УДК 004.62

К.А. Кудим, В.А. Резниченко, Г.Ю. Проскудина, О.М. Овдей

ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ АСПЕКТЫ ОТНОШЕНИЙ И ОПЕРАЦИЙ ДЛЯ ПОСЛЕДОВАТЕЛЬНОСТЕЙ ВРЕМЕННЫХ ИНТЕРВАЛОВ

Рассмотрены понятия момента времени, временного интервала и временной цепочки — последовательности интервалов времени. Определены основные отношения и операции над этими объектами. Приведены эффективные алгоритмы для реализации операций над временными цепочками. Рассматривается практическое применение временных цепочек.

Введение

Данные в компьютерных системах так или иначе связаны с различными событиями, интервалами времени. Отношения и операции над временными интервалами определены уже довольно давно и широко используются [1–3]. В этой работе исследуются такие структуры времени как последовательности интервалов времени или временные цепочки, которые можно рассматривать как обобщение временных интервалов. В работе определены отношения между временными цепочками, их свойства, а также операции над ними. Использование временных цепочек проиллюстрировано на практических примерах. Разработаны эффективные алгоритмы для реализации операций над временными цепочками.

1. Представление времени и временные последовательности

Понятие времени в компьютерных системах представляется как правило моментами времени либо временными интервалами. В дополнение к этим двум базовым представлениям введём ещё одно, а именно временную цепочку или, другими словами, последовательность временных интервалов.

Момент времени – точка на оси времени. Любое событие можно связать с фактом возникновения, которому ставится в соответствие момент времени.

Интервал времени – непустой отрезок временной шкалы. Как правило интервал времени задается моментами начала и конца. Некоторое утверждение или факт в предметной области может быть действительным в течение определенного интервала времени. То есть с интервалом может быть связан любой факт, который имеет продолжительность своего существования.

Временная цепочка – это упорядоченное множество интервалов времени. Интервал времени является частным случаем временной цепочки, состоящая лишь из одного интервала. С помощью временной цепочки можно выразить, когда имел место какой-либо факт или, другими словами, можно выразить истинность некоторого факта предметной области во времени. Например, в предметной области «Отдел кадров» фиксируется факт «Человек работает в отделе». Из приведенной далее таблицы видно, что И.В. Петров работал в отделе В1 два интервала времени, т. е. можно сказать, что с этим фактом связана временная цепочка.

ФИО	Отдел	Дата начала	Дата конца
Петров И.В	В1	22.06.2009	03.09.2010
Петров И.В	В3	03.09.2010	01.02.2011
Петров И.В	В1	01.02.2011	01.01.2013

Далее рассмотрим более формально упомянутые выше базовые конструкции времени.

© К.А. Кудим, В.А. Резниченко, Г.Ю. Проскудина, О.М. Овдей, 2013

2. Определения, отношения и операции над элементами времени

2.1. Точка

Определение. Пусть **P** – дискретное линейно упорядоченное множество. Назовём элементы данного множества точками. Это могут быть: моменты времени, целые числа и др. Точки будем обозначать строчными буквами латинского алфавита a, b, c, …

Отношения. $a < b$, $a = b$, $a \geq b$ …

Операции.

– Сдвиг точки на n единиц: $a + n$.

2.2. Интервал

Определение. Пусть $a, b \in \mathbf{P}$, тогда если $a < b$, то

$[a,b] = \{x \in \mathbf{P}: a \leq x \leq b\}$;

$[a,b) = \{x \in \mathbf{P}: a \leq x < b\}$;

$(a,b] = \{x \in \mathbf{P}: a < x \leq b\}$;

$(a,b) = \{x \in \mathbf{P}: a < x < b\}$ — интервалы.

Из данного определения видно, что интервал может как включать, так и не включать начальный и конечный моменты времени, являясь таким образом закрытым [a,b], открытым (a,b), закрыто-открытым [a,b) или открыто-закрытым (a,b] интервалом.

