Language Resources and Evaluation (2026) 60:10
https://doi.org/10.1007/s10579-025-09883-8

ORIGINAL PAPER

TLEX: an efficient method for extracting exact timelines from TimeML temporal graphs

Mustafa Ocal[1] · **Ning Xie**[2] · **Mark A. Finlayson**[3]

Received: 14 April 2025 / Accepted: 16 November 2025 / Published online: 6 January 2026

Abstract

A timeline provides a total ordering of events and times, and is useful for a number of natural language understanding tasks. However, qualitative temporal graphs that can be derived directly from text—such as TimeML annotations—usually explicitly reveal only *partial* orderings of events and times. In this work, we apply prior work on solving point algebra problems to the task of extracting timelines from TimeML annotated texts, and develop an exact, end-to-end solution which we call TLEX (TimeLine EXtraction). TLEX transforms TimeML annotations into a collection of timelines arranged in a trunk-and-branch structure. Like what has been done in prior work, TLEX checks the consistency of the temporal graph and solves it; however, it adds two novel functionalities. First, it identifies specific relations involved in an inconsistency (which could then be manually corrected) and, second, TLEX performs a novel identification of sections of the timelines that have indeterminate order, information critical for downstream tasks such as aligning events from different timelines. We provide detailed descriptions and analysis of the algorithmic components in TLEX, and conduct experimental evaluations by applying TLEX to 385 TimeML annotated texts from four corpora. We show that 123 of the texts are inconsistent, 181 of them have more than one "real world" or main timeline, and there are 2,541 indeterminate sections across all four corpora. A sampling evaluation showed that TLEX is 98–100% accurate with 95% confidence along five dimensions: the ordering of time-points, the number of main timelines, the placement of time-points on main versus subordinate timelines, the connecting point of branch timelines, and the location of the indeterminate sections. We provide the extracted timelines for all texts, the manual corrections to the temporal graphs of the inconsistent texts, as well as code to reproduce our experiments.

Keywords Timeline · TimeML · Temporal reasoning · Temporal information extraction

Ning Xie and Mark A. Finlayson contributed equally to this work.

Extended author information available on the last page of the article

1 Introduction

A timeline is a data structure that organizes events and times in a total ordering. Timelines are useful for a number of natural language understanding tasks, such as **question answering**, which may require understanding the overall temporal order of events to produce the correct answer (Saquete et al., 2004); **cross-document event co-reference (CDEC)** and **cross-document alignment**, whose accuracy can be improved by access to a total ordering of events (Navarro-Colorado & Saquete, 2016); and **summarization** and **visualization** (Liu et al., 2012), where timelines can enhance human understanding of the events as well as the temporal structure of the texts.

Unfortunately, timelines are rarely explicit in text, and usually cannot be extracted directly. Instead, texts usually explicitly reveal only *partial* orderings of events and times. Such partial information can be used—either through automatic analyzers (Verhagen et al., 2005), or manual annotation (Pustejovsky et al., 2003a), or some combination of both—to construct a temporal graph labeled in some temporal representation language, such as a temporal algebra (Allen, 1983) or TimeML (Sauri et al., 2006).

For temporal graphs labeled with a temporal algebra (Barták et al., 2014), prior work in the field of *Qualitative Spatial and Temporal Reasoning* (QSTR) has shown how to extract an exact timeline (Gereveni & Schubert, 1995; Kreutzmann & Wolter, 2014; van Beek, 1992) that represents one possible solution of total ordering. However, this approach does not work for natural language involving temporal information that cannot be encoded in a strictly temporal algebra (i.e., expressions of possible, counterfactual, or conditional worlds—called *subordinated* events here), which is the reason why schemes such as TimeML were developed. Also, natural language is often ambiguous, hence it is useful to know which portions of a timeline are *indeterminate*—that is, have multiple possible orderings all consistent with the temporal graph.

There has been some prior work in the field of Natural Language Processing (NLP) on extracting timelines from TimeML graphs using machine learning, but these solutions fall short in dealing with all possible relations and can result in outputs that contain ordering errors (see Sect. 2.3). These errors are a natural consequence of using statistical approaches to solve the problem.

