Compact Answers to Temporal Path Queries

Muhammad Adnan, Diego Calvanese, Julien Corman, Anton Dignös[0000−0002−7621−967X], Werner Nutt[0000−0002−9347−1885], and Ognjen Savković[0000−0002−9141−3008]

Free University of Bozen-Bolzano, Italy

Abstract. We study path-based graph queries that, in addition to navigation through edges, also perform navigation through time. This allows asking questions about the dynamics of networks, like traffic movement, cause-effect relationships, or the spread of a disease. In this setting, a graph consists of triples annotated with validity intervals, and a query produces pairs of nodes where each pair is associated with a binary relation over time. For instance, such a pair could be two airports, and the temporal relation could map potential departure times to possible arrival times. An open question is how to represent such a relation in a compact form, and maintain this property during query evaluation. To address this, we investigate four compact representations of answers to a such queries. We discuss their respective advantages and drawbacks, in terms of conciseness, uniqueness, and computational cost. Notably, the most refined encoding guarantees that query answers over dense time can be finitely represented.

Keywords: graph databases · temporal databases · regular path queries

1 Introduction

Temporal databases [5] have traditionally focused on computing relations where each tuple is annotated with a *set* of time points (or a set of intervals) for validity. In comparison, little attention has been paid to the case where each tuple is annotated with a *relation* over time points, which indicates how different events are related temporally. This is the focus of this paper, more precisely *binary* relations over time points.

One application is linking potential departure times of a road trip to their corresponding arrival times, e.g. considering uncertainties like traffic. Such a relation maps each departure time to a range of possible arrival times (or equivalently each arrival time to possible departures). Another example is modelling cause-effect scenarios with uncertain delays. For instance, one may compute the relation that associates (in time) the potential malfunction a component in an airplane to the subsequent malfunction of another component. A third application of binary temporal relations, which we will use as a running example (inspired by [2]), is modeling the spread of phenomena such as messages or diseases. For instance, in epidemiology, the latency and infectiousness periods of a

virus induce a temporal relation that maps potential infection times to possible subsequent transmissions, which allows modeling disease propagation within a population.

These examples suggest at least three elementary operations that one may want to perform over binary temporal relations: *(i) filtering* such a relation (based on some knowledge about events that took place), *(ii) combining* two such relations, and *(iii) composing* them (e.g. to model transitive cause-effect or spread). Or in database terms, selection, union and join respectively. These operations can be seen as a natural generalization of their counterparts in classical temporal databases (if we abstract away from non-temporal data). Indeed, in these systems, the set of time points associated to a fact (for validity) can be viewed as a unary relation over time, which we generalize here to a binary one.

However, prior to performing such operations, a fundamental question is how to *represent* their input and output. This is the main problem addressed in this paper. Precisely, how to represent a binary temporal relation in a compact way (or even finitely over dense time), and ensure that compactness (resp. finiteness) is preserved by selection, union and/or join. To our knowledge, this has not been addressed in the literature, and as we will see, the answer to this (apparently simple) question is not trivial. In particular, such a relation cannot be represented as a set of intervals, as opposed to the sets of time points (a.k.a. unary relations) that temporal databases traditionally focus on.

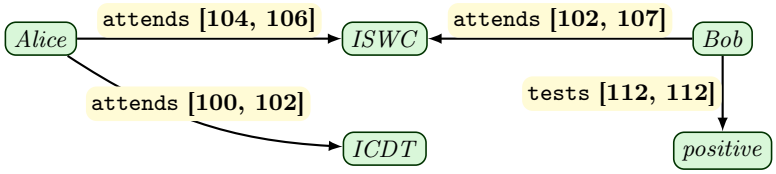

Fig. 1: A temporal graph

Concretely, we focus on the query language recently introduced in [2], which computes and performs operations on binary temporal relations, along with a graph data model used to filter such relations. To our knowledge, this is the first query language that manipulates such relations. For presentation purposes, we simplify this language and data model to a fragment that is essentially Regular Path Queries (RPQs)[1] extended with a temporal navigation operator, and evaluated over a time-labeled graphs. RPQs are used to navigate through the graph, whereas the temporal operator enables navigation in time, by a certain *range*.

An RPQ q is a regular expression, and a pair $\langle n_1, n_2 \rangle$ of nodes is an answer to q over a graph G if there exists a path from n_1 to n_2 in G whose concatenated labels match this regular expression. The language of [2], called *Temporal RPQs*

[1] RPQs are a central building block of navigational graph query languages such as Cypher [15] and SPARQL [17].

(TRPQs), extends RPQs with a temporal operator that allows navigation from one node at a certain time to the same node in past or future moments. Accordingly, an answer is a pair $\langle\langle n_1, t_1\rangle, \langle n_2, t_2\rangle\rangle$, where n_1 and n_2 are each associated with a time point, which mark the starting and arrival time of the temporal navigation respectively.

More precisely, this temporal operator, which we denote as $\mathbf{T}_\delta$ here, allows navigation in time within the range specified as the interval δ. For instance, the latency and infectiousness periods of a virus may imply that a person can only become infectious at least three days after the infection time t, and remain infectious at most five days after t. If $\mathcal{T}$ is our temporal domain, this induces a binary temporal relation

$$R = \{(t, t+d) \mid t \in \mathcal{T} \text{ and } d \in [3,5]\}.$$

Accordingly, if $\mathcal{N}_G$ is the set of nodes in the queried graph G, then the query $\mathbf{T}_{[3,5]}$ outputs all pairs $\langle\langle n, t_1\rangle, \langle n, t_2\rangle\rangle$ such that $n \in \mathcal{N}_G$ and $(t_1, t_2) \in R$. This output can be joined and filtered based on events registered in G, effectively restricting the initial temporal relation R, and associating each returned time point with a meaningful node. For instance, the graph of Figure 1 represents data about attendance at a conference, where each fact is a labelled edge, annotated with an interval for validity. We can extend our query as follows

$$q_1 = \mathbf{T}_{[3,5]}/\texttt{attends}/\texttt{attends}^-.$$

This query outputs all pairs $\langle\langle n_1, t_1\rangle, \langle n_2, t_2\rangle\rangle$ such that n_1 and n_2 attended a same event, and n_1 may have transmitted the virus to n_2 (during this event) at time t_2 if n_1 got infected at time t_1. These include the two pairs $\langle\langle \mathit{Alice}, 100\rangle, \langle \mathit{Bob}, 104\rangle\rangle$, and $\langle\langle \mathit{Alice}, 100\rangle, \langle \mathit{Bob}, 105\rangle\rangle$, meaning that if Alice contracted the virus on day 100, then she may have transmitted it to Bob during day 104 or day 105 (among other possibilities). In this query, the "/" operator is a join, and the subqueries `attends` and $\texttt{attends}^-$ act as filters on our initial temporal relation R, restricting it based on the time intervals contained in the graph. We can further restrict this relation by requiring that person n_2 tested positive at most a week after the possible transmission, with

$$q_2 = q_1/?\Big(\mathbf{T}_{[0,7]}/\texttt{tests}/(= \mathit{positive})\Big).$$

In this query, the operator $?q$ acts as an existential quantifier: it only requires the existence of a match for the subquery q, so that the output of q_2 is a subset of the output of q_1.