Интервал, не содержащий ни одной точки, назовём пустым и обозначим []. Множество интервалов обозначим **R**. Интервалы будем обозначать прописными буквами латинского алфавита A, B, C, ...

Длина интервала — это количество точек, содержащихся в интервале либо количество единичных интервалов, составляющих интервал, и задается операцией: $length([a,b)) = b - a$.

Определим операции для нахождения граничных точек.

Начало интервала: $begin(A) = a$, где

$A=[a,b] \vee A=(a,b] \vee A=[a,b) \vee A=(a,b)$.

Конец интервала: $end([a,b)) = b$, где

$A=[a,b] \vee A=(a,b] \vee A=[a,b) \vee A=(a,b)$.

Далее, для простоты изложения, мы ограничимся только закрыто-открытым интервалом A: $[a,b) = \{x \in \mathbf{P}: a \leq x < b\}$.

Отношения. Определим ряд полезных отношений между точкой и интервалом и между двумя интервалами.

Точка-интервал

1. Точка x находится до интервала A: $x < A \Leftrightarrow x < begin(A)$.

2. Точка a – начальная точка интервала A: $Begin(A, a) \Leftrightarrow begin(A)=a$.

3. Точка b – конечная точка интервала A: $End([A,b) \Leftrightarrow end(A)=b$.

Интервал-интервал

Дж. Аллен показал [1–2], что между двумя непустыми интервалами времени существует 13 основных бинарных отношений или предикатов взаимного расположения интервалов друг относительно друга, которые сейчас так и принято называть отношениями Аллена.

1. Интервал A находится перед интервалом B:

$A\ before\ B \Leftrightarrow A < B \Leftrightarrow end(A) < begin(B)$.

Из этого отношения вытекает следующее свойство:

$A\ before\ B \Rightarrow \exists C \in \mathbf{R}, C \neq \emptyset: A\ meets\ C\ meets\ B$.

2. Интервал A находится после интервала B:

$A\ after\ B \Leftrightarrow B\ before\ A \Leftrightarrow B < A \Leftrightarrow end(B) < begin(A)$.

3. Интервал A равен интервалу B:

$A\ equals\ B \Leftrightarrow A = B \Leftrightarrow begin(A) = begin(B)\ \&\ end(A) = end(B)$.

4. Интервал B следует непосредственно за A (или: интервал B касается интервала A справа):

$A\ meets\ B \Leftrightarrow end(A) = begin(B)$.

5. Интервал A следует непосредственно за B (или: интервал A касается интервала B справа):

$$A\ met\_by\ B \Leftrightarrow B\ meets\ A \Leftrightarrow end(B) = begin(A).$$

6. Интервал B перекрывает интервал A:

$$A\ overlaps\ B \Leftrightarrow begin(A) \le end(B)\ \&\ begin(B) \le end(A).$$

7. Интервал A перекрывает интервал B:

$$A\ overlapped\_by\ B \Leftrightarrow B\ overlaps\ A \Leftrightarrow begin(B) \le end(A)\ \&\ begin(A) \le end(B).$$

8. Интервал A находится в интервале B (или интервал B содержит интервал A):

$$A\ during\ B \Leftrightarrow A \subseteq B \Leftrightarrow begin(B) \le begin(A)\ \&\ end(A) \le end(B).$$

9. Интервал B находится в интервале A (или интервал A содержит интервал B):

$$A\ contains\ B \Leftrightarrow B\ during\ A \Leftrightarrow B \subseteq A.$$

10. Интервал A начинается одновременно с интервалом B:

$$A\ starts\ B \Leftrightarrow begin(A) = begin(B).$$

11. Интервал B начинается одновременно с интервалом A:

$$A\ started\_by\ B \Leftrightarrow begin(B) = begin(A).$$

12. Интервал A заканчивается одновременно с интервалом B:

$$A\ finishes\ B \Leftrightarrow end(A) = end(B).$$

13. Интервал B заканчивается одновременно с интервалом A:

$$A\ finished\_by\ B \Leftrightarrow end(B) = end(A).$$

Перечисленные отношения для удобства можно комбинировать. Например, часто используют следующее отношение.

