Reasoning with Temporal Preferences over Data Streams

Marcos Roberto Ribeiro, Maria Camila Barioni, Sandra de Amo, Claudia Roncancio, Cyril Labbé

Proceedings of the Thirtieth International Florida Artificial Intelligence Research Society Conference

Reasoning with Temporal Preferences over Data Streams

Marcos Roberto Ribeiro
Instituto Federal de Minas Gerais, Bambuí, Brazil
Universidade Federal de Uberlândia, Uberlândia, Brazil
marcos.ribeiro@ifmg.edu.br

Maria Camila N. Barioni, Sandra de Amo
Universidade Federal de Uberlândia, Uberlândia, Brazil
camila.barioni@ufu.br, deamo@ufu.br

Claudia Roncancio, Cyril Labbé
Univ. Grenoble Alpes, LIG, F-38000 Grenoble, France
claudia.roncancio@imag.fr, cyril.labbe@imag.fr

Abstract

The growing emergence of new applications where the data changes rapidly has boosted the development of several researches related to data streams processing. Preference reasoning is an example of a useful task that can be used to monitor data streams for information that best fit the users wishes. In this paper, we revisited the formalism TPref in order to propose a new approach for processing data streams according to temporal conditional preferences. Our approach, named StreamPref, covers important issues not addressed yet in preference reasoning with temporal conditional preferences.

1 Introduction

The development of data streams monitoring technologies are primordial to support real-time decisions in several domain applications such as stock markets, sport players monitoring, among others. The issues related to the provision of efficient techniques to extract useful information from these possibly infinite data sets has challenged many research fields including artificial intelligence (AI) and databases (DB). Preference queries (Petit et al. 2012; de Amo and Bueno 2011) are interesting examples of tasks being tailored to couple with data streams.

A fundamental research issue related to preference queries is the development of techniques for preference reasoning. Such techniques allow the selection of data that best fit the user preferences. In the field of preference reasoning, which is the focus of this paper, the works based on the conditional preferences model have proved especially useful to this task (Boutilier et al. 2004; Cornelio 2015). In this model the user preferences over an attribute can be affected by values of another attribute. If we consider the *temporal* aspect, the preference specification becomes more expressive as an instant of time may influence the preferences in a future time moment (See Example 1).

Example 1. *Let us suppose a soccer coach who has access to an information system that provides real-time data about soccer matches. These data include the current player positioning, ball possession and the type of player moves. The coach wishes to select the best players according to the following preferences: **[P1]** If, in a given moment, the team has the ball possession and, immediately before this moment, the player was at defensive intermediary then I prefer middle-field place than defensive intermediary place, independent of move type; **[P2]** If, in a given moment, the team does not have the ball possession and, immediately before this moment, the player was at offensive intermediary and, always before this moment, the team had the ball, then I prefer middle-field place than offensive intermediary place; **[P3]** Lateral moves are better than forward moves.*

Although there are research works that employs the conditional preference model (de Amo and Bueno 2011; Petit et al. 2012) in the context of data streams, to the best of our knowledge, there is no proposal concerning data stream processing according to temporal conditional preferences. In this paper we present the formalism StreamPref that revisits the TPref formalism proposed in (de Amo and Giacometti 2007) with the aim of allowing the specification and reasoning with temporal conditional preferences over sequences extracted from data streams.

The main contributions of this paper are summarized as follows: ***(1)*** A new formalism for preference reasoning over data streams according to temporal conditional preferences; ***(2)*** A feasible consistency test for any set of rules in our language; ***(3)*** An incremental algorithm for the extraction of sequences from data streams; ***(4)*** Algorithms for dominance tests of sequences (i.e. comparison of sequences according to preferences); ***(5)*** A set of experiments that enforces the effectiveness of our approach.

2 The StreamPref Language

A remarkable feature of data streams environments is the sequential format of the data. Taking advantage of this characteristic, the StreamPref language was designed for reasoning over sequences of objects (Definition 1).

Definition 1 (Sequence)**.** *Let $X = \{A_1, ..., A_l\}$ be a set of attributes and $\mathcal{O}(X) = \textbf{\textit{Dom}}(A_1) \times ... \times \textbf{\textit{Dom}}(A_l)$ be the set of objects over X, where $\textbf{\textit{Dom}}(A_i)$ is the domain of A_i. A sequence $s = \langle o_1, ..., o_k \rangle$ over X is an ordered set of objects, such that $o_i \in \mathcal{O}(X)$ for all $i \in \{1, ..., k\}$.*

The length of a sequence $s = \langle o_1, ..., o_k \rangle$ is denoted by $|s| = k$. An object in a position i of s is denoted by $s[i]$ and the notation $s[i].A$ represents the attribute A in $s[i]$. We denote by $\textbf{Seq}(X)$ the set of all possible sequences over a set of attributes X. Given two sequences $s = \langle o_1, ..., o_k \rangle$ and $s' = \langle o'_1, ..., o'_{k'} \rangle$, the concatenation between s and s', denoted by $s + s'$, is $s'' = \langle o_1, ..., o_k, o'_1, ..., o'_{k'} \rangle$.

