− = 1) C, |F| = 1, 50 ≤ |B| ≤ 60, 45 ≤ |C| ≤ 60, q → Fs(35 ≤ E−F− ≤ 40) E, Bb(F−B− = 1) E, Bb(E−B−) E, ~ p → q, q → Fd(F−E− ≥ 1; E−F− = 40) E).

The second interval ontology represents knowledge about the workflow from Example 1. Here A, B, C, D and E are the names we use for brevity instead of the event names OrderProcessing, InvoiceRegistration, Transportation, PaymentRegistration and Unloading (correspondingly).

As in every logic, there is the relation “|=” of logical consequence in μLA+. Let O be an interval ontology and φ be a μLA+ sentence. Then O |= φ if and only if there is no interpretation such that all sentences of O are true but φ is false.

3 Deduction and query answering in the logic μLA+

We shall define the analytic tableaux deduction system for the logic μLA+.

Table 2 and Table 3 contain the inference rules for this deduction system. The rules from Table 3 are obtained from the semantics of the basic Allen’s relations.

Take, for example, the eleventh rule of Table 3 (let us denote this rule by T3(11)). This rule corresponds to the assertion that A s B implies (A−B−) ∧ (A−<B+), and so, (A−B− ≥ 0) ∧ (B−A− ≥ 0) ∧ (B−A+ ≥ 1). The rule T3(6) corresponds to the assertion that A d B implies

~ [(B−A− ≥ 0) ∧ (A−A− ≥ 0) ∧ (A−B+ ≥ 0)], and so, (B−A− ≥ 0) ∧ (A−A− ≥ 0) ∧ (A−B+ ≥ 0). The rule T3(15) is actually a scheme of rules. For example, let α be o and λ be the constraint b−A− ≥ 2; B−A− ≥ 3. Then we obtain the rule which from +A o(b−A− ≥ 2; B−A− ≥ 3) B derives two formulas +A o B and +B−A− ≥ 2; B−A− ≥ 3 and adds them as the “fork” to any branch passing through this formula.

<table>
<thead>
<tr>
<th>Table 2: Inference rules for propositional</th>
</tr>
</thead>
<tbody>
<tr>
<td>antecedent</td>
</tr>
<tr>
<td>+ ¬φ</td>
</tr>
<tr>
<td>− ¬φ</td>
</tr>
<tr>
<td>+φ ∧ ψ</td>
</tr>
<tr>
<td>− φ ∧ ψ</td>
</tr>
<tr>
<td>+φ ∨ ψ</td>
</tr>
<tr>
<td>− φ ∨ ψ</td>
</tr>
<tr>
<td>+φ → ψ</td>
</tr>
<tr>
<td>− φ → ψ</td>
</tr>
</tbody>
</table>

Consider an example of deduction in the logic μLA+.

Example 3. Take the ontology O1 from the Example 2 and the μLA+ formula B d(D−D− ≥ 5) D. Let us prove that O1 |= B d(d(D−D− ≥ 5) D). For this purpose we construct the inference tree starting with the branch

+p → A b(b−A− ≥ 3) B, +B b(2 ≤ B−A− ≤ 4) C, +A o(b−D− ≥ 3) D, p ∧ D o(d−C− ≥ 3) C

(see Figure 2).

At step 1 the rule T2(3) is applied to +p ∧ A m B. As a result the formulas +p and +A m B are added to the branch. Also the label “[1]” is written on the right of +A m B and the labels “1:” are written on the left of these formulas. At step 9 the rule T3(6) is applied to −B d(d−D− ≥ 5) D and as a result the “fork” of the formulas −B d D and −D− ≥ 5 is added to the current branch. The resulting inference tree has four branches. For each branch, write down all inequivalences (without signs “+” and “−”) and add the standard inequivalences A−A− ≥ 1, B−B− ≥ 1, C−C− ≥ 1, A−A− ≥ 1. Let S1, S2, S3 and S4 be the obtained sets of the inequivalences. These sets of inequivalences are inconsistent. Indeed, for example, S1 contains the inequalities D−D− ≥ 4, A−A− ≥ 1, B−A− ≥ 0, B−B− ≥ 1, C−B− ≥ 2, D−C− ≥ 2. Adding these inequalities up, we obtain (D−D−)+(A−A−)
We have considered the problem of specifying temporal relations in workflows. For solving this problem we have developed the logic $\mu L A^+$ which is an extension of the Allen's interval logic. The deduction system in the style of analytic tableaux has been defined. We have shown how to find, by applying the method of analytic tableaux, answers to queries for workflow ontologies.

### Acknowledgement:
This work was supported by Russian Foundation for Basic Research (project 14-07-0387) and Ministry of Education and Science of Kazakhstan (project 0115 RK 00532)

### References


[8] Bi H.H., Zhao J.L., Applying propositional logic to workflow verification, Information Technology and Management, 2004, 3-4(S), 293-318