Два интервала либо соприкасаются либо пересекаются:

$$Merges(A,B) \Leftrightarrow \neg(A\ before\ B)\ \&\ \neg(B\ before\ A) \Leftrightarrow \Leftrightarrow \neg(A < B)\ \&\ \neg(B < A) \Leftrightarrow \Leftrightarrow \neg(end(A) < begin(B)) \& \neg(end(B) < begin(A)) \Leftrightarrow \Leftrightarrow begin(B) \le end(A)\ \&\ begin(A) \le end(B).$$

Операции над интервалами. Выше были приведены операции для нахождения длины интервала и нахождения граничных точек. Часто для интервалов также используются операции объединения, пересечения, разности и сдвига:

1. Объединение двух интервалов:

$A \cup B = [\min(begin(A), begin(B)), \max(end(A), end(B)))$, если $Merges(A,B)$, в противном случае не определено.

2. Пересечение двух интервалов:

$A \cap B = [\max(begin(A), begin(B)), \min(end(A), end(B)))$, если A overlaps B; в противном случае не определено.

3. Разность двух интервалов A – B равна:

$[begin(A), \min(begin(B), end(A)))$, если $begin(A) < begin(B)\ \&\ end(A) \le end(B)$;

либо равна:

$[\max(end(B), end(A)), end(A)))$, ли $begin(A) \ge begin(B)\ \&\ end(A) > end(B)$,

в противном случае не определено.

4. Сдвиг интервала:

$$A + n = [begin(A)+n, end(A)+n).$$

Однако, рассматриваемые операции не являются замкнутыми относительно интервалов времени. Объединение двух непересекающихся интервалов возвращает два интервала. Разность двух интервалов может возвратить пустое множество, один или два интервала.

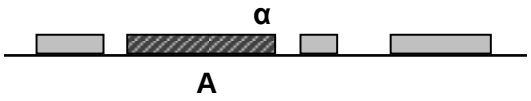
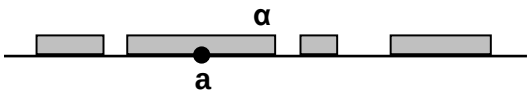
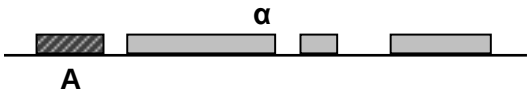
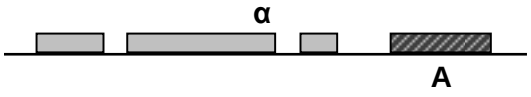
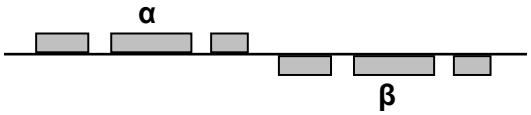
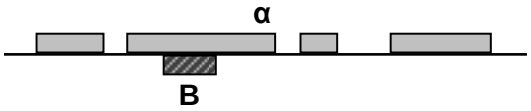
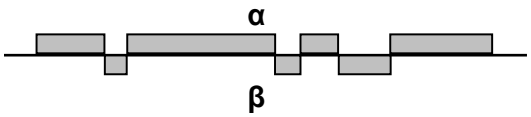
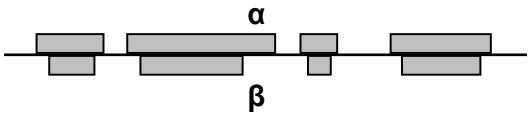
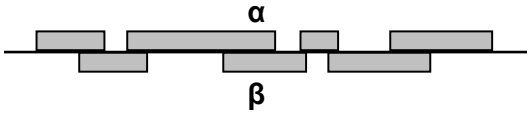
2.3. Цепочка

Определение. Последоватьльность непустых интервалов назовём *цепочкой*, если для любых двух соседних её интервалов найдётся непустой интервал, лежащий между ними. Формально:

α = <A1, ..., An>, где Ai ∈ **R**, Ai ≠ [), и для i=1…n-1 Ai before Ai+1.