Example 2. *Consider the set of attributes* `Pl` *(player positioning),* `Tb` *(ball possession of team) and* `Mp` *(moving type). The values for* `Pl` *are defensive area (da), defensive intermediary (di), middle field (mf), offensive intermediary (oi) and offensive area (oa). We use 1 when the team has ball possession and 0 for otherwise. The moving types are forward (fw), rewind (rw) and lateral (la). The sequence* $s_1 = \langle (mf, 0, fw), (mf, 1, la), (di, 1, rw) \rangle$ *represents the player* 1 *moves displayed on Figure 1(a). The circled numbers means that team has the ball possession.*

The StreamPref language employs rules that follows the logical formalism proposed in the TPref (de Amo and Giacometti 2007) to express the user preferences. Basically, these rules have the format "if certain condition is true then I have this preference".

The Preference Specification. Our formalism uses past and present formulas based on Propositional Temporal Logic (PTL) (Prior 1967). In PTL, the propositions are variables $Q_1, ..., Q_n$. In the present work, a proposition is a predicate $(A\theta a)$, where $a \in \mathbf{Dom}(A)$ and $\theta \in \{<, \leq, =, \neq, \geq, >\}$. We use the notation $a \models Q(A)$ to state that the value a satisfies the proposition $Q(A)$. The notation $S_{Q(A)} = \{a \in \mathbf{Dom}(A) \mid a \models Q(A)\}$ denotes the set of values satisfying $Q(A)$. The propositions are used to construct basic formulas (Definition 2).

Definition 2 (Basic Formulas)**.** *Basic formulas are defined as follows: **(1) True** and **False** are basic formulas; **(2)** If F is a proposition then F is a basic formula; **(3)** If F and G are basic formulas then $\neg F$, $\neg G$, $(F \wedge G)$, $(F \vee G)$ and $(F$ **Since** $G)$ are basic formulas. The notation **Att**(F) denotes the attributes appearing in F.*

The notion of satisfaction of a formula F by a sequence $s = \langle o_1, ..., o_k \rangle$ at a position $i \in \{1, ..., k\}$, denoted by $(s, i) \models F$, is inductively defined as follows: ***(1)*** $(s, i) \models Q(A)$ if and only if $s[i].A \models Q(A)$; ***(2)*** $(s, i) \models (F \wedge G)$ if and only if $(s, i) \models F$ and $(s, i) \models G$; ***(3)*** $(s, i) \models (F \vee G)$ if and only if $(s, i) \models F$ or $(s, i) \models G$; ***(4)*** $(s, i) \models \neg F$ if and only if $(s, i) \not\models F$; ***(5)*** $(s, i) \models (F$ **Since** $G)$ if and only if there exists j where $1 \leq j < i$ and $(s, j) \models G$ and $(s, k) \models F$ for all k such that $j < k \leq i$. The **True** formula is always satisfied and the **False** formula is never satisfied. We also define the following derived formulas:

Prev $Q(A)$**:** Equivalent to **False Since** $Q(A)$, $(s, i) \models$ **Prev** $Q(A)$ if and only if $i > 1$ and $(s, i-1) \models Q(A)$;

SomePrev $Q(A)$**:** Equivalent to **True Since** $Q(A)$, $(s, i) \models$ **SomePrev** $Q(A)$ if and only if there exists j such that $1 \leq j < i$ and $(s, j) \models Q(A)$;

AllPrev $Q(A)$**:** Equivalent to $\neg$**SomePrev**$\neg Q(A)$, $(s, i) \models$ **AllPrev** $Q(A)$ if and only if $(s, j) \models Q(A)$ for all $1 \leq j < i$;

First: Equivalent to $\neg$**Prev True**, $(s, i) \models$ **First** if and only $i = 1$.

The propositions and the derived formulas are called by *atomic formulas*. These atomic formulas are used to compose the *temporal conditions* (Definition 3). So, the temporal conditions are employed in the construction of *tcp-rules* and *tcp-theories* (Definition 4).

Definition 3 (Temporal Conditions)**.** *A temporal condition is an empty formula or a formula $F = (F_1 \wedge ... \wedge F_p)$, where the terms $F_1, ..., F_p$ are atomic formulas. The notation F^- is the conjunction of all derived formulas in F. The notation F^0 is the conjunction of all propositions in F and not in F^-.*

Definition 4 (TCP-Rules and TCP-Theories)**.** *Let X be a set of attributes. A temporal conditional preference rule, or tcp-rule, is an expression in the format $\varphi : C_\varphi \rightarrow Q^+_\varphi(A_\varphi) \succ Q^-_\varphi(A_\varphi)[W_\varphi]$, where: **(1)** $A_\varphi \in X$ is the preference attribute and $W_\varphi \subset X$ is the set of indifferent attributes such that $A_\varphi \notin W_\varphi$; **(2)** $Q^+_\varphi(A_\varphi)$ and $Q^-_\varphi(A_\varphi)$ are propositions representing the preferred values and non preferred values for A_φ, respectively, such that $S_{Q^+_\varphi(A_\varphi)} \cap S_{Q^-_\varphi(A_\varphi)} = \emptyset$; **(3)** C_φ is a temporal condition such that **Att**$(C^0_\varphi) \cap (\{A_\varphi\} \cup W_\varphi) = \emptyset$. A temporal conditional preference theory, or tcp-theory, is a finite set of tcp-rules.*