In this work, we apply the theoretical formalism and techniques from QSTR to TimeML annotated texts, formulate timeline extraction as a *topological sort problem*, and show that one can achieve a theoretically *exact* solution, unlike prior NLP approaches. Furthermore, our framework handles all possible temporal relations (including aspectual and subordination relations) and detects indeterminacy. Specifically, we present TLEX (TimeLine EXtraction), a method for extracting a set of timelines from a TimeML annotated text, which handles all TimeML relations and achieves perfect accuracy modulo the correctness of an underlying TimeML graph. TLEX outputs multiple timelines, one for each temporally connected subgraph, including *subordinated timelines*, which represent possible (modal), counterfactual, or conditional situations—essentially describing "possible worlds" rather than events that occurred in the real world. For example, someone recounting their day might mention

forgetting to go to the supermarket, and if they had gone, they could have bought some milk. While there is a temporal relationship between the *going* and *buying* events, they did not actually happen in reality. These subordinated events in subordinated timelines, which we discuss in detail in Sect. 2.2.2, are connected to the main timeline in a trunk-and-branch timeline structure,[1] Like prior work, TLEX checks the TimeML graph for consistency and performs topological sorting to produce a timeline. However, unlike prior approaches, TLEX extends inconsistency detection to include ALINKs and systematically identifies indeterminate sections of timelines, providing more comprehensive insights into temporal structures and supporting manual corrections.

Ensuring total consistency in temporal graphs is critical for practical applications. Without full consistency, a global timeline cannot be extracted. Inconsistent temporal constraints create logical contradictions that prevent establishing any valid event ordering. Moreover, consistency checking serves as quality assurance for temporal annotations, as inconsistencies typically indicate annotation errors requiring correction. The importance of temporal consistency extends to numerous downstream tasks. In question answering systems, inconsistent temporal information leads to contradictory answers about event sequences. For cross-document event coreference and narrative understanding, consistent timelines are essential for accurately merging events and maintaining coherent plot structures. Temporal consistency analysis also has forensic applications. For instance, detecting temporally inconsistent testimony that might indicate deception, or identifying contradictions in witness statements and legal documents that warrant further investigation. In clinical settings, inconsistent medical timelines could reveal documentation errors affecting patient care decisions.

We provide a formal proof of TLEX 's correctness, as well as experimental evaluations using 385 manually annotated texts comprising 129,860 words across four corpora. This revealed 123 inconsistent temporal graphs, for which we provide manual corrections. We also checked five different features of the extracted timelines using a sampling evaluation. Our measurements show that TLEX achieves an accuracy of 98–100% with 95% confidence on all these measurements. In prior work we have released reference implementations of the TLEX algorithm in both Python and Java (Ocal et al., 2023; Singh et al., 2024). Using that code, we further provide (a) the timelines extracted from the corpora, (b) corrections to the corpora, and (c) code to reproduce our experiments.[2]

There are four main contributions of this work. **First**, we describe the TLEX algorithm in detail, providing a clear explanation of how it works, a formal proof of its correctness, and a reference implementation. **Second**, we introduce a new indeterminacy marked trunk-and-branch timeline structure as TLEX 's final output. This structure separates main timelines (the "real world" or the trunks) from subordinated timelines (the "possible worlds" or the branches). This explicit recognition of the

[1] It is worth mentioning that, although TLEX is the first system that converts a TimeML graph into trunk-and-branch timeline structure, the idea of separating subordinated events in annotations was previously introduced in the multi-axis annotation scheme model (Ning et al., 2018) in which the authors defined different axis for different types of events in the annotations such as intention/opinion events (on an orthogonal axis), generic events (on a parallel axis), negations (not on an axis), and static events (other axis).

[2] https://cognac.cs.fiu.edu/jtlex/.

separation between "real world" and other types of events is important in the semantics of natural language and has not been dealt with in TimeML-to-timeline conversion work before. The identification of indeterminacy in timelines is also novel. **Third**, we demonstrate a new technique for identifying the specific temporal links that contribute to temporal inconsistency, which allows targeted manual correction of inconsistent TimeML annotations. **Last**, we experimentally evaluate TLEX on four TimeML corpora. Through a sampling evaluation, TLEX 's performance is found to be accurate within error bounds, confirming our theoretical results.