Цепочку, не содержащую ни одного интервала, назовём пустой и обозначим <>. Множество всех цепочек обозначим **C**.

Отношения. Пусть α и β – цепочки, A, Ai ,B ,Bi – непустые интервалы, a – точка. Введем следующие отношения:

1. Интервал в цепочке: A ∈ α ⇔ α=<A1,...,An> & ∃i: A = Ai	
2. Точка принадлежит цепочке: a ∈ α ⇔ ∃A ∈ α: a ∈ A	
3. Первый интервал цепочки: First(α, A) ⇔ α=<A, B1,...,Bn>	
4. Последний интервал цепочки: Last(α, A) ⇔ α=<B1,...,Bn, A>	
5. Цепочка α предшествует цепочке β: α < β ⇔ last(α) < first(β)	
6. Интервал B входит в цепочку α: B⊆<A1,A2,..An> ⇔ B⊆A1 ∨ B⊆A2 ∨ ... ∨ B⊆An	
7. Цепочка β=<B1,...,Bn-1> дополняет цепочку α=<A1,...,An> ⇔ A1 meets B1 meets A2 meets B2 meets ... meets Bn-1 meets An Свойство: β дополняет α ⇒ α+β=<A1+B1+A2+B2+...+Bn-1+An>	
8. Цепочка β входит во все интервалы цепочки α: B1⊆<A1 & B2⊆A2 &.. & Bn⊆An	
9. Цепочка β перекрывает все промежутки цепочки α: ∀A∈α, ∃B∈β: A overlaps B	

Далее определим ряд полезных отношений для цепочек:

Операции над цепочками

1. Возвращает первый интервал цепочки:

first(α) = A: First(α, A).

2. Возвращает последний интервал цепочки:

last(α) = A: Last(α, A).

2a. Мощность:

cardinality(α) = n для α = <A1,...,An>.

2b. kth(α,k) = Ak для

α =<A1,...,Ak,...,An>, если 1 ≤ k ≤ length(α);

в противном случае не определено.

2c. Протяжённость цепочки:

length(α) = ∑ length(Ai).

2d. Охват цепочки или общая длина от начала первого интервала до конца последнего:

coverage(α) = end(An) — begin(A1).

3. Объединение цепочки с пустой цепочкой: α ∪ <> = α.

3a. <[]> = <>.

4. Объединение двух цепочек, одна из которых состоит из одного интервала:

<A1,...,An> ∪ <B> =

=<A1,...,Ak,[min(begin(Ak+1),begin(B)),

max(end(Al—1),end(B))],Al,...,An>,

где k=argmax[Ai<B](end(Ai)),

l = argmin[B<Ai](begin(Ai)).

5. Объединение двух цепочек:

α ∪ <B1, B2, ..., Bn> = (α ∪ <B1>) ∪

∪ <B2,..., Bn>.

Свойство: α∪α = α.

6. Пересечение цепочки с элементарной цепочкой:

<A1,...,An>∩<B> = <A1∩B> ∪ ...

... ∪ <An∩B>.

7. Пересечение двух цепочек:

α∩<B1,...,Bn> = (α∩B1)∪...∪(α∩Bn).

8. Разность цепочки и элементарной цепочки:

<A1,...,An> — <B> = <A1 — B> ∪ ...

... ∪ <An — B>.

9. Разность двух цепочек:

A — <B1,...,Bn> = (α — B1)∪...∪(α — Bn).

10. Сдвиг цепочки:

α + n = <A1 + n, ..., An + n>

11. Периодическая цепочка:

(α,n)* = ... ∪(α-2n)∪

∪ (α-n)∪α∪(α+n)∪(α+2n)+...

Из приведенных определений следует, что все рассматриваемые операции относительно цепочек, в отличие от аналогичных операций относительно интервалов, являются замкнутыми, т. е. в результате дают цепочку.

3. Примеры использования цепочек данных

Представим факт «Человек работает в отделе» из приведенного выше примера (см. раздел 1.) в виде цепочек интервалов.