Example 3. *Consider the preferences of Example 1 and the attributes of Example 2. We can express **[P1]**, **[P2]** and **[P3]** by the tcp-theory $\Phi = \{\varphi_1, \varphi_2, \varphi_3\}$, where:*

φ_1 : ***Prev***$(\texttt{Pl} = di) \wedge (\texttt{Tb} = 1) \rightarrow (\texttt{Pl} = mf) \succ (\texttt{Pl} = di)[\texttt{Mp}]$;

φ_2 : ***AllPrev***$(\texttt{Tb} = 1) \wedge (\texttt{Tb} = 0) \wedge$ ***Prev***$(\texttt{Pl} = oi) \rightarrow (\texttt{Pl} = mf) \succ (\texttt{Pl} = oi)$;

φ_3 : $\rightarrow (Mp = la) \succ (Mp = fw)$.

Let Φ be a tcp-theory. Let $\varphi \in \Phi$ be a tcp-rule. Let s and s' be two sequences. We say that s is preferred to s' according to φ, denoted by $s \succ_\varphi s'$, if there exists i such that: ***(1)*** $s[j] = s'[j]$ for all $j \in \{1, ..., i-1\}$; ***(2)*** $(s, i) \models C_\varphi$ and $(s', i) \models C_\varphi$; ***(3)*** $s[i].A_\varphi \models Q^+_\varphi(A_\varphi)$ and $s'[i].A_\varphi \models Q^-_\varphi(A_\varphi)$ and ***(4)*** $s[i].A' = s'[i].A'$ for all $A' \notin (\{A_\varphi\} \cup W_\varphi)$.

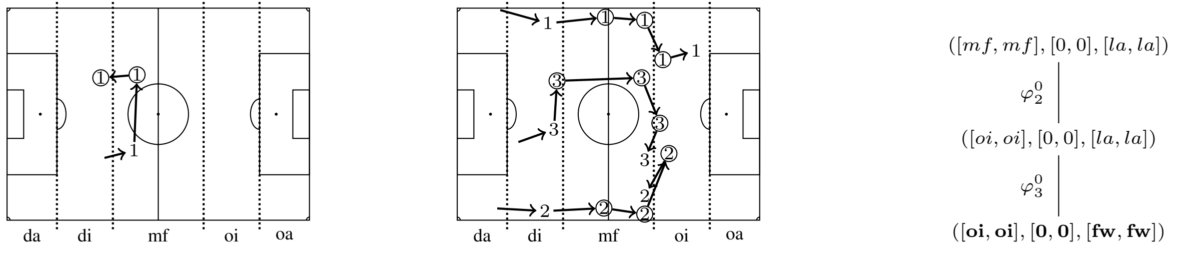
The notation $\succ_\Phi$ represents the transitive closure of $\bigcup_{\varphi \in \Phi} \succ_\varphi$. Let two sequences $s, s' \in \mathbf{Seq}(X)$, the notation $s \succ_\Phi s'$ means that s is preferred to s' according to Φ. When two sequences cannot be compared, they are incomparable.

Example 4. *Let us consider the tcp-theory $\Phi = \{\varphi_1, \varphi_2, \varphi_3\}$ of Example 3 and the sequences $s_a = \langle (di, 1, fw), (mf, 1, fw), (oi, 1, fw), (oi, 0, la) \rangle$, $s_b = \langle (di, 1, fw), (mf, 1, fw), (oi, 1, fw), (mf, 0, la) \rangle$, $s_c = \langle (di, 1, fw), (di, 1, la), (mf, 1, fw), (mf, 0, rw) \rangle$.*

We have that $s_a \succ_{\varphi_1} s_c$ and $s_b \succ_{\varphi_1} s_c$ (at position 2) and $s_b \succ_{\varphi_2} s_a$ at (position 4). Therefore, $s_b \succ_\Phi s_a \succ_\Phi s_c$.

Consistency Test. Consistency issues have to be carefully analyzed when dealing with the order induced by rules, since the irreflexive property is desirable during reasoning tasks. The transitive closure of $\bigcup_{\varphi \in \Phi} \succ_\varphi$ may generate an inconsistent preference order, that is, it can be inferred that "a sequence is preferred to itself" (See Example 5).

Example 5. *Suppose $\Phi' = \{\varphi'_1, \varphi'_2, \varphi'_3\}$, where:*



da
di
mf
oi
oa
1
2
3
$([mf, mf], [0, 0], [la, la])$
φ_2^0
$([oi, oi], [0, 0], [la, la])$
φ_3^0
$([\mathbf{oi}, \mathbf{oi}], [\mathbf{0}, \mathbf{0}], [\mathbf{fw}, \mathbf{fw}])$
(a)
(b)
(c)

Figure 1: Players moves and search tree: (a) Moves of s_1; (b) Moves three players; (c) Search tree of *SearchDom* algorithm.