As we have seen above, the initial temporal relation R can be easily represented in a compact way, using the temporal domain $\mathcal{T}$ for the set of initial infection points, and a single interval of temporal distances $[3, 5]$ that indicates by how much each initial time point can be shifted. This holds regardless of whether $\mathcal{T}$ is discrete or continuous. However, this does not hold anymore for the outputs of our queries. For instance, the temporal relation induced by the answers to q_2 (which all have *Alice* and *Bob* as first and second node respectively) is uneven:

over discrete time (and with a granularity of one day), 100 is associated with a single day 105, whereas 101 is associated with both 105 and 106. Moreover, without a compact representation of query answers, altering the time granularity (e.g. from days to hours) may significantly increase the number of output tuples (exponentially in the size of the input, assuming that interval boundaries are encoded in binary). Besides, even in the case where the output of a query can be represented compactly, computing all answers before summarizing them may be prohibitive, because the number of intermediate results may impact the performance of (worst-case quadratic) join operations. Hence the need to not only represent answers in a compact way, but also maintain compactness during query evaluation. This is even a necessity over dense time, where the lack of a finite representation may forbid query evaluation. To our knowledge, these are still open questions (in particular, left open in [2]), which effectively rule out arbitrary selection, union and joins over binary temporal relations.

Contributions and Organization. In this work, we propose four alternative compact representations of answers to TRPQs. In Section 2, we provide an informal, graphical overview of each of them. In Section 3, we define the syntax and semantics of the query language that we study. In Section 4, we formally define and study our four representations: first, in Section 4.1, we present the two simpler ones, which aggregate tuples along a single temporal dimension, either starting points or distances; then, in Section 4.2, we present the two more complex representations, which aggregate tuples along both dimensions. We analyze the respective advantages and drawbacks of each representation in terms of finiteness (over dense time), compactness (when finite), uniqueness, and the computational cost of query answering and minimizing a set of tuples. Notably, the fourth (more complex) representation guarantees that compactness and finiteness are maintained throughout query evaluation. In Section 5, we discuss related work. An extended version of this paper, with full definitions and proofs, is available at https://arxiv.org/abs/2507.22143.

2 Overview

This section provides a high-level overview of the four compact representations of answers to TRPQs defined and studied in this article. As a running example, the following query q_3 outputs all pairs $\langle\langle n_1, t_1\rangle, \langle n_2, t_2\rangle\rangle$ such that, if Alice got infected at time t_1 while attending event n_1, then she may have transmitted the virus at time t_2 while attending event n_2:

$$q_3 = \texttt{attends}^- / (= \mathit{Alice}) / \mathbf{T}_{[3,5]} / \texttt{attends}.$$

Over discrete time, with days as temporal granularity, evaluating q_3 over the graph G of Figure 1 outputs a set $[\![q_3]\!]_G$ of 7 pairs, each of the form $\langle\langle \mathit{ICDT}, t_1\rangle, \langle \mathit{ISWC}, t_2\rangle\rangle$. Since the two nodes ($\mathit{ICDT}$ and ISWC) are identical for all answers, we can focus on the binary temporal relation induced by these, i.e. the relation $R = \{(t_1, t_2) \mid \langle\langle \mathit{ICDT}, t_1\rangle, \langle \mathit{ISWC}, t_2\rangle\rangle \in [\![q_3]\!]_G\}$. Each

pair $(t_1, t_2) \in R$ can equivalently be represented as the pair $(t_1, t_2 - t_1)$, where the second component is the temporal distance between t_1 and t_2. Accordingly, Figure 2a displays these 7 pairs over the Euclidean plane of (discrete) time per distance.

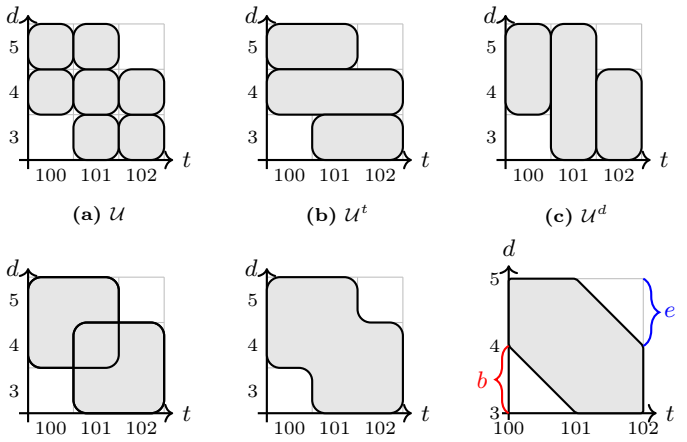

Fig. 2: Alternative representations of the answers to Query q_3, in the plane of time per distance

Each of the four compact representations that we study in this article is an alternative way to reduce the number of such pairs. The first representation, which we call $\mathcal{U}^t$, groups these pairs by distance (i.e. horizontally in our figure), while aggregating time points as intervals. In this example, this yields three compact answers, namely $([100, 101], 5)$, $([100, 102], 4)$, and $([101, 102], 3)$. Each of these is a rectangle with height 1 in our plane, as illustrated with Figure 2b, and their union is indeed the area of interest. The second representation, called $\mathcal{U}^d$, is symmetric, in the sense that it groups results by time points and aggregates distances, so that compact answers are now rectangle with width 1, as illustrated with Figure 2c. The first representation $\mathcal{U}^t$ ensures that the number of compact answers is independent of the size of the intervals present in the graph. However, this number is linear in the size of the intervals used in the query (via the temporal navigation operator). And conversely for $\mathcal{U}^d$.

A third natural attempt (which we call $\mathcal{U}^{td}$) consists in aggregating tuples along both dimensions, representing our binary relation R as a set of pairs of intervals, one of time points, the other of distances. These correspond to arbitrary rectangles in our plane. As can be seen on Figure 2d, two such rectangles are still needed to cover the whole area (or three if we forbid overlapping rectangles). Besides, some desirable properties of the first two representations are lost: the most compact representation (i.e. smallest set of rectangles needed to cover the area) is not unique anymore (e.g. if the area is an "L"-shaped polygon), and

minimizing the number of answers (i.e. finding a minimal set of such rectangles) becomes intractable (if we allow rectangles to overlap).

Moreover, for all three representations, the number of tuples needed to aggregate all answers to q_3 may grow with a change of time granularity. This can once again be seen in Figure 2a: if we adopt hours rather than days, then the number of "steps" (bottom left and top right of the area) gets multiplied by 24. At the limit (i.e. when granularity approaches 0), we reach dense time, and the area to cover becomes a rectangle cropped by two lines with slope -1 (drawn in Figure 2f). Clearly, such an area cannot be covered by finitely many rectangles.

This is why we introduce a fourth representation, which we call $\mathcal{U}^c$. Over dense time, a tuple in $\mathcal{U}^c$ precisely stands for an area like the one of Figure 2f, by means of two intervals (one for times, the other for distance), which intuitively represent the rectangle to be cropped, together with two values b and e that specify where the two cropping lines intersect respectively with this rectangle. This representation is also useful over discrete time: in our example, the whole area can be captured with a single tuple that consists of the interval $[100, 102]$ for times, as illustrated with Figure 2e, the interval $[3, 5]$ for distance, and the values 1 and 1 for b and e. Our most technical result is that binary relations that can be represented as such geometric shapes are closed under composition. As a consequence, a TRPQ can only produce a union of such shapes. So this last format $\mathcal{U}^c$ overcomes the limitations of the previous ones, in the sense that answers to a TRPQ can always be represented in a finite way. This is also the most compact of these four representations. However, as for $\mathcal{U}^{td}$, minimizing a set of tuples under this view is intractable, and the most compact representation of a set of answers is not unique.

Our main findings about each of the four representations are summarized at the end of the dedicated section, in Figure 6. As a brief summary, only the fourth representation ($\mathcal{U}^{td}$) guarantees that query answers remain finite and independent of the sizes of the input intervals. In comparison, the two first representations $\mathcal{U}^t$ and $\mathcal{U}^d$ benefit from a unique compact representation of query answers, and the fact that computing it (out of a non-compact one) is tractable. However, these can only be used over discrete time, and if the size of the input query (resp. graph) intervals remains small.