ФИО	Отдел	Время работы
Петров И.В.	B1	<[1998,2001), [2010,2012)>
Петров И.В.	B3	<[1996–1998), [2002–2006), [2007–2010)>
Иванов И.И.	B1	<[1996,2000)>

Приведем примеры операций над цепочками данных, дающих ответы на содержательно сформулированные запросы с использованием этой структуры данных.

1. Когда Петров работал в отделе B1?

α = <[1998,2001), [2010,2012)>.

2. Когда Петров не работал в отделе B1?

α=<[1800,2013)>-<[1998,2001),[2010,2012)>=

= <[1800,1998), [2001,2010), [2012,2013)>.

3. Когда Петров и Иванов работали вместе в отделе B1?

α=<[1996,2000)>∩<[1998,2001),[2010,2012)>=

= <[1998,2000)>.

4. Когда Петров или Иванов работал в отделе B1?

α=<[1996,2000)>∪<[1998,2001),[2010,2012)>=

= <[1996,2001), [2010,2012)>.

Другой пример использования цепочек: регистрация факта пребывания пользователя на сайте X.

α=<[2013-04-14 09:02:33, 2013-04-14 12:22:35),

[2013-04-15 16:02:33, 2013-04-15 16:22:35),

[2013-04-15 18:12:33, 2013-04-15 18:32:33)>

В этом случае, используя операцию протяженности цепочки length(α), можно легко ответить на следующий запрос:

Сколько Сидоров провел времени на сайте X?

4. Алгоритмы операций над цепочками

Цепочку <A1,...,An> представим массивом M[1..n]. Интервал A представим парой целых чисел [A.begin, A.end). Q – большое число.

На рис. 1, 2, 3 соответственно показаны алгоритмы выполнения операций объединения, пересечения и дополнения данных, имеющих тип цепочка.

Сложность алгоритма линейная: если C1.length = n, C2.length = m, то сложность алгортима O(n+m).

```
                    CHAIN_UNION(M1, M2, M)
i ← 1
j ← 1
k ← 0
in1 ← false
in2 ← false
while true
do    if i ≤ M1.length
        then     A1 ← M1[i]
        else     A1.begin ← Q, A1.end ← Q

    if i ≤ M2.length
        then     A2 ← M2[i]
        else     A2.begin ← Q, A2.end ← Q

        if A2.begin = Q and A1.begin = Q
        then return

    if in1 and in2
    then    if A1.end ≤ A2.end
        then    in1 ← false, B.end ← A1.end, i ← i+1
        if A2.end ≤ A1.end
        then    in2 ← false, B.end ← A2.end, j ← j+1
            if A1.end = A2.end
              then    k ← k+1, M[k] ← B
    else if in1 and not in2
    then    if A1.end ≤ A2.begin
        then    in1 ← false, i ← i+1, k ← k+1, M[k] ← B
        if A2.begin ≤ A1.end
        then    in2 ← true
    else if not in1 and in2
    then    if A1.begin ≤ A2.end
        then    in1 ← true
        if A2.end ≤ A1.begin
        then    in2 ← false, j ← j+1, k ← k+1, M[k] ← B
    else if not in1 and not in2
    then    if A1.begin ≤ A2.begin
        then    in1 ← true, B.begin ← A1.begin
        if A2.begin ≤ A1.begin
        then    in2 ← true, B.begin ← A2.begin
M.length = k
```