φ'_1 : ***Prev***$(\texttt{Pl} = di) \to (\texttt{Pl} = mf) \succ (\texttt{Pl} = di)$;

φ'_2 : $(\texttt{Tb} = 1) \to (\texttt{Pl} = oi) \succ (\texttt{Pl} = mf)$;

φ'_3 : ***Prev***$(\texttt{Tb} = 0) \to (\texttt{Pl} = di) \succ (\texttt{Pl} = oi)$.

Also consider the sequences: $s_a = \langle(di, 0, fw), (mf, 1, la)\rangle$; $s_b = \langle(di, 0, fw), (di, 1, la)\rangle$; $s_c = \langle(di, 0, fw), (oi, 1, la)\rangle$. According to tcp-rules of Φ', we have $s_a \succ_{\varphi'_1} s_b$, $s_b \succ_{\varphi'_3} s_c$ and $s_c \succ_{\varphi'_2} s_a$, then $s_a \succ_{\Phi'} s_a$. Thus, the induced order $\succ_{\Phi'}$ is reflexive and Φ' is inconsistent.

Table 1: Temporal Compatibility between Atomic Formulas

	$Q(A)$	**First**	**Prev** $Q(A)$	**SomePrev** $Q(A)$	**AllPrev** $Q(A)$
$Q'(A')$	✓	✓	✓	✓	✓
First	✓	✓	×	×	×
Prev $Q'(A')$	✓	×	*	✓	*
SomePrev $Q'(A')$	✓	×	✓	✓	*
AllPrev $Q'(A')$	✓	×	*	*	*

Our consistency test takes into consideration the *temporal compatibility*. The temporal compatibility between atomic formulas is given by Table 1, where $Q(A)$ and $Q'(A')$ are propositions. The symbol * means that the atomic formulas are temporally compatible if one of the following conditions are satisfied: ***(1)*** $A \neq A'$; ***(2)*** $A = A'$ and $S_{Q(A)} \cap S_{Q'(A')} \neq \emptyset$. The temporal compatibility between tcp-rules is established by Definition 5.

Definition 5 (Temporal Compatibility)**.** *Let φ and φ' be two tcp-rules such that $C_\varphi = (F_1 \wedge ... \wedge F_p)$ and $C_{\varphi'} = (F'_1 \wedge ... \wedge F'_q)$. These tcp-rules are temporally compatible if F_i and F'_j are temporally compatible for all $i \in \{1, ..., p\}$ and for all $j \in \{1, ..., q\}$. A tcp-rule with empty condition is temporally compatible with any tcp-rule.*

Let Φ be a tcp-theory and $\varphi \in \Phi$ be a tcp-rule. The notation Φ_φ denotes the set of tcp-rules in Φ temporally compatible with φ. The consistency test of the subset Φ_φ can be done using just the non temporal components of rules. Given a tcp-rule φ, the notation φ^0 denotes a rule obtained from φ replacing C_φ by C_φ^0. When we perform such replacement over the rules of Φ_φ, the resulting set has the same format of the cp-theories used by CPrefSQL (Ribeiro, Pereira, and Dias 2016). For instance, for tcp-rule φ_1 of Example 3, we have $\varphi_1^0 : (\texttt{Tb} = 1) \to (\texttt{Pl} = mf) \succ (\texttt{Pl} = di)[\texttt{Mp}]$.

The cp-theory extracted from a set of tcp-rules Φ_φ is defined as $\Gamma(\Phi_\varphi) = \{\varphi'^0 \mid \varphi' \in \Phi_\varphi\}$. As there is a well defined consistency test of cp-theories, we can check the consistency of a tcp-theory Φ running the consistency test of cp-theories over $\Gamma(\Phi_\varphi)$ for every $\varphi \in \Phi$. The Lemma 1 and the Theorem 2 give us the sufficient conditions to verify if a tcp-theory is consistent. More details about the consistency test of cp-theories are found in (Wilson 2004).

Lemma 1. *Let φ and φ' be two tcp-rules. If there exists a sequence s containing a position i such that $(s, i) \models C_\varphi^-$ and $(s, i) \models C_{\varphi'}^-$, then φ and φ' are temporally compatible.*