3 Preliminaries

Temporal Graphs. For maximal generality (and readability), we use a very simple graph data model, where each fact is a labelled edge, annotated with a set of time intervals for validity. Formally, we assume two infinite sets $\mathcal{N}$ of nodes and $\mathcal{E}$ of edge labels. As is conventional (e.g. in RDF), we represent a fact as a triple $(s, p, o) \in \mathcal{N} \times \mathcal{E} \times \mathcal{N}$ (where s, p and o intuitively stand for "subject", "property" and "object" respectively). In addition, for the temporal dimension of our data, we assume an underlying temporal domain $\mathcal{T}$ that may be either discrete or dense. For simplicity, we will use $\mathbb{Z}$ in the former case, and $\mathbb{Q}$ in the latter. We also use $\mathsf{intv}(\mathcal{T})$ for the set of nonempty intervals over $\mathcal{T}$.

In our model, a database instance simply assigns (finitely many) intervals of validity to (finitely many) triples. Precisely, a *temporal graph* $G = \langle \mathcal{T}_G, \mathcal{F}_G, \mathsf{val}_G \rangle$ consists of a bounded *effective temporal domain* $\mathcal{T}_G \in \mathsf{intv}(\mathcal{T})$, a finite set of triples $\mathcal{F}_G \subseteq \mathcal{N} \times \mathcal{E} \times \mathcal{N}$, and a function $\mathsf{val}_G \colon \mathcal{F}_G \to 2^{\mathsf{intv}(\mathcal{T}_G)}$ that maps a triple to a finite set of intervals over $\mathcal{T}_G$. For instance, in Figure 1, val_G assigns the (singleton) set of intervals $\{[104, 106]\}$ to the triple $(\mathit{Alice}, \texttt{attends}, \mathit{ISWC})$.

Temporal Regular Path Queries. We adopt the query language introduced in [2], but in a simpler form, made possible by the simplified data model defined above. We emphasise that this is without loss of expressivity.

A *Temporal Regular Path Query* (TRPQ) is an expression for the symbol "trpq" in the following grammar:

$$\begin{aligned}
\mathsf{trpq} &::= \mathsf{edge} \mid \mathsf{node} \mid \mathbf{T}_\delta \mid (\mathsf{trpq}/\mathsf{trpq}) \mid (\mathsf{trpq} + \mathsf{trpq}) \mid \mathsf{trpq}[m, n] \mid \mathsf{trpq}[m, _] \\
\mathsf{edge} &::= \mathit{label} \mid \mathsf{edge}^- \\
\mathsf{node} &::= \mathit{pred} \mid \leq k \mid (?\mathsf{trpq}) \mid \neg\, \mathsf{node}
\end{aligned}$$

with $\mathit{label} \in \mathcal{E}$, $k \in \mathcal{T}, \delta \in \mathsf{intv}(\mathcal{T})$, $m, n \in \mathbb{N}^+$, and $m \leq n$.

Here, edge and node are filters on edges and nodes respectively. The terminal symbol *pred* stands for a Boolean predicate that can be evaluated locally for one node n, which we write $n \models \mathit{pred}$ (for instance, in Figure 1, the node *positive* satisfies the predicate $(= \mathit{positive})$). Similarly, the Boolean predicate $\leq k$ evaluates whether a time point is less than or equal to k. The expression $(?\mathsf{trpq})$ filters the nodes that satisfy trpq, and $\neg$ represents logical negation. The temporal navigation operator $\mathbf{T}_\delta$ stands for navigation in time by any distance in the interval δ, and the remaining operators are regular path query (RPQ) operators: $/$ stands for join, $+$ for union, and $\mathsf{trpq}[m, n]$ for the "repetition" of trpq from m to n times. In particular, $\mathsf{trpq}[0, \_]$ represents Kleene closure (equivalent to the $*$ operator in regular expressions).

The formal semantics of TRPQs is provided in Figure 3, where $[\![q]\!]_G$ is the evaluation of a TRPQ q over a temporal graph G. In this definition, we use q^i for the TRPQ defined inductively by $q^1 = q$ and $q^{j+1} = q^j/q$. For convenience, we represent (w.l.o.g.) an answer as two nodes together with a time point and a distance, rather than two nodes and two time points, i.e. we use tuples of the form $\langle n_1, n_2, t, d \rangle$ rather than $\big\langle \langle n_1, t \rangle, \langle n_2, t + d \rangle \big\rangle$. We also use $\mathcal{N}_G$ to denote the set of nodes that intuitively appear in G, i.e. all n such that the triple (n, p, o) or (s, p, n) is in $\mathcal{F}_G$ for some s, p and o.

Operations on intervals. For two intervals $\alpha, \beta \in \mathsf{intv}(\mathcal{T})$, we use $\alpha \oplus \beta$ (resp. $\alpha \ominus \beta$) to denote the interval $\{a + b \mid a \in \alpha \text{ and } b \in \beta\}$ (resp. $\{a - b \mid a \in \alpha \text{ and } b \in \beta\}$). We also use $\alpha + b$ (resp. $\alpha - b$) for $\alpha \oplus [b, b]$ (resp. $\alpha \ominus [b, b]$).

4 Compact answers

In this section, we define and study the four representations of answers to a TRPQ sketched in Section 2. Let $\mathcal{U}$ denote the universe of all tuples that may be

$$
\begin{aligned}
[\![\mathit{label}]\!]_G &= \{\langle n_1, n_2, t, 0\rangle \mid t \in \tau \text{ for some } \tau \in \mathsf{val}_G(n_1, \mathit{label}, n_2)\} \\
[\![\mathsf{edge}^-]\!]_G &= \{\langle n_2, n_1, t, 0\rangle \mid \langle n_1, n_2, t, 0\rangle \in [\![\mathsf{edge}]\!]_G\} \\
[\![\mathit{pred}]\!]_G &= \{\langle n, n, t, 0\rangle \mid n \models \mathit{pred} \text{ and } t \in \mathcal{T}_G\} \\
[\![\leq k]\!]_G &= \{\langle n, n, t, 0\rangle \mid n \in \mathcal{N}_G, t \in \mathcal{T}_G \text{ and } t \leq k\} \\
[\![\mathbf{T}_\delta]\!]_G &= \{\langle n, n, t, d\rangle \mid n \in \mathcal{N}_G, t \in \mathcal{T}_G, d \in \delta \text{ and } t + d \in \mathcal{T}_G\} \\
[\![?\mathsf{trpq}]\!]_G &= \{\langle n, n, t, 0\rangle \mid \langle n, n', t, d\rangle \in [\![\mathsf{trpq}]\!]_G \text{ for some } n' \in \mathcal{N} \text{ and } d \in \mathcal{T}\} \\
[\![\neg\mathsf{node}]\!]_G &= \big(\{\langle n, n\rangle \mid n \in \mathcal{N}_G\} \times \mathcal{T}_G \times \{0\}\big) \setminus [\![\mathsf{node}]\!]_G \\
[\![\mathsf{trpq}_1/\mathsf{trpq}_2]\!]_G &= \{\langle n_1, n_3, t, d_1 + d_2\rangle \mid \\
&\qquad \exists n_2 \colon \langle n_1, n_2, t, d_1\rangle \in [\![\mathsf{trpq}_1]\!]_G \text{ and } \langle n_2, n_3, t + d_1, d_2\rangle \in [\![\mathsf{trpq}_2]\!]_G\} \\
[\![\mathsf{trpq}_1 + \mathsf{trpq}_2]\!]_G &= [\![\mathsf{trpq}_1]\!]_G \cup [\![\mathsf{trpq}_2]\!]_G \\
[\![\mathsf{trpq}[m, n]]\!]_G &= \bigcup_{k=m}^{n} [\![\mathsf{trpq}^k]\!]_G \\
[\![\mathsf{trpq}[m, \_]]\!]_G &= \bigcup_{k\geq m} [\![\mathsf{trpq}^k]\!]_G
\end{aligned}
$$

Fig. 3: Semantics of TRPQs

output by TRPQs, i.e. $\mathcal{U} = \mathcal{N}\times\mathcal{N}\times\mathcal{T}\times\mathcal{T}$. Then each of our four representations can be viewed as an alternative format to encode subsets of $\mathcal{U}$. We specify each of these four formats as a set of admissible tuples, denoted $\mathcal{U}^t$, $\mathcal{U}^d$, $\mathcal{U}^{td}$ and $\mathcal{U}^c$ respectively.