The rest of this paper is organized as follows. First, we discuss related work, which we build upon and extend (Sect. 2). Second, we provide a detailed description of TLEX, breaking the procedure into five main steps, and present proof of correctness and time complexity analysis for each step (Sect. 3). We then describe our experimental evaluations and results (Sect. 4). We conclude with a discussion of possible future work (Sect. 5) and a list of the contributions (Sect. 6).

2 Background and related work

Temporal reasoning is a fundamental aspect of natural language understanding, enabling systems to infer event orderings, durations, and dependencies within a given text. Various formal frameworks have been developed to model temporal relationships, ranging from qualitative approaches like Allen's interval algebra to hybrid annotation schemes such as TimeML. While these frameworks provide structured ways to represent temporal information, extracting a coherent timeline from annotated texts remains a challenge due to inconsistencies, ambiguities, and gaps in temporal graphs. This section provides an overview of the key theoretical foundations and prior work relevant to timeline extraction. We begin with a discussion of temporal algebras and reasoning (Sect. 2.1), followed by a detailed examination of TimeML and its role in temporal information extraction (Sect. 2.2). Finally, we review previous approaches to timeline extraction (Sect. 2.3) and highlight the limitations that motivate the development of TLEX.

2.1 Temporal algebras and reasoning

2.1.1 Interval algebra

Allen's interval algebra (Allen, 1983) is a calculus for temporal reasoning that defines possible relations between a pair of time intervals and the inference rules for relation compositions. Allen's algebra was one of the first attempts to computationally model temporal knowledge and temporal reasoning. In this formalism, every event is abstracted as a time interval I, which comprises a start point (I^-) and end timepoint (I^+), where the start point comes strictly before the end point ($I^- < I^+$). Two intervals can be related by a set of 13 mutually exclusive *basic* temporal relations: *before*, *meets*, *overlaps*, *starts*, *during*, and *finishes*, their inverses, and *equal*. Allen's interval algebra allows any subset of these 13 basic relations to form a general relation (hence there are in total 2^{13} relations); two intervals A and B can be related by a

basic relation as *A before B*, or by a more complex relation such as *A {before, during, starts} B* when we are less certain about their temporal relation. Allen presented a composition table for composing pairs of temporal relations, which can be used to infer the temporal relation between intervals *A* and *C* given the relation between *A* and *B* and the relation between *B* and *C*. A fundamental question in Allen's algebra is the following *satisfiability* problem: given *n* interval $I_1, \ldots, I_n$ and *m* pairwise relations among these intervals, does there exist a *configuration of intervals*[3] that satisfies all the relations? Unfortunately, it was shown that the satisfiability problem is NP-complete (Vilain & Kautz, 1986; Vilain et al., 1990).

Using Allen's algebra, we can construct a *temporal graph* from texts by viewing events and times as intervals and representing them as the vertices of the graph, and translating the relations between two events/times expressed in natural language into Allen's temporal relations and representing them as the labels of the edges in the graph. Allen's approach is referred to as a *qualitative* temporal algebra because it does not represent exact times or durations. A special type of temporal graph called an *atomic* temporal graph, in which all edge labels are basic relations, is of particular importance for our purposes.

In a great deal of later work, Allen's qualitative framework was generalized and extended to the quantitative case, where time-points and durations are given specific metric values. Researchers have proved quite a number of formal results concerning both qualitative and quantitative formalisms (reviewed in Barták et al., 2014). Of particular importance are those results related to checking the *consistency* of temporal graphs, because a timeline can only be constructed for consistent temporal graphs. To check the consistency of graphs constructed using his algebra, Allen developed a polynomial time, constraint-propagation type algorithm that repeatedly uses the composition table to reconcile the relation between every pair of vertices, to ensure that their relation is consistent with those relations imposed by all length-two paths between them (so-called *path consistency*) (Allen, 1983). However, the algorithm does not detect all inconsistencies, and it was later shown that checking the consistency of general temporal graphs with all Allen relations is NP-complete (Vilain & Kautz, 1986; Vilain et al., 1990), which makes the task intractable under the commonly believed $\mathcal{P} \neq \mathcal{NP}$ assumption.