Proof. Le us suppose by absurd that there exists a position i in s such that $(s, i) \models C_\varphi^-$ and $(s, i) \models C_{\varphi'}^-$, but φ and φ' are not temporally compatible. If φ and φ' are not temporally compatible than there exist at least one atomic formula F in C_φ^- and at least one atomic formula G in $C_{\varphi'}^-$ such that F and G are not temporally compatible. For this to happen, there are three cases: ***(1)*** The formulas are $F =$ **First** and $G =$ **Prev** $Q(A)$, $G =$ **SomePrev** $Q(A)$ or $G =$ **AllPrev** $Q(A)$; ***(2)*** The formulas are $F =$ **Prev** $Q(A)$ and $G =$ **Prev** $Q'(A)$ or $G =$ **AllPrev** $Q'(A)$ such that $S_{Q(A)} \cap S_{Q'(A)} = \emptyset$; ***(3)*** The formulas are $F =$ **AllPrev** $Q(A)$ and $G =$ **Prev** $Q'(A)$, $G =$ **SomePrev** $Q'(A)$ or $G =$ **AllPrev** $Q'(A)$ such that $S_{Q(A)} \cap S_{Q'(A)} = \emptyset$.
(Case 1) Let us suppose that $F =$ **First** and $G =$ **Prev** $Q(A)$, $G =$ **SomePrev** $Q(A)$ or $G =$ **AllPrev** $Q(A)$. If $i = 1$ then $(s, i) \models F$, but $(s, i) \not\models G$. On the other hand, if $i > 1$ then $(s, i) \models G$, but $(s, i) \not\models F$. In both situations there is an absurd. *(Case 2)* Let us suppose that $S_{Q(A)} \cap S_{Q'(A)} = \emptyset$, $F =$ **Prev** $Q(A)$ and $G =$ **Prev** $Q'(A)$ or $G =$ **AllPrev** $Q'(A)$. Thus, there is no position i in s such that $(s, i) \models F$ and $(s, i) \models G$, this is an absurd. *(Case 3)* Let us suppose that $S_{Q(A)} \cap S_{Q'(A)} = \emptyset$, $F =$ **AllPrev** $Q(A)$ and $G =$ **Prev** $Q'(A)$, $G =$ **SomePrev** $Q'(A)$ or $G =$ **AllPrev** $Q'(A)$. There is no position i such that $(s, i) \models F$ and $(s, i) \models G$, this is an absurd. □

Theorem 2. *Let Φ be a tcp-theory. If $\Gamma(\Phi_\varphi)$ is consistent for all $\varphi \in \Phi$, then Φ is consistent.*

Proof. Let us suppose by absurd that $\Gamma(\Phi_\varphi)$ is consistent for all $\varphi \in \Phi$, but Φ is not consistent. If Φ is not consistent then there exists rules $\varphi_1, ..., \varphi_m \in \Phi$ and sequences $s_1, ..., s_m \in$ **Seq**(X) such that $C_{s_1} \succ_{\varphi_1} ... \succ_{\varphi_{m-1}} C_{s_m} \succ_{\varphi_m} s_1$. By definition, there exists a common position i in the sequences $s_1, ..., s_m$ where the comparisons were made. In this way, $s_1[1, i-1] = ... = s_m[1, i-1]$ and $(s_1, i) \models \varphi_1, ..., (s_1, i) \models \varphi_m$. By Lemma 1, the rules $\varphi_1, ..., \varphi_m$ are temporally compatible. Therefore these rules are in the same set of compatible rules. Thus, the order induced by this set is reflexive. However, the cp-theories $\Gamma_\varphi(\Phi)$ resulting from compatible sets Φ_φ are consistent and that is an absurd. $\square$

3 Algorithms

Sequence Extraction. The sequence extraction retrieves identified sequences (Definition 6) over a data stream S according to a set of identifier attributes Z. At every instant, each object must have unique values over identifier attributes Z in order to keep a relation one-to-one between objects and resulting sequences. Note that identifier attributes are not part of sequence objects.

Definition 6 (Identified Sequences)**.** *Let X be a set of attributes. Let Y and Z be two disjoint sets such that $Y \cup Z = X$. An identified sequence $s_z = \langle o_1, ..., o_k \rangle$ is a sequence where $o_i \in \mathcal{O}(Y)$ for all $i \in \{1, ..., k\}$ and $z \in \mathcal{O}(Z)$.*

The Algorithm 1 (*ExtractSeq*) uses a hash-table $H^\#$ to perform the sequences extraction incrementally. Moreover, this algorithm receives a temporal range parameter k in seconds. Thus, the sequence extraction is performed over a *time-base sliding window* (Petit et al. 2012) with a range of k seconds and a slide of one second. Our algorithm can be adapted to work with different models of sliding windows.

Algorithm 1: *ExtractSeq*(S, Z, k)

```
1 foreach $o \in S[\tau]$ do
2     $z \leftarrow GetId(o, Z)$;
3     if $z \notin H^\#.Keys()$ then $s_z \leftarrow \langle o/Z \rangle_z$ ;
4     else $s_z \leftarrow H^\#.Get(z) + \langle o/Z \rangle$ ;
5     $H^\#.Put(z, s_z)$;
6 foreach $s_z \in H^\#$ do RemoveExpired$(s_z, \tau - k)$;
7 return sequences in $H^\#$;
```

The hash-table $H^\# : (z \mapsto s_z)$ associates each identifier z to the identified sequence s_z. The operation o/Z removes the identifier attributes in Z from object o. The set of objects in stream S at the current instant is denoted by $S[\tau]$. Before the first iteration, $H^\#$ is an empty hash-table. Thenceforward, at every instant, the *ExtractSeq* algorithm updates $H^\#$ and returns the built sequences until the current instant τ. The instruction *RemoveExpired*$(s_z, \tau - k)$ remove the expired objects from s_z. An object expires if its timestamp is less or equal to $\tau - k$.