Definition 1 (Unfolding, compact set of tuples, finite representation). *Let $\mathcal{U}^x$ be one of $\mathcal{U}^t$, $\mathcal{U}^d$, $\mathcal{U}^{td}$ or $\mathcal{U}^c$. A tuple $\mathbf{u}$ in $\mathcal{U}^x$ represents a subset of $\mathcal{U}$, which we call the* unfolding *of $\mathbf{u}$. And the unfolding of a* set *$U \subseteq \mathcal{U}^x$ of such tuples is the union of the unfoldings of the elements of U. We say that U is* compact *if it is finite and if no strictly smaller (w.r.t. cardinality) subset of $\mathcal{U}^x$ has the same unfolding. A set $V \subseteq \mathcal{U}$ can be* finitely represented *(in $\mathcal{U}^x$) if there is a finite $U \subseteq \mathcal{U}^x$ with unfolding V.*

4.1 Folding time points ($\mathcal{U}^t$) or distances ($\mathcal{U}^d$)

Each of our two first compact representations aggregates tuples in $\mathcal{U}$ along one dimension: either the time point associated to the first node, or the distance between times points associated to each node. The corresponding universes $\mathcal{U}^t$ and $\mathcal{U}^d$ of tuples are $\mathcal{N}\times\mathcal{N}\times\mathsf{intv}(\mathcal{T})\times\mathcal{T}$ and $\mathcal{N}\times\mathcal{N}\times\mathcal{T}\times\mathsf{intv}(\mathcal{T})$ respectively. The unfolding of a tuple $\langle n_1, n_2, \tau, d\rangle \in \mathcal{U}^t$ is $\{\langle n_1, n_2, t, d\rangle \mid t \in \tau\}$, and analogously $\{\langle n_1, n_2, t, d\rangle \mid d \in \delta\}$ for a tuple $\langle n_1, n_2, t, \delta\rangle \in \mathcal{U}^d$.

Inductive representation. In order to understand when the answers $[\![q]\!]_G$ to a TRPQ q over a temporal graph G can be finitely represented in $\mathcal{U}^t$ or $\mathcal{U}^d$, and what the size of such representations may be, we define by structural induction on q two (not necessarily compact) representation of $[\![q]\!]_G$ in $\mathcal{U}^t$ and $\mathcal{U}^t$, noted $(\!|q|\!)^t_G$ and $(\!|q|\!)^d_G$ respectively (these representations also pave the way for implementing query evaluation). For instance, in the case where q is of the form *pred*, we define $(\!|\mathit{pred}|\!)^t_G$ as $\{\langle n, n, \mathcal{T}_G, 0\rangle \mid n \models \mathit{pred}\}$.

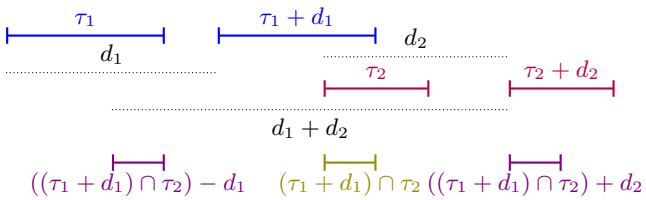

Fig. 4: Join of two tuples $\langle n_1, n_2, \tau_1, d_1\rangle$ and $\langle n_2, n_3, \tau_2, d_2\rangle$ in $\mathcal{U}^t$. Each tuple is depicted as two intervals τ_i and $\tau_i + d_i$. The two corresponding intervals τ_o and $\tau_o + d_o$ for the output tuple $\langle n_1, n_3, \tau_o, d_o\rangle$ are in violet.

Due to space limitations, we only provide the full definition of $(\!|q|\!)^t_G$ and $(\!|q|\!)^d_G$ in the extended version of this paper (together with proofs of correctness). We highlight here the least obvious operators. The first one is the temporal join $\mathsf{trpq}_1/\mathsf{trpq}_2$, which intuitively composes temporal relations. If we assume (by induction) that the answers to each operand (trpq_1 and trpq_2) are represented in $\mathcal{U}^t$, then the representation of $\mathsf{trpq}_1/\mathsf{trpq}_2$ in $\mathcal{U}^t$ can be computed as a regular join together with simple arithmetic operations on interval boundaries, as follows:

$$(\!|\mathsf{trpq}_1/\mathsf{trpq}_2|\!)^t_G = \Big\{\langle n_1, n_3, ((\tau_1 + d_1) \cap \tau_2) - d_1, d_1 + d_2\rangle \mid \exists n_2 \colon$$
$$\langle n_1, n_2, \tau_1, d_1\rangle \in (\!|\mathsf{trpq}_1|\!)^t_G, \langle n_2, n_3, \tau_2, d_2\rangle \in (\!|\mathsf{trpq}_2|\!)^t_G \text{ and } (\tau_1 + d_1) \cap \tau_2 \neq \emptyset \Big\}.$$

This observation also holds for $\mathcal{U}^d$, but with different operations on interval boundaries:

$$(\!|\mathsf{trpq}_1/\mathsf{trpq}_2|\!)^d_G = \Big\{\langle n_1, n_3, t_1, t_2 - t_1 + \delta_2\rangle \mid \exists n_2 \colon \langle n_1, n_2, t_1, \delta_1\rangle \in (\!|\mathsf{trpq}_1|\!)^d_G,$$
$$\langle n_2, n_3, t_2, \delta_2\rangle \in (\!|\mathsf{trpq}_2|\!)^d_G \text{ and } t_2 - t_1 \in \delta_1\Big\}.$$

The less obvious of these two operations is $(\!|\mathsf{trpq}_1/\mathsf{trpq}_2|\!)^t_G$, which we illustrate with Figure 4).

Another case of interest is the temporal navigation operator $\mathbf{T}_\delta$, for the second representation $\mathcal{U}^d$ (the case of $\mathcal{U}^t$ is trivial). Consider a query of the form $q/\mathbf{T}_\delta$ (or symmetrically $\mathbf{T}_\delta/q$). If the subquery $\mathbf{T}_\delta$ is evaluated independently, then the output of this subquery may be infinite over dense time, which rules out in practice an inductive evaluation:

$$(\!|\mathbf{T}_\delta|\!)^d_G = \{\langle n, n, t, ((\delta + t) \cap \mathcal{T}_G) - t\rangle \mid n \in \mathcal{N}_G, t \in \mathcal{T}_G, (\delta + t) \cap \mathcal{T}_G \neq \emptyset\}.$$

However, if $(\!|q|\!)^d_G$ is finite, then the output of the whole query $q/\mathbf{T}_\delta$ can be represented finitely, and effectively computed as follows:

$$(\!|q/\mathbf{T}_\delta|\!)^d_G = \{\langle n_1, n_2, t, (\delta' \oplus \delta) \cap \mathcal{T}_G\rangle \mid \langle n_1, n_2, t, \delta'\rangle \in (\!|q|\!)^d_G, (t + (\delta' \oplus \delta)) \cap \mathcal{T}_G \neq \emptyset\}.$$

Finiteness over dense time. Over discrete time, trivially, $[\![q]\!]_G$ can be finitely represented in $\mathcal{U}$,[2] therefore as well in any of our four compact representations.

[2] Recall that we assume the effective temporal domain $\mathcal{T}_G$ of G to be bounded.