Because the full power of Allen's interval algebra is likely to be intractable, many researchers introduced tractable subclasses of Allen's algebra (Freksa, 1992; Nebel & Bürckert, 1995; Vilain et al., 1990). In a celebrated paper (Nebel & Bürckert, 1995), a subclass called ORD-HORN is shown to be the unique maximal tractable subclass of the full Allen algebra: its consistency can be checked in polynomial time by the path-consistency algorithm while checking any subclass that is greater than the ORD-HORN subclass is NP-complete.

[3] A *configuration* for a finite set of intervals $\mathcal{I} = \{I_1, \ldots, I_n\}$ is a set of *n* bounded and closed intervals (Ligozat, 2013, p. 9). More formally, a configuration is a function $C : \mathcal{I} \to \mathbb{R} \times \mathbb{R}$ satisfying the property that, for every $1 \leq i \leq n$, if $C(I_i) = \langle a_i, b_i \rangle$ (i.e., if interval I_i is mapped to the closed interval $[a_i, b_i]$) then $-\infty < a_i \leq b_i < \infty$.

2.1.2 Point algebra

Introduced by Vilain and Kautz (1986), the point algebra (PA) is a symbolic calculus intended to work with qualitative ordering constraints between pairs of time-points; for systematic treatments, see Barták et al. (2014) and references therein. PA defines three basic temporal constraints between two time-points: <, >, and =. If there is no information between two time-points *a* and *b*, then we say *a* {<, >, =} *b*. Temporal reasoning in PA is much more efficient since it has only three basic relations. For instance, the CSPAN algorithm checks the consistency of a PA temporal graph in $O(n^2)$ time, where *n* is the number of time-points in the graph (van Beek, 1992).

An interval sub-algebra of Allen's algebra is called *pointisable* if its relations can be equivalently defined by a PA on the start and end points of the intervals involved (Leeuwenberg & Moens, 2019). For example, it is easy to check that *A before B* is equivalent to the constraints imposed by the PA temporal graph in which A^-, A^+, B^- and B^+ are the nodes and each of the 12 edges is labeled with an appropriate basic < or > relation (according to the ordering $A^- < A^+ < B^- < B^+$). Moreover, it can be easily verified that all basic relations are pointisable. However, one can also show that *A {before, before_inverse} B* cannot be represented as PA constraints on the same set of 4 time-points. In fact, it is known that there are only 167 pointisable relations among all $2^{13} - 1$ non-null relations in Allen's algebra. The translation of the basic Allen's algebra relations into PA relations is shown in Table 1.

Table 1 Translation of the Allen's Algebra relation and TimeML temporal and aspectual relations into basic temporal relations between interval start and end points

Allen's algebra	TimeML	Point algebra[a]
A Before B	A before B	$(A^+ < B^-)$
A After B	A after B	$(B^+ < A^-)$
A Meets B	A ibefore B	$(A^+ = B^-)$
A MetBy B	A iafter B	$(B^+ = A^-)$
A Starts B	A begins B	$(A^- = B^-) \wedge (A^+ < B^+)$
A StartedBy B	A begun_by B	$(A^- = B^-) \wedge (B^+ < A^+)$
A Finishes B	A ends B	$(B^- < A^-) \wedge (A^+ = B^+)$
A FinishedBy B	A ended_by B	$(A^- < B^-) \wedge (A^+ = B^+)$
A During B	A includes B	$(A^- < B^-) \wedge (B^+ < A^+)$
A DuringInverse B	A is_included B	$(B^- < A^-) \wedge (A^+ < B^+)$
A Equals B	A simultaneous B	$(A^- = B^-) \wedge (A^+ = B^+)$
	A identity B	$(A^- = B^-) \wedge (A^+ = B^+)$
	A during B	$(B^- = A^-) \wedge (A^+ = B^+)$
	A during_inv B	$(A^- = B^-) \wedge (B^+ = A^+)$
	A initiates B	Same as A begins B
	A culminates B	Same as A ends B
	A terminates B	Same as A ends B
	A continues B	Same as A is_included B
	A reinitiates B	Same as A is_included B