Example 6. *Let us consider the moves of the players* 1, 2 *and* 3 *displayed on Figure 1(b). The objects have the same attributes of Example 2 and the attribute* `Id` *(player identification). Let us consider that a coach wants to analyze the sequences of the last three plays. These sequences can be extracted by ExtractSeq algorithm considering a range of 3 seconds and the attribute* `Id` *the identifier. The extracted sequences, instant by instant, are presented as follows:*

Instant 1: *The sequences $s_1 = \langle (di, 0, fw) \rangle$, $s_2 = \langle (di, 0, fw) \rangle$ and $s_3 = \langle (di, 0, fw) \rangle$ are created;*

Instant 2: *The incoming objects are attached into the end of existing sequences. The result is: $s_1 = \langle (di, 0, fw), (mf, 1, fw) \rangle$, $s_2 = \langle (di, 0, fw), (mf, 1, fw) \rangle$ and $s_3 = \langle (di, 0, fw), (di, 1, la) \rangle$;*

Instant 3: *Again, the incoming objects are attached and we have: $s_1 = \langle (di, 0, fw), (mf, 1, fw), (mf, 1, fw) \rangle$, $s_2 = \langle (di, 0, fw), (mf, 1, fw), (mf, 1, fw) \rangle$ and $s_3 = \langle (di, 0, fw), (di, 1, la), (mf, 1, fw) \rangle$;*

Instant 4: *After processing the incoming objects, the sequences has length greater than 3, then the algorithm drop their first positions. The result is: $s_1 = \langle (mf, 1, fw), (mf, 1, fw), (oi, 1, la) \rangle$, $s_2 = \langle (mf, 1, fw), (mf, 1, fw), (oi, 1, la) \rangle$, $s_3 = \langle (di, 1, la), (mf, 1, fw), (oi, 1, la) \rangle$;*

Instant 5: *The process is analogous to instant 4. So, we have: $s_1 = \langle (mf, 1, fw), (oi, 1, la), (oi, 0, fw) \rangle$, $s_2 = \langle (mf, 1, fw), (oi, 1, la), (mf, 0, la) \rangle$ and $s_3 = \langle (mf, 1, fw), (oi, 1, la), (mf, 0, la) \rangle$.*

The first loop (lines 1-5) of *ExtractSeq* algorithm performs the insertion of the objects from S into their respective sequences. This task has cost of $O(nl \times |H^\#.Get(z)|)$ where n is the objects number in $S[\tau]$ and l is the attributes number of S. As we have a unique object per identifier z at every instant, then $|H^\#.Get(z)| = 1$. Thus, the cost of the objects insertion is $O(nl)$. The cost of the second loop (line 6) is $O(nk)$ where k is the temporal range. Thus, in the worst case, the complexity of the *ExtractSeq* algorithm is $O(nl + nk)$. If we assume a constant factor to attributes number and temporal range, the complexity of *ExtractSeq* is $O(n)$.

Dominant Sequences. The Algorithm 2 (*DomSeq*) receives a set of sequences T, a tcp-theory Φ and returns the *dominant sequences* in T according to Φ. A sequence $s \in T$ is dominant if there is no $s' \in T$ such that $s' \succ_\Phi s$. First, the algorithm copies T to T'. Next, for every pair of sequences $s, s' \in T'$, the algorithm performs the dominance tests $s \succ_\Phi s'$ and $s' \succ_\Phi s$. At the end, just the dominant sequences remain in T'.

Algorithm 2: *DomSeq*(T, Φ)

```
1 $T' \leftarrow T$;
2 foreach $s, s' \in T'$ do
3     if Dominates$(\Phi, s, s')$ then $T' \leftarrow T' - \{s'\}$ ;
4     else if Dominates$(\Phi, s', s)$ then $T' \leftarrow T' - \{s\}$ ;
5 return $T'$;
```

Example 7. *Let us consider the extracted sequences of the Example 6 and the tcp-theory of the Example 3. The execution of the DomSeq algorithm is shown as follows:*

Instant 1: *All sequences are incomparable, so the algorithm returns* $T' = \{s_1, s_2, s_3\}$*;*

Instants 2 and 3: *We have* $s_1 \succ_\Phi s_3$ *and* $s_2 \succ_\Phi s_3$*, then* $T' = \{s_1, s_2\}$*;*

Instant 4: *Again, all sequences are incomparable, and* $T' = \{s_1, s_2, s_3\}$*;*

Instant 5: *We have* $s_2 \succ_\Phi s_1$ *and* $s_3 \succ_\Phi s_1$*, then* $T' = \{s_2, s_3\}$.

The Algorithm 3 (*Dominates*) performs the dominance test $s \succ_\Phi s'$. First, the algorithm finds the first position i where the sequences are different. Next, the algorithm creates a cp-theory Γ containing the cp-rules correspondent to tcp-rules valid at position i. Thus, the *SearchDom* routine (Algorithm 4) is called to verify if the object $s[i]$ is better than the object $s'[i]$.