But over dense time, this is not always possible. For instance, if q_t is the query $(= \mathit{positive})/\mathtt{tests}^-/\mathbf{T}_{[-7,0]}$, and if the graph G of Figure 1 is interpreted over dense time, then $\langle \mathit{positive}, \mathit{Bob}, 112, d\rangle \in [\![q_t]\!]_G$ for every rational number d in $[-7, 0]$. And no tuple in $\mathcal{U}^t$ can represent more than one of these tuples. From the definition of $(\!|q|\!)^t_G$, the only possible source of non-finiteness for $\mathcal{U}^t$ is the temporal navigation operator $\mathbf{T}_\delta$, and only if δ is not a singleton interval (i.e., if it specifies a certain *range* rather than a fixed distance).

Similarly, if q_d is the query $(= \mathit{Bob})/\mathtt{attends}$, then $\langle \mathit{Bob}, \mathit{ISWC}, t, 0\rangle \in [\![q]\!]_G$, for every rational number t in $[102, 107]$, and no tuple in $\mathcal{U}^d$ can represent more than one of these.

Compactness. If $[\![q]\!]_G$ can be finitely represented in $\mathcal{U}^t$ or $\mathcal{U}^d$, then a natural requirement on these representations is conciseness. It is easy to see that a finite set $U \subseteq \mathcal{U}^t$ is compact iff all time intervals for the same n_1, n_2 and d within U are coalesced. Formally, let $\sim$ denote the binary relation over $\mathcal{U}^t$ defined by $\langle n_1, n_2, \tau_1, d_1\rangle \sim \langle n_3, n_4, \tau_2, d_2\rangle$ iff $\langle n_1, n_2, d_1\rangle = \langle n_3, n_4, d_2\rangle$ and $\tau_1 \cup \tau_2 \in \mathsf{intv}(\mathcal{T})$. Then U is compact iff $\mathbf{u}_1 \not\sim \mathbf{u}_2$ for all $\mathbf{u}_1, \mathbf{u}_2 \in U$ s.t. $\mathbf{u}_1 \neq \mathbf{u}_2$. More, there is a unique way to coalesce a finite set of intervals. Therefore if $V \subseteq \mathcal{U}$ can be finitely represented in $\mathcal{U}^t$, then V also has a unique compact representation in $\mathcal{U}^t$. A symmetric observation holds for a set $U \subseteq \mathcal{U}^d$, where we coalesce distance intervals rather than time intervals.

Coalescing a set of intervals is known to be in $O(n \log n)$, and efficient implementations have been devised (see Section 5). For this reason, coalescing intermediate results in $(\!|q|\!)^t_G$ or $(\!|q|\!)^d_G$ may be an interesting query evaluation strategy. For instance, this could reduce the size of the operands of a (worst-case quadratic) temporal join.

Size of compact answers. Our two representations $\mathcal{U}^t$ and $\mathcal{U}^d$ exhibit an interesting symmetry when it comes to the size of compact answers to a query, intuitively growing with the size of the intervals present in q in the case of $\mathcal{U}^t$, and the intervals present in G in the case of $\mathcal{U}^d$. This suggests that $\mathcal{U}^t$ is be better suited to large graphs intervals and small temporal navigation intervals, and conversely for $\mathcal{U}^d$.

Unfortunately, this does not hold for arbitrary queries. Indeed, the size of a compact representation of $[\![q]\!]_G$ may be affected by the size of the effective temporal domain $\mathcal{T}_G$ alone, even if all time intervals in the query and graph are singletons, as soon as q contains an occurrence of the closure operator $[m, \_]$, as illustrated with the following example:

Example 1. Consider a temporal graph G s.t. $\mathsf{val}_G(n_1, e, n_2) = \{[0, 0]\}$ and consider the query $q = e/(\mathrm{T}_{[2,2]})[1, \_]$. The compact representation of $[\![q]\!]_G$ is $\{\langle n_1, n_2, [0, 0], d\rangle \mid d \in D\}$ in $\mathcal{U}^t$, and $\{\langle n_1, n_2, 0, [d, d]\rangle \mid d \in D\}$ in $\mathcal{U}^d$, where $D = \{d \in \mathcal{T}_G \cap \mathbb{N}^+ \mid d \bmod 2 = 0\}$. □

However, for queries without closure operator, which we call *closure-free*, the symmetry sketched above holds. To formalize this, we introduce a notation that we will reuse for the other two representations.

Definition 2 (Size of compact answers to closure-free queries). *Let $\mathcal{U}^x$ be one of $\mathcal{U}^t$, $\mathcal{U}^d$, $\mathcal{U}^{td}$ or $\mathcal{U}^c$. Consider a temporal graph G and a closure-free query q, such that $[\![q]\!]_G$ can be finitely represented in $\mathcal{U}^x$. We fix G and q, with the exception of $\mathcal{T}_G$, and either* (i) *the intervals present in G, or* (ii) *the intervals present in q (with the requirement that $[\![q]\!]_G$ can still be finitely represented in $\mathcal{U}^x$). In case (i), let n be the cumulated length of the intervals in G, and let V be a compact representation of $[\![q]\!]_G$ in $\mathcal{U}^x$. We use $\#\mathsf{answers}^\tau(\mathcal{U}^x)$ for the function that maps n to the the cardinality of V. In case (ii), we use $\#\mathsf{answers}^\delta(\mathcal{U}^x)$ with an identical meaning, but where n is the cumulated size of the intervals in q.*

The following results says that the size of the compact representation of $[\![q]\!]_G$ in $\mathcal{U}^t$ (when it exists) may be affected by the size of the intervals present in q, but not the ones used to label triples in G, and conversely for $\mathcal{U}^d$

Proposition 1. *In the worst case,*[3]

(fixed query intervals)	*(fixed graph intervals)*
$\#\mathsf{answers}^\tau(\mathcal{U}^t) = O(1)$	$\#\mathsf{answers}^\delta(\mathcal{U}^t) = \Omega(n)$
$\#\mathsf{answers}^\tau(\mathcal{U}^d) = \Omega(n)$	$\#\mathsf{answers}^\delta(\mathcal{U}^d) = O(1)$

Complexity of query answering. Let $\mathcal{U}^x$ be one of $\mathcal{U}^t$, $\mathcal{U}^d$, $\mathcal{U}^{td}$ or $\mathcal{U}^c$. We formulate a decision problem analogous to the classical boolean query answering problem (for atemporal databases), in such a way that it remains defined even if $[\![q]\!]_G$ does not admit a finite representation in $\mathcal{U}^x$. We say that a tuple $\mathbf{u}$ in $\mathcal{U}^x$ is a *compact answer* to q over G if its unfolding is a subset of $[\![q]\!]_G$ and is maximal among the tuples in $\mathcal{U}^x$ that satisfy this condition. We can now define our (four) problems (where x is either t, d, td or c):

CompactAnswer^x
Input: temporal graph G, TRPQ q, tuple $\mathbf{u} \in \mathcal{U}^x$
Decide: $\mathbf{u}$ is a compact answer to q over G

Complexity for these problems is (partly) driven by the size of the input time intervals, and there is no reason a priori to assume that intervals in the graphs are larger than the ones in the query. This is why we do not focus on data complexity (where the query is fixed), but instead on combined complexity, where the size of the query and data may vary (we leave for future work a finer-grained analysis). We show in the extended version of this paperthat the results proven in [3] for answering TRPQs in $\mathcal{U}$ immediately transfer to $\mathcal{U}^t$ (resp. $\mathcal{U}^d$), even in the case where $[\![q]\!]_G$ cannot be finitely represented in $\mathcal{U}^t$ (resp. $\mathcal{U}^d$):

Proposition 2.
CompactAnswer^t and CompactAnswer^d are PSpace-complete.

We also observe that hardness can be proven with a graph of fixed size, with the exception of the effective temporal domain $\mathcal{T}_G$. However, the number of operators of the query used in this reduction is not fixed.

[3] We write "wort case" here to clarify that this is for the worst possible inputs with size n (as opposed to average size for instance).