For an interval *I* (an event or a time), the start point of the interval is denoted by I^- and the end point is denoted by I^+

[a] Here we omit the trivial relations $I^- < I^+$ for all intervals *I* involved to save space

Allen's interval algebra and point algebra (PA) are actually subclasses of a more general problem called *Temporal Constraint Satisfaction Problems* (TCSPs) (Dechter et al., 1991). In a TCSP, time-points are represented as a set of variables, and temporal information is represented by a set of unary or binary constraints over these variables, each specifying a set of permitted intervals. A special case of TCSP, *Simple Temporal Problem* (STP), was also proposed and studied in Dechter et al. (1991). The input to an STP is a set of time-point variables together with constraints that bound the difference between pairs of these variables, and the output is required to be a succinct representation of the set of all possible assignments to time-point variables that meet those constraints. Because STP is solvable in polynomial time and at the same time captures much of the characteristics of temporal constraint problems in real life, it has been extensively studied and applied in a wide range of fields, such as planning, scheduling, robotics, and control theory. For more background on STP and recent extensions, interested readers are referred to (Ostuni et al., 2021).

2.2 Temporal information extraction and annotation in text

2.2.1 TIDES

Because of the utility of temporal frameworks for reasoning about time, and also the relevance of time to understanding natural language, researchers in Natural Language Processing (NLP) have sought to apply temporal algebras to text understanding. This requires annotation schemes that would allow a person or a machine to annotate a text for the time-points, events, and temporal relations expressed.

Time expressions (**TIMEXes**) are sequences of tokens (words, numbers, or characters) in text that denote time, including expressions of when something happens, how often something occurs, or how long something takes. Starting from the early 2000s, researchers developed a sequence of **TIMEX** annotation schemes (Ferro et al., 2001; Pustejovsky et al., 2003b; Setzer, 2001). This allows the annotation of expressions such as *at 3 p.m.* (when), or *for 1 h* (how long). Because events are also involved in temporal relations, these approaches were extended into schemes for capturing both times and events. For example, the Translingual Information Detection, Extraction, and Summarization scheme (TIDES) (Ferro et al., 2001) integrates TIMEX2 expressions as well as a scheme for annotating events. TIDES includes annotations for temporal expressions, events, and temporal relations. TIDES uses only six temporal relation types to represent the relations between events, therefore it provides only limited temporal information from texts.

2.2.2 TimeML

Deficiencies in TIDES led to the development of TimeML (Sauri et al., 2006), another markup language for annotating temporal information, originally designed for news. As shown in Table 1, TimeML contains all Allen's basic temporal relations. TimeML does not have Allen's OVERLAPS and OVERLAPPED_BY relations directly, however, these relations can be expressed in the temporal representation of a TimeML document by combinations of other relations (Llorens et al., 2015). TimeML adds addi-

tional representational facilities. First, Allen's EQUALS relation indicates two intervals have the same start and end time. In contrast, TimeML offers more refined relations: SIMULTANEOUS represents two events that happened simultaneously, DURING represents an event persists throughout a temporal expression (duration), and IDENTITY represents event coreference relation between two events. Furthermore, TimeML includes relations for expressing sub-event structure (aspectual relations) and relations of conditional, hypothetical, or counterfactual nature (subordinating relations). Because the input to TLEX is a TimeML graph, we give here a detailed exposition of the representational structure of TimeML.

TimeML uses TIMEX3 expressions, which introduce additional attributes and features to capture more complex temporal expressions accurately. These features include quantifiers ("quant"), frequencies ("freq"), anchoring for relative expressions ("anchorTimeID"), and comments for additional information. While TIMEX3 retains the three basic types of temporal expressions from TIMEX - dates (DATE), times (TIME), and durations (DURATION) - it also introduces a new type, "SET," to represent recurring events (e.g., *twice a week*).