Algorithm 3: *Dominates*(Φ, s, s')

```
1  $j \leftarrow \min\{|s|, |s'|\}$;
2  foreach $i \in \{1, ..., j\}$ do
3      if $s[i] \neq s'[i]$ then
4          $\Gamma \leftarrow \{\}$;
5          foreach $\varphi \in \Phi$ do
6              if $((s, i) \models C_\varphi)$ and $((s', i) \models C_\varphi)$ then
7                  $\Gamma \leftarrow \Gamma \cup \{\varphi^0\}$
8          return SearchDom$(\Gamma, s[i], s'[i])$;
9  return False;
```

The *SearchDom* routine uses a depth-first-search strategy and looks for a chain of rules from object o^+ to object o^-. At every iteration, if the goal was not reached, we get the next objects using *Change* routine (Algorithm 5) and push them into stack *next*. The *IntervalObject*(o) instruction converts the values of o into intervals. For example, if $o.A_i = 3$ then, after transformation, $o.A_i = [3, 3]$. Thus, the goal test *IsGoal*(o^-, o) checks if $o^-.A_i$ intersects $o.A_i$, for every attribute A_i.

Algorithm 4: *SearchDom*(Γ, o^+, o^-)

```
1  visited $\leftarrow \{\}$;
2  next $\leftarrow$ Stack(IntervalObject$(o^+)$);
3  while $|next| > 0$ do
4      $o \leftarrow$ next.pop();
5      visited $\leftarrow$ visited $\cup \{o\}$;
6      if IsGoal$(o^-, o)$ then return True;
7      else
8          foreach $\varphi \in \Gamma$ do
9              $o'' \leftarrow$ Change$(o, \varphi)$;
10             if $(o'' \neq$ Null$)$ and $(o'' \notin$ visited$)$ then
11                 next.push$(o'')$;
12 return False
```

The *Change* routine gets a new object from o by applying a transformation according to the rule φ. This transformation is possible when o satisfies the rule condition ($o \models C_\varphi$) and o has a preferred value. If an attribute $o.A$ is an interval, then $o.A \models Q(A)$ when $o.A \cap S_{Q(A)} \neq \emptyset$. The transformation swaps the value of the preference attribute (A_φ) by the non preferred values of rule φ. In addition, all indifferent attributes receives the interval $[-\infty, +\infty]$ since these attributes can have any value during the comparison.

Algorithm 5: *Change*(o, φ)

```
1  if $o \not\models C_\varphi$ or $o \not\models Q^+_\varphi(A_\varphi)$ then return Null ;
2  $o.A_\varphi \leftarrow$ Interval$(Q^-_\varphi(A_\varphi))$;
3  foreach $A_i \in W_\varphi$ do $o.A_i \leftarrow [-\infty, +\infty]$;
4  return $o$;
```

Example 8. *Let us consider the sequences* s_1 *and* s_2 *at instant 5 of Example 6. The dominance test* $s_2 \succ_\Phi s_1$ *is performed by the Dominates algorithm as follows:*

- *In the third iteration of the outer loop, the algorithm finds the position to be compared* ($i = 3$)*;*
- *The inner loop scans* Φ *looking for tcp-rules whose conditions satisfies* s_1 *and* s_2 *at position* 3*;*
- *The tcp-rules with satisfied conditions are* φ_2 *and* φ_3*. The resulting cp-theory* Γ *is composed by the rules:*
  $\varphi_2^0 : (\texttt{Tb} = 0) \rightarrow (\texttt{Pl} = mf) \succ (\texttt{Pl} = oi)$ *and*
  $\varphi_3^0 : \rightarrow (Mp = la) \succ (Mp = fw)$*;*
- *Thus, the algorithm calls the SearchDom*(Γ, o^+, o^-) *routine, where* $o^+ = (mf, 0, la)$ *and* $o^- = (oi, 0, fw)$*;*
- *Figure 1(c) shows the search tree of SearchDom routine. As the search tree reaches the goal (in bold), the algorithm returns **True** for the dominance test.*

The complexity analysis of *DomSeq* algorithm considers that preference rules are specified over a set of attributes $X = \{A_1, ..., A_l\}$. The *Change* routine has the cost $O(|\mathbf{Att}(C_\varphi)| + |W_\varphi|)$. In the worst case, $|\mathbf{Att}(C_\varphi)| = |W_\varphi| = l$, where $l = |X|$ is the attributes number. The search tree of the *SearchDom* routine has height and node degree equal to m in the worst case, where m is the rules number. Thus, the complexity of the *SearchDom* routine is $O(lm^m)$ because we must check every combination of rules.

The cost of the *Dominates* routine is $O(km + klm^m) = O(klm^m)$, where k is the maximum sequence length. The complexity of the *DomSeq* algorithm is $O(n^2klm^m)$, where $n = |T|$ is the sequences number. If we assume a constant factor to attributes number and sequences length, the complexity of *DomSeq* is $O(n^2m^m)$.

4 Related Work

This paper focuses on reasoning with qualitative conditional preferences. The CP-Nets proposed by (Boutilier et al. 2004) were a notable work on this topic. The CP-Nets are a graphical preference model for representing conditional preferences under *ceteris paribus* semantics. The formalism introduced by (Wilson 2004) uses sets of rules that allow the specification of preferences more generic than CP-Nets. This approach was also employed by the language CPref-SQL for the evaluation of database queries containing conditional preferences (Ribeiro, Pereira, and Dias 2016).