4.2 Folding time points and distances ($\mathcal{U}^{td}$ and $\mathcal{U}^{c}$)

Each of the two compact representations $\mathcal{U}^t$ and $\mathcal{U}^d$ defined above aggregates tuples in $\mathcal{U}$ along one dimension (either time points or distances), and both may fail to represent the answers to a query compactly (or even finitely over dense time). So a first natural attempt to address this limitation consists in aggregating tuples along both dimensions. Accordingly, we define the universe $\mathcal{U}^{td}$ as $\mathcal{N} \times \mathcal{N} \times \mathsf{intv}(\mathcal{T}) \times \mathsf{intv}(\mathcal{T})$, and the unfolding of $\langle n_1, n_2, \tau, \delta \rangle \in \mathcal{U}^{td}$ as $\{\langle n_1, n_2, t, d \rangle \mid t \in \tau, d \in \delta\}$.

This representation is more compact than the two previous ones, since unfolding a tuple in $\mathcal{U}^t$ or $\mathcal{U}^d$ yields a subset of some unfolded tuple of $\mathcal{U}^{td}$. However, as we will see below, minimizing the cardinality of a set of query answers becomes intractable. Besides, maybe surprisingly, this new format may also fail to represent query answers in a compact fashion (or finitely over dense time). To see this, observe that for two fixed nodes, a tuple in $\mathcal{U}^{td}$ has a natural representation as a rectangle in the Euclidean plane P of times per distances. The following example shows a query whose output, depicted in Figure 5b, cannot be represented (over dense time) as a finite set of rectangles in P. This example also provides insight about the formalization of our fourth representation (below).

Example 2. Consider a temporal graph G with two edges such that $\mathsf{val}_G(n_1, e_1, n_2) = \{[0, 2]\}$ and $\mathsf{val}_G(n_2, e_2, n_3) = \{[1, 3]\}$ Let q be the query $e_1/\mathbf{T}_{[0,2]}/e_2$. Then $[\![q]\!]_G = \{\langle n_1, n_2, t, d \rangle \mid t \in [0, 2], d \in [0, 2] \text{ and } t + d \in [1, 3]\}$. So for an answer $\langle n_1, n_2, t, d \rangle \in [\![q]\!]_G$, we have

$$1 \leq t + d \leq 3, \quad \text{that is,} \quad 1 - t \leq d \leq 3 - t.$$

Besides, from the query q, we get $0 \leq d \leq 2$. Therefore

$$\max(1 - t, 0) \leq d \leq \min(3 - t, 2).$$

This observation gives us an interval

$$\delta_t = [\max(1 - t, 0), \min(3 - t, 2)]$$

of admissible distances for each $t \in [0, 2]$, so that $[\![q]\!]_G = \{\langle n_1, n_2, t, d \rangle \mid t \in [0, 2] \text{ and } d \in \delta_t\}$. Extending this kind of reasoning to the general case, we derive the formula (1) presented below for δ_t. □

As shown in Figure 5b, the area covered by the output of this query can be viewed as a rectangle cropped by two parallel lines, each with slope -1. These cropped rectangles turn out to have an essential property: if two temporal relations have such a shape, then their composition also does. As a consequence, associating each tuple with such a cropped rectangle (as opposed to a regular rectangle in $\mathcal{U}^{td}$) allows us to solve our initial problem: compute inductively the output of a TRPQ q over a graph G in such a way that the number of output tuples remains independent of the size of the intervals present in either q or G. To represent these cropped rectangles, we introduce a fourth format, which we call $\mathcal{U}^c$ (where

"c" stands for "cropped"), where each tuple carries, in addition to a time interval τ and a distance interval δ (a.k.a. a rectangle), the two values $b, e \in \mathcal{T}$ depicted in Figure 5b, which intuitively indicate where the cropping lines intersect the rectangle induced by τ and δ. For each time point $t \in \mathcal{T}$, these define a specific range of distances $\delta_t \subseteq \delta$, as

$$\delta_t = {}_{\delta}\lfloor\ b_\delta + \max(0, b - t)\ ,\ e_\delta - \max(0, t - e)\ \rfloor_\delta \tag{1}$$

where ${}_{\delta}\lfloor$ and $\rfloor_\delta$ stand for the left and right delimiters of δ, and b_δ and e_δ for its left and right boundaries For instance, if $\delta = [2, 6)$, then ${}_{\delta}\lfloor$ is "[", b_δ is 2, e_δ is 6 and $\rfloor_\delta$ is ")".

Accordingly, a tuple $\langle n_1, n_2, \tau, \delta, b, e\rangle \in \mathcal{N} \times \mathcal{N} \times \mathsf{intv}(\mathcal{T}) \times \mathsf{intv}(\mathcal{T}) \times \mathcal{T} \times \mathcal{T}$ is in $\mathcal{U}^c$ iff δ_t is nonempty for every $t \in \tau$. And the unfolding of this tuple is $\{\langle n_1, n_2, t, d\rangle \mid t \in \tau, d \in \delta_t\}$.

Inductive representation. As we did for our first two representations, we define (in the extended version of this paper) by structural induction on q two representations $(\!|q|\!)^{td}_G$ and $(\!|q|\!)^{c}_G$ of $[\![q]\!]_G$ in $\mathcal{U}^{td}$ and $\mathcal{U}^c$ respectively. Our most technical result is correctness of this definition for the join operator in $\mathcal{U}^c$ (and to a lesser extent the temporal navigation operator).

We reproduce these definitions here to make apparent the fact these can be computed by means of simple arithmetic operations on time points and interval boundaries (together with a regular join on nodes for the join operator). The inductive evaluation $(\!|\mathsf{trpq}_1/\mathsf{trpq}_2|\!)^c_G$ in $\mathcal{U}^c$ of the query $\mathsf{trpq}_1/\mathsf{trpq}_2$ is defined as:

$$\{\mathbf{u}_1 \overline{\bowtie} \mathbf{u}_2 \mid \mathbf{u}_1 \in (\!|\mathsf{trpq}_1|\!)^c_G, \mathbf{u}_2 \in (\!|\mathsf{trpq}_2|\!)^c_G, \mathbf{u}_1 \sim \mathbf{u}_2\}$$

where $\mathbf{u}_1 \sim \mathbf{u}_2$ and $\mathbf{u}_1 \overline{\bowtie} \mathbf{u}_2$ are defined as follows.
For $\mathbf{u}_1 = \langle n_1, n_2, \tau_1, \delta_1, b_1, e_1\rangle$ and $\mathbf{u}_2 = \langle n_3, n_4, \tau_2, \delta_2, b_2, e_2\rangle$, let

$$\begin{aligned}\delta'_1 &= {}_{\delta_1}\lfloor\ b_{\delta_1} + \max(0, b_1 - b_{\tau_1}), e_{\delta_1} - \max(0, e_{\tau_1} - e_1)\ \rfloor_{\delta_1},\ \text{and}\\ \tau &= (((\tau_1 \oplus \delta'_1) \cap \tau_2) \ominus \delta'_1) \cap \tau_1.\end{aligned}$$

Then the relation $\sim\ \subseteq \mathcal{U}^c \times \mathcal{U}^c$ is defined as $\mathbf{u}_1 \sim \mathbf{u}_2$ iff $n_2 = n_3$ and $\tau \neq \emptyset$.
And if $\mathbf{u}_1 \sim \mathbf{u}_2$, then $\mathbf{u}_1 \overline{\bowtie} \mathbf{u}_2$ is defined as $\langle n_1, n_4, \tau, \delta_1 \oplus \delta_2, b, e\rangle$, with $b = \max(b_1, b_2 - b_{\delta_1})$ and $e = \min(e_1, e_2 - e_{\delta_1})$.