TimeML has three different types of links: temporal (TLINK), subordinating (SLINK), and aspectual (ALINK). TLINKs comprise 14 different types of temporal relations between events and times. Figure 1 graphically shows all types of TLINKs in TimeML. The following example represents one of the TLINKs, called IS_INCLUDED. This TLINK represents that the event (*went*) IS_INCLUDED in (*on*) the time (*Monday*).

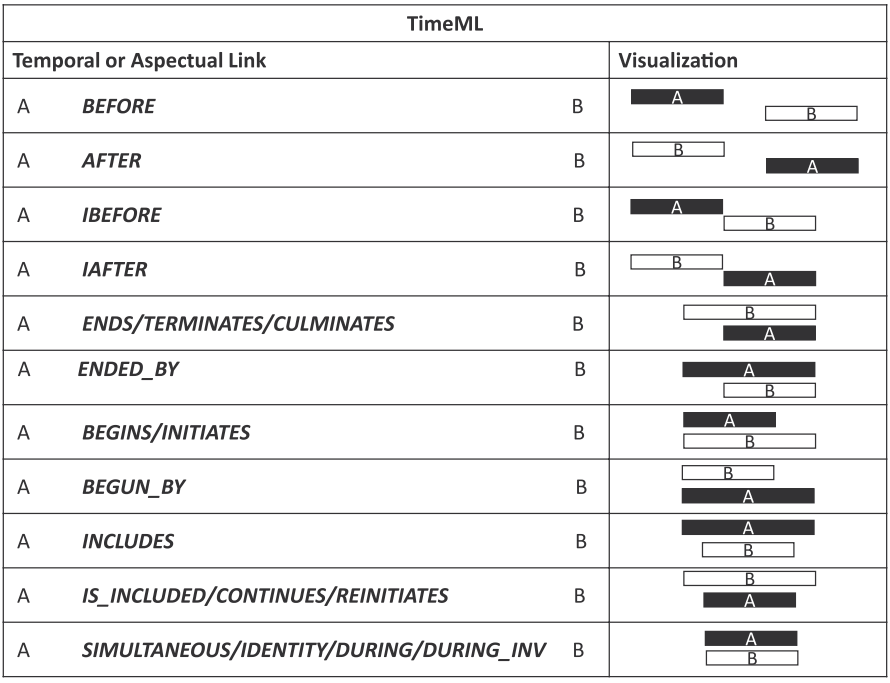

Fig. 1 The 14 temporal and 5 aspectual TimeML link types. Temporal links can hold between two events, an event and a time, and two times

The visual representation of this TLINK and the other examples can be seen in Fig. 2. 1 is the first node of the link, 2 is the second node of the link, and *X* is the TimeML link type.

(1) David **went**$_1$ to the school on **Monday**$_2$. (IS_INCLUDED) ALINKs represent the relation between an aspectual event and its argument event. There are five types of ALINKs (shown in Fig. 1): INITIATES, REINITIATES, TERMINATES, CULMINATES, and CONTINUES. The following example shows the aspectual relation between events, where the first event (*started*) INITIATES the second event (*study*).
(2) Mike **started**$_1$ to **study**$_2$. (INITIATES) Note that some ALINKs and TLINKs may have the same temporal constraints (Table 1), but semantically they are different in TimeML. For example, INITIATES and BEGINS have the same temporal constraints, however, INITIATES happen between an aspectual event and its argument event while BEGINS does not require an aspectual event. SLINKs are used for contexts introducing possible (modal), counterfactual, or conditional relations between two events, spanning six types: MODAL, FACTIVE, COUNTER_FACTIVE, EVIDENTIAL, NEGATIVE_EVIDENTIAL, and CONDITIONAL. The MODAL relation introduces a relation to a possible event. In the following example, the event (*buy*) is merely a possibility and has not actually happened yet.
(3) Cindy **promised**$_1$ him to **buy**$_1$ some nachos. (MODAL) The FACTIVE relation marks an entailment of an event's veracity. On the other hand, a COUNTER_FACTIVE relation marks a presupposition about the event's non-veracity. In other words, FACTIVE indicates a presupposition as to whether the event happened in the real world and COUNTER_FACTIVE indicates a presupposition as to whether the event did not happen in the real world. The following examples demonstrate the difference between these two SLINKs.
(4) Katy **forgot**$_1$ that she **was**$_1$ in Miami last year. (FACTIVE)
(5) Katy **forgot**$_1$ to **buy**$_1$ some chocolate. (COUNTER_FACTIVE) EVIDENTIAL relations are introduced by reporting events asserting that the argument event happened. Similarly, NEGATIVE_EVIDENTIAL relations are introduced by reporting events asserting that the argument event did not happen. The following examples show these two SLINK types.
(6) Colin **said**$_1$ he **went**$_2$ to the store. (EVIDENTIAL)
(7) Darius **denied**$_1$ that he **has coronavirus**$_2$. (NEGATIVE_EVIDENTIAL) Finally, the CONDITIONAL relation identifies two events linked in a conditional manner, for example, with a signal such as "if".
(8) If Sydney **marries**$_1$ him, she will **be happy**$_2$. (CONDITIONAL)