Рис. 1. Алгоритм объединения двух цепочек

```
                CHAIN_INTERSECTION(M1, M2, M)
i ← 1
j ← 1
k ← 0
in1 ← false
in2 ← false
while true
do    if i ≤ M1.length
        then     A1 ← M1[i]
        else     A1.begin ← Q, A1.end ← Q

    if i ≤ M2.length
        then     A2 ← M2[i]
        else     A2.begin ← Q, A2.end ← Q

        if A2.begin = Q and A1.begin = Q
        then return

    if in1 and in2
    then    if A1.end ≤ A2.end
        then    in1 ← false
                i ← i+1, B.end ← A1.end
        if A2.end ≤ A1.end
        then    in2 ← false
                j ← j+1, B.end ← A2.end
            k ← k+1, M[k] ← B
    else if in1 and not in2
    then    if A1.end ≤ A2.begin
        then    in1 ← false, i ← i+1
            if A2.begin ≤ A1.end
        then    in2 ← true, B.begin ← A2.begin
      else if not in1 and in2
    then    if A1.begin ≤ A2.end
            then    in1 ← true, B.begin ← A1.begin
        if A2.end ≤ A1.begin
        then    in2 ← false, j ← j+1
    else if not in1 and not in2
    then    if A1.begin ≤ A2.begin
        then    in1 ← true
        if A2.begin ≤ A1.begin
        then    in2 ← true
        if A1.begin = A2.begin
        then     B.begin ← A1.begin
M.length = k
```

Рис. 2. Алгоритм пересечения двух цепочек

```
                CHAIN_COMPLEMENT(M1, M2, M)
i ← 1
j ← 1
k ← 0
in1 ← false
in2 ← false
while true
do    if i ≤ M1.length
        then     A1 ← M1[i]
        else     A1.begin ← Q, A1.end ← Q

    if i ≤ M2.length
        then     A2 ← M2[i]
        else     A2.begin ← Q, A2.end ← Q

        if A1.begin = Q and A2.begin = Q
```

```
            then return

        if in1 and in2
        then    if A1.end ≤ A2.end
            then    in1 ← false, i ← i+1
            if A2.end ≤ A1.end
            then    in2 ← false, j ← j+1, B.begin ← A2.end
            else if in1 and not in2
        then    if A1.end ≤ A2.begin
            then    in1 ← false, i ← i+1, B.end ← A1.end
            if A2.begin ≤ A1.end
            then    in2 ← true, B.end ← A2.begin
                k ← k+1, M[k] ← B
        else if not in1 and in2
        then    if A1.begin ≤ A2.end
            then    in1 ← true
            if A2.end ≤ A1.begin
            then    in2 ← false, j ← j+1
            if A1.begin = A2.end
            then B.begin = A1.begin
        else if not in1 and not in2
        then    if A1.begin ≤ A2.begin
            then    in1 ← true, B.begin ← A1.begin
            if A2.begin ≤ A1.begin
            then    in2 ← true
    M.length = k
```

Рис. 3. Алгоритм дополнения двух цепочек

Заключение

Использование временных интервалов в программировании достаточно подробно изучено. Временные цепочки можно рассматривать как обобщение временных интервалов. В работе определены отношения между временными цепочками, их свойства, а также операции над ними. В работе показано, что в отличии от операций над интервалами операции над цепочками замкнуты, что снимает ряд ограничений и тем самым упрощает их использование. Разработаны эффективные алгоритмы для реализации операций над временными цепочками. Рассмотрены примеры их практического использования. Таким образом предложена эффективная структура для хранения и манипулирования темпоральными данными.

1. *James F.* Allen Maintaining knowledge about temporal intervals // Communications of ACM. – 1983. – Vol. 26. – N 11.
2. *James F.* Allen Towards a General Theory of Action and Time // Artificial Intelligence. – 1984. – N 23. – P. 123–154.
3. *Richard Snodgrass.* Developing Time-Oriented Database Applications in SQL, Morgan Kaufmann, 1999.

Получено 10.01.2013

Об авторах:

Кудим Кузьма Алексеевич,
младшний научный сотрудник,

Резниченко Валерий Анатольевич,
ведущий научный сотрудник,

Проскудина Галина Юрьевна,
научный сотрудник,

Овдей Ольга Михайловна,
младший научный сотрудник.

Место работы авторов:

Институт программных систем
НАН Украины,
03187, Киев-187,
Проспект Академика Глушкова, 40.
Тел. +38(044)526 6033

e-mail: reznich@isofts.kiev.ua
kuzmaka@mail.ru
gupros@isofts.kiev.ua
olga.ovdiy@gmail.com