Similarly, if $b_{\mathcal{T}_G}$ and $e_{\mathcal{T}_G}$ are the boundaries of the interval $\mathcal{T}_G$, then $(\!|\mathbf{T}_\delta|\!)^c_G$ is defined as

$$\{\langle n, n, \mathcal{T}_G, \delta, b_{\mathcal{T}_G}, e_{\mathcal{T}_G}\rangle \overline{\bowtie} \langle n, n, \mathcal{T}_G, [0, 0], b_{\mathcal{T}_G}, e_{\mathcal{T}_G}\rangle \mid n \in \mathcal{N}_G\}.$$

Finiteness over dense time. We already illustrated in Figure 5b why $[\![q]\!]_G$ may not be finitely representable in $\mathcal{U}^{td}$. In contrast, $[\![q]\!]_G$ can be finitely represented in $\mathcal{U}^c$, as a direct consequence of the (correctness of the) inductive definition of $(\!|q|\!)^c_G$, which produces finitely many tuples (notably, the operator $\mathsf{trpq}_1/\mathsf{trpq}_2$ produces at most one tuple per pair $(\mathbf{u}_1, \mathbf{u}_2) \in (\!|\mathsf{trpq}_1|\!)^c_G \times (\!|\mathsf{trpq}_2|\!)^c_G$).

Compactness. Let $U \subseteq \mathcal{U}^{td}$ and $V \subseteq \mathcal{U}$ be two sets of tuples that share the same nodes n_1 and n_2. Then U unfolds as V iff they intuitively cover the same

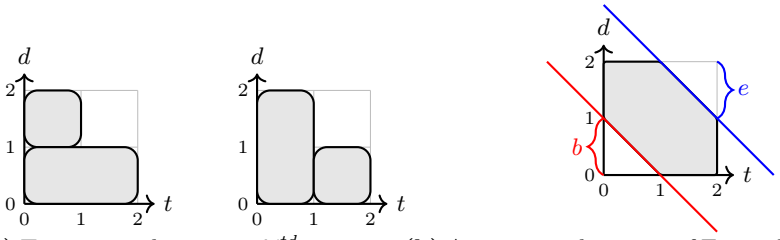

Fig. 5: **(a)** Two minimal covers in $\mathcal{U}^{td}$ **(b)** Answers to the query of Example 2

area in the Euclidean plane of times per distances, i.e. if

$$\bigcup\{\tau \times \delta \mid \langle n_1, n_2, \tau, \delta\rangle \in U\} = \{(t,d) \mid \langle n_1, n_2, t, d\rangle \in V\}.$$

There may be several compact representations in $\mathcal{U}^{td}$ for the same $V \subseteq \mathcal{U}$. For instance, the minimal number of rectangle needed to cover an "L"-shaped polygon is two, and there are several such covers, as illustrated with Figure 5a. This argument easily generalizes to discrete time.

Besides, minimizing the representation of $[\![q]\!]_G$ in $\mathcal{U}^{td}$ (i.e. computing a minimal set of tuples in $\mathcal{U}^{td}$ with unfolding $[\![q]\!]_G$) is intractable. To see this, we first observe that computing such a set is harder than deciding whether there exists one with size k (for a given k). Next, the following problem is known to be NP-complete: given a rectilinear polygon r and a number k, decide whether there is a set of at most k (possibly overlapping) rectangles that exactly cover r [9,4]. This problem reduces to ours, observing that for any rectilinear polygon r, a query q (with only unions) and graph G can be constructed in polynomial time so that $[\![q]\!]_G$ covers exactly r. This hardness result also immediately translates to $\mathcal{U}^c$: a rectangle is a specific case of a cropped rectangle, therefore the same reduction can be used, and if there is a minimization with at most k tuples in $\mathcal{U}^{td}$, then there is also one in $\mathcal{U}^c$.

An important difference though between these two representations and the two previous ones ($\mathcal{U}^t$ and $\mathcal{U}^d$) is that a compact set U of answers in $\mathcal{U}^{td}$ or $\mathcal{U}^c$ may be *redundant*, meaning that two tuples $\mathbf{u}_1$ and $\mathbf{u}_2$ in U may have overlapping unfoldings. This can for instance be seen in Figure 2d, for $\mathcal{U}^{td}$. If we require tuples to be non-redundant, then the representations of answers to a query is in general less concise. But it may be better suited for downstream tasks, such as aggregation. Besides, tractability of minimization in $\mathcal{U}^{td}$ is regained, because finding a cover with a minimal number of non-overlapping rectangles is tractable [20,14]. However, uniqueness is not regained, as shown in Fig. 5a.

Size of compact answers. If $V \subseteq \mathcal{U}$ can be finitely represented in $\mathcal{U}^{td}$, then trivially, a compact representation of V in $\mathcal{U}^{td}$ must be smaller than the compact representation of V in $\mathcal{U}^t$ (resp. $\mathcal{U}^d$), if the latter exists. So compact answers under this representation must be smaller than under the two previous ones. However, the size of a compact representation may still be affected by the size of time intervals in q (for a closure-free q already). In contrast, for $\mathcal{U}^c$, immediately from the definition of $(\!|q|\!)^c_G$, the number of tuples in $(\!|q|\!)^c_G$ (for a closure-free q)

	Finite (dense time)	Unique	Size (closure-free q)		Minimization	Query answering
			graph intervals	query intervals		
$\mathcal{U}^t$	no	yes	$O(1)$	$\Omega(n)$	$O(n \log n)$	PSPACE-c
$\mathcal{U}^d$	no	yes	$\Omega(n)$	$O(1)$	$O(n \log n)$	PSPACE-c
$\mathcal{U}^{td}$	no	no	$O(1)$	$\Omega(n)$	NP-h / $O(n^{2.5})$	PSPACE-c
$\mathcal{U}^c$	yes	no	$O(1)$	$O(1)$	NP-h	PSPACE-h

Fig. 6: Summary of results. For applications with discrete time and short intervals in the query (resp. graph), $\mathcal{U}^t$ (resp. $\mathcal{U}^d$) provides an efficient solution. $\mathcal{U}^{td}$ is more compact than both, and therefore potentially more efficient. However, minimizing the number of query answers is more expensive, which makes it less suitable for applications requiring aggregation. For applications with dense time or long intervals in both the graph and the query, $\mathcal{U}^c$ is the only viable option.

is independent of the size of the intervals in G or q (even though $(\!|q|\!)^c_G$ is not necessarily compact).

Proposition 3. *In the worst case,*

(fixed query intervals)	*(fixed graph intervals)*
$\#\mathsf{answers}^\tau(\mathcal{U}^{td}) = O(1)$	$\#\mathsf{answers}^\delta(\mathcal{U}^{td}) = \Omega(n)$
$\#\mathsf{answers}^\tau(\mathcal{U}^c) = O(1)$	$\#\mathsf{answers}^\delta(\mathcal{U}^c) = O(1)$

Complexity of query answering. The hardness results of [3] over $\mathcal{U}$ can also be lifted to $\mathcal{U}^{td}$ and $\mathcal{U}^c$, and this bound is tight for $\mathcal{U}^{td}$:

Proposition 4.
CompactAnswer^{td} is PSpace-complete, CompactAnswer^c is PSpace-hard.

5 Related work

In temporal relational databases, tuples (or attributes) are most commonly associated with a *single* time interval that represents either validity or transaction time [5]. Intervals are commonly used instead of time points as a compact representation. To maintain a compact and unique representation through operations, the *coalescing operator*, which merges value-equivalent tuples over consecutive or overlapping time intervals, has received a lot of attention. Böhlen et al. [6] showed that coalescing can be implemented in SQL, and provided a comprehensive analysis of various coalescing algorithms. Al-Kateb et al. [1] investigated coalescing in the attribute timestamped CME temporal relational model and the work of [12] defines coalescing for temporal multi-set relations. Zhou et al. [23] exploited SQL:2003's analytical functions for the computation of coalescing, which to date, is the state-of-the-art technique and can be computed efficiently in $\mathcal{O}(n \log n)$. In our first two representations $\mathcal{U}^t$ and $\mathcal{U}^d$, which use a single interval, coalescing can be used to achieve a compact representation. However, this is not applicable for $\mathcal{U}^{td}$ and $\mathcal{U}^c$. Bitemporal databases [18] use

two intervals (or rectangles) as a compact representation for two time points, typically one for validity and one for transaction times. Our third representation $\mathcal{U}^{td}$ also uses two intervals. but both are used for (a generalized form of) validity (we are not modeling transaction times). In bitemporal databases, coalescing does not provide a unique representation [22] and, since it is designed as a non-redundant operation, may not provide the most compact representation.