Language introducing an SLINKs does not always entail a temporal relation, but if it does, a TLINK representing that relation should be added to the graph as well. Furthermore, three types of SLINKs explicitly indicate that the event is presumed to not have

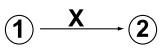

Fig. 2 The visual representation of a generic link of TimeML: node 1 and node 2 represent two intervals, and *X* represent the TimeML link type between the two intervals

happened in the "real world" of the text. Therefore, ignoring subordinating relations or treating them as normal temporal relations gives an incorrect and impaired view of the temporal structure of the text.

Out of the three types of TimeML links, ALINKs and TLINKs represent temporal information in documents, while SLINKs indicate non-temporal relations. On the other hand, SLINKs and ALINKs can only connect two events, while TLINKs can relate two events, two times, or an event and a time.

A well-formed TimeML graph will have at most a single TLINK and a single ALINK relation between two nodes. This is a consequence of the TimeML rule, which says that between two event instances (or between an event instance and a time expression), there cannot be two or more TLINK or ALINK types. If there is both a TLINK relation and an ALINK relation, they must be temporally consistent (Pustejovsky et al., 2003a, 2003b).[4] As a consequence, TimeML graphs are atomic temporal graphs, hence a member of the tractable subclass of the interval algebra (Valdes-Perez, 1986, 1987) (atomic graphs are called "singleton networks" in these papers). Another way to see this is by combining the fact that atomic temporal graphs can be equivalently transformed (in linear time) into their corresponding PA temporal graphs, and that consistency of PA temporal graphs can be checked in $O(n^2)$ time (van Beek, 1992).

2.2.3 TimeML annotations

There are a number of manually and automatically annotated TimeML corpora. In this work, we tested our approach on four manually annotated TimeML corpora: TimeBank 1.2, the N2 corpus, the ProppLearner corpus, and the NewsReader MEANTIME corpus, all in English. The number of texts, words, events, temporal expressions, and relations are listed in Table 2. TimeBank 1.2 is drawn from various sources such as ABC, CNN, and the Wall Street Journal (Pustejovsky et al., 2003a). The ProppLearner corpus was developed to enable the machine learning of Vladimir Propp's morphology of Russian hero tales and has 18 different layers of syntax and semantics annotated on it, including TimeML (Finlayson, 2017). Similarly, the N2 corpus is a collection of narratives relating to Islamic Extremism with 14 layers of annotation, including TimeML (Finlayson et al., 2014). Finally, the NewsReader MEANTIME corpus is a semantically annotated corpus of English Wikinews articles (Minard et al., 2016) (the corpus also includes TimeML annotations of article translations in Italian, Spanish and Dutch), covering four news topics: "Airbus and Boeing", "Apple", "the Stock market" and "General Motors, Chrysler and Ford."