The works described in (de Amo and Bueno 2011; Petit et al. 2012) use the preference model of CPrefSQL to design incremental algorithms for the evaluation of preference queries over data streams. Such algorithms construct a preference hierarchy over data elements and update this hierarchy when new elements arrive or old elements expire.

As mentioned early herein, our approach is based on the TPref framework (de Amo and Giacometti 2007). The TPref and StreamPref formalism uses the PTL, but they have significant differences. First, the propositions of TPref formalism are simple equalities. In contrast, our propositions are more expressive and support predicates like $(A \leq 10)$. The basic formulas of TPref include the temporal predicate **Until**. This predicate allows the composition of future formulas. We restricted the StreamPref formalism to the present and the past formulas because they are more suitable for data stream scenarios. Another improvement in StreamPref is the support to indifferent attributes that allows to break the *ceteris paribus* semantic for some attributes. Our work also proposes algorithms for dominance test, while the work presented in (de Amo and Giacometti 2007) just established the theoretical foundations of this task.

The most important difference between TPref and StreamPref is the conditions format. The temporal conditions of TPref are arbitrary PTL formulas, but this imposes great difficult for consistency test. The work described in (de Amo and Giacometti 2007) shows there is no feasible consistency test for tcp-theories containing arbitrary formulas in the temporal conditions. On the other hand, the temporal condition format proposed herein makes it feasible to perform the consistency test for any tcp-theory in our language.

5 Experimental Results

In order to demonstrate the effectiveness of our approach, we implemented the algorithms *ExtractSeq* and *DomSeq* to conduct experiments under diverse temporal ranges. Our experiments employed a real dataset of the 2014 soccer world cup containing 167,801 objects of 64 matches[1].

The cost to compute the dominant sequences is highly correlated to temporal range since greater temporal ranges produces longer sequences that require more time to be compared. Therefore, to analyze the behavior of the *ExtractSeq* and the *DomSeq* algorithms we considered the following values for the temporal range: 5, 10, 20, 40, 80 and 160. Considering the objects number, the temporal ranges used is equivalent to the experiments with sliding windows in (Petit et al. 2012). Our experiments used the same tcp-theory of Example 3.

For each temporal range, the algorithms were executed five times to obtain the average runtime per match. The runtime per match is the total runtime of all matches divided by the number of matches. All experiments were carried out on a machine with a 3.2 GHz twelve-core processor and 32 GB of main memory, running Linux. Figure 2 shows the runtime of the algorithms.

As expected, the *ExtractSeq* algorithm has the smallest runtime since this algorithm just appends the stream objects

[1]http://data.huffingtonpost.com/2014/world-cup

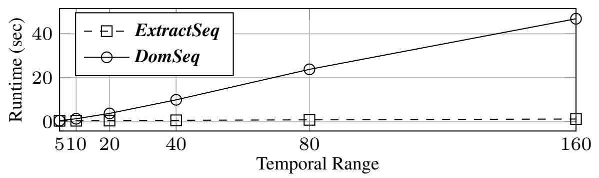


Figure 2: Runtime for *ExtractSeq* and *DomSeq* algorithms

to the correspondent sequences. On the other hand, the *DomSeq* algorithm has the greatest runtime due to the expensive cost of the dominance test.

It is worth mentioning that a soccer match has a duration of at least 5,400 seconds. Moreover, our algorithms had a runtime under 50 seconds considering the 160 seconds range. Thus, according to the experiments results, our algorithms had acceptable efficiency for the analyzed scenario.

6 Conclusion

In this paper we presented the StreamPref language for reasoning with temporal conditional preferences over data streams. We proposed a feasible consistency test in order to check if a given set of rules does not impose a reflexive order over the sequences.

We also designed the algorithm *ExtractSeq* to extract sequences from a data stream and the algorithm *DomSeq* to perform dominance test between sequences. The algorithm *ExtractSeq* is incremental, but the algorithm *DomSeq* is not. Currently, we are working on an incremental version for the *DomSeq* algorithm in order to optimize the sequence comparisons in data streams scenarios.

Acknowledgments. The authors thanks to the Research Agencies CNPq, CAPES and FAPEMIG for supporting this work.

References

Boutilier, C.; Brafman, R. I.; Domshlak, C.; Hoos, H. H.; and Poole, D. 2004. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *JAIR* 21:135–191.

Cornelio, C. 2015. Models for conditional preferences as extensions of cp-nets. In *IJCAI*, 4355–4356.

de Amo, S., and Bueno, M. L. P. 2011. Continuous processing of conditional preference queries. In *SBBD*.

de Amo, S., and Giacometti, A. 2007. Temporal conditional preferences over sequences of objects. In *ICTAI*, 246–253.

Petit, L.; de Amo, S.; Roncancio, C.; and Labbé, C. 2012. Top-k context-aware queries on streams. In *DEXA*, 397–411.

Prior, A. N. 1967. *Past, Present and Future*. Oxford, New York, USA: Oxford University Press.

Ribeiro, M. R.; Pereira, F. S. F.; and Dias, V. V. S. 2016. Efficient algorithms for processing preference queries. In *ACM SAC*, 972–979.

Wilson, N. 2004. Extending cp-nets with stronger conditional preference statements. In *AAAI*, 735–741.