Also relevant to our work is the efficient computation of temporal joins over intervals. There has been a long line of research on temporal joins [16], ranging from partition-based [10,8], index-based [13,19], and sorting based [21,7] techniques. Recently, in [11] it has been shown that a temporal join with the overlap predicate can be transformed into a sequence of two range joins. Our inductive representations of answers require temporal joins and range joins (e.g. for $\mathcal{U}^t$ and $\mathcal{U}^d$ respectively).

Finally, this paper builds upon the original proposal made in [2] to extend regular path queries over property graphs with a temporal navigation operator—effectively allowing selection, join, and union of binary temporal relations. Our work focuses on producing compact answers that are finite even in the case of continuous time, a problem that was left open in [2].

6 Conclusions

We investigated how to compactly represent answers to queries over binary temporal relations in the TRPQ setting, where both data and queries may include time intervals for validity and temporal navigation, respectively. To our knowledge, this question was previously open. We defined and analyzed four alternative representations of compact answers to TRPQs, varying in conciseness and potential use. Notably, the fourth representation ensures that query answers are always finitely representable, and that their number is independent of the length of the input (graph and query) intervals. We see this as a useful step towards integrating temporal navigation into database systems. An open question is whether non-redundancy is tractable under this fourth representation, in particular whether tractability for minimizing compact answers is regained.

Supplemental Material Statement. An extended version with proofs of our results is available online, at https://arxiv.org/abs/2507.22143.

Acknowledgments. This research has been partially supported by the HEU project CyclOps (GA n. 101135513), by the Province of Bolzano and FWF through project OnTeGra (DOI 10.55776/PIN8884924), by the Province of Bolzano and EU through projects ERDF-FESR 1078 CRIMA and ERDF-FESR 1047 AI-Lab, by MUR through the PRIN project 2022XERWK9 S-PIC4CHU, and by the EU and MUR through the PNRR project PE0000013-FAIR.

References

1. Al-Kateb, M., Mansour, E., El-Sharkawi, M.E.: CME: A temporal relational model for efficient coalescing. In: Proc. of the 12th Int. Symp. on Temporal Representation and Reasoning (TIME). pp. 83–90. IEEE Computer Society (2005)
2. Arenas, M., Bahamondes, P., Aghasadeghi, A., Stoyanovich, J.: Temporal regular path queries. In: Proc. of the 38th IEEE Int. Conf. on Data Engineering (ICDE). pp. 2412–2425. IEEE Computer Society (2022)
3. Arenas, M., Bahamondes, P., Stoyanovich, J.: Temporal regular path queries: Syntax, semantics, and complexity. CoRR Technical Report arXiv:2107.01241, arXiv.org e-Print archive (2021), https://arxiv.org/abs/2107.01241, available at https://arxiv.org/abs/2107.01241
4. Aupperle, L., Conn, H.E., Keil, J.M., O'Rourke, J.: Covering Orthogonal Polygons with Squares. Johns Hopkins University, Department of Computer Science (1988)
5. Böhlen, M.H., Dignös, A., Gamper, J., Jensen, C.S.: Temporal data management – An overview. In: Tutorial Lectures of the 7th European Summer School on Business Intelligence and Big Data (eBISS). Lecture Notes in Business Information Processing, vol. 324, pp. 51–83. Springer (2017)
6. Böhlen, M.H., Snodgrass, R.T., Soo, M.D.: Coalescing in temporal databases. In: Proc. of the 22nd Int. Conf. on Very Large Data Bases (VLDB). pp. 180–191 (1996)
7. Bouros, P., Mamoulis, N., Tsitsigkos, D., Terrovitis, M.: In-memory interval joins. Very Large Database J. **30**(4), 667–691 (2021)
8. Cafagna, F., Böhlen, M.H.: Disjoint interval partitioning. Very Large Database J. **26**(3), 447–466 (2017)
9. Culberson, J.C., Reckhow, R.A.: Covering polygons is hard. J. of Algorithms **17**(1), 2–44 (1994)
10. Dignös, A., Böhlen, M.H., Gamper, J.: Overlap interval partition join. In: Proc. of the 35th ACM Int. Conf. on Management of Data (SIGMOD). pp. 1459–1470 (2014)
11. Dignös, A., Böhlen, M.H., Gamper, J., Jensen, C.S., Moser, P.: Leveraging range joins for the computation of overlap joins. Very Large Database J. **31**(1), 75–99 (2022)
12. Dignös, A., Glavic, B., Niu, X., Gamper, J., Böhlen, M.H.: Snapshot semantics for temporal multiset relations. Proc. of the VLDB Endowment **12**(6), 639–652 (2019)
13. Enderle, J., Hampel, M., Seidl, T.: Joining interval data in relational databases. In: Proc. of the 25th ACM Int. Conf. on Management of Data (SIGMOD). pp. 683–694 (2004)
14. Eppstein, D.: Graph-theoretic solutions to computational geometry problems. In: Revised Papers the 35th Int. Workshop on Graph-Theoretic Concepts in Computer Science (WG). Lecture Notes in Computer Science, vol. 5911, pp. 1–16 (2009)
15. Francis, N., Green, A., Guagliardo, P., Libkin, L., Lindaaker, T., Marsault, V., Plantikow, S., Rydberg, M., Selmer, P., Taylor, A.: Cypher: An evolving query language for property graphs. In: Proc. of the 39th ACM Int. Conf. on Management of Data (SIGMOD). pp. 1433–1445 (2018)
16. Gao, D., Jensen, C.S., Snodgrass, R.T., Soo, M.D.: Join operations in temporal databases. Very Large Database J. **14**(1), 2–29 (2005)
17. Harris, S., Seaborne, A.: SPARQL 1.1 query language. W3C Recommendation, World Wide Web Consortium (Mar 2013), available at http://www.w3.org/TR/sparql11-query

18. Jensen, C.S., Snodgrass, R.T.: Bitemporal relation. In: Encyclopedia of Database Systems. Springer, 2nd edn. (2018)
19. Kaufmann, M., Manjili, A.A., Vagenas, P., Fischer, P.M., Kossmann, D., Färber, F., May, N.: Timeline index: a unified data structure for processing queries on temporal data in SAP HANA. In: Proc. of the 34th ACM Int. Conf. on Management of Data (SIGMOD). pp. 1173–1184 (2013)
20. Keil, J.M.: Minimally covering a horizontally convex orthogonal polygon. In: Proc. of the 2nd Annual ACM SIGACT/SIGGRAPH Symposium on Computational Geometry (SCG). pp. 43–51 (1986)
21. Piatov, D., Helmer, S., Dignös, A.: An interval join optimized for modern hardware. In: Proc. of the 32th IEEE Int. Conf. on Data Engineering (ICDE). pp. 1098–1109. IEEE Computer Society (2016)
22. Toman, D.: Point-based temporal extensions of SQL and their efficient implementation. In: Temporal Databases, Dagstuhl. Lecture Notes in Computer Science, vol. 1399, pp. 211–237. Springer (1997)
23. Zhou, X., Wang, F., Zaniolo, C.: Efficient temporal coalescing query support in relational database systems. In: Proc. of the 17th Int. Conf. on Database and Expert Systems Applications (DEXA). Lecture Notes in Computer Science, vol. 4080, pp. 676–686. Springer (2006)