Table 2 Summary of the corpora used in the experiments

Corpus name	Texts	Words	Events	TIMEXs	Links	Text types
ProppLearner	15	18,862	3,438	142	2,778	Russian Hero tales
N2 Corpus	67	28,462	2,345	349	4,854	Religious texts, magazine
MEANTIME	120	13,981	2,096	525	1,717	WikiNews articles
TimeBank	183	68,555	7,935	1,414	9,615	Newswire, biography, etc
Total	385	129,860	15,814	2,430	18,964	

[4] As can be seen in Fig. 4, there's at most a single ALINK and a single TLINK between every node.

Automatic TimeML annotation can be done via temporal information extraction systems for TimeML temporal expressions, events, and temporal relations (Bethard, 2013; Chambers et al., 2014; Mirza & Tonelli, 2016; Verhagen et al., 2005), or via temporal dependency parsers (Zhang & Xue, 2018). While not perfect, these automatic TimeML annotators attain a reasonable, even good, level of performance under some conditions. Because our goal in this paper is to present details of the TLEX method for extracting timelines in an exact manner, we leave as future work the problem of applying our approach on automatically generated TimeML graphs.[5]

2.3 Prior approaches to timeline extraction

The goal of timeline extraction under *quantitative* (i.e., metric) frameworks is to produce timelines for temporal networks with metric temporal information for all times, events, and relations (i.e., numerical durations of nodes and edges in the temporal graph). Note that a solution for a quantitative framework does not immediately extend to a solution for qualitative cases such as Allen's algebra or TimeML. This is because qualitative frameworks explicitly allow the use of non-metric information in their temporal graphs; indeed, most of the temporal information in natural language is non-metrical.

2.3.1 Qualitative spatial and temporal reasoning approaches

In the field of qualitative spatial and temporal reasoning (QSTR), researchers have given significant attention to timeline extraction. van Beek (1992) presented an approach that transformed Allen's interval algebra temporal graphs into a point algebra network and solved them using backtracking techniques. Gereveni and Schubert (1995) built a "timegraph" given a set of point algebra relations and solved the timegraph using their own algorithm. Later, Wallgrün et al. (2006) presented the *SparQ (Spatial Reasoning done Qualitatively)* tool, which comprises a set of modules to provide different services for qualitative spatial reasoning, which is a similar problem to qualitative temporal reasoning. SparQ transforms a quantitative description of a spatial configuration into a qualitative description, applies the operations in the calculi to spatial relations, and finally performs computations on constraint networks. Similarly, Gantner et al. (2008) built the *Generic Qualitative Reasoner (GQR)* to solve binary qualitative constraint graphs, which takes a temporal calculus description and one or more constraint graphs as input and solves them using path consistency and backtracking. To check the consistency of large qualitative spatial networks, Sioutis and Condotta (2014) presented *Sarissa*, which produces random scale-free-like qualitative spatial networks using the Barabási-Albert (BA) model and uses a hash table-based adjacency list to efficiently represent and reason with them. Finally, Kreutzmann and Wolter (2014) showed how to use AND-OR linear

[5] We could have used LLM-generated annotations; however, in our current work, we tested GPT-3.5 with unseen data, and it had a worse F_1 score on TLINK detection than ML-based systems. Additionally, the generated annotations resulted in highly disconnected graphs, meaning that using LLM annotations would not contribute substantially to our approach.

programming (LP) and mixed integer linear programming (MILP) to solve qualitative graphs, a set that includes graphs represented in Allen's algebra.

2.3.2 NLP approaches

In the NLP field, researchers have explored different methods for timeline extraction in general, and from TimeML-annotated texts in particular. A prominent example of general timeline extraction is the news timeline summarization (TLS) task. Starting from the seminal work of Allan and his collaborators (Allan et al., 2001; Swan & Allan, 2000), timeline summarization focuses on finding specific date mentions in texts and organizing the text or its subpart along a timeline indexed by date. TLS has attracted much attention in the past two decades; for the latest developments, see Ghalandari and Ifrim (2020) and references therein. While this is a useful task for many purposes, it is much different from (and much more granular than) the task we set ourselves here, which is organizing all the events and times mentioned in a text into a (non-metric) timeline. That said, our approach can still contribute to TLS research. Many TLS systems, such as those in Ghalandari and Ifrim (2020), rely on collecting and processing mentions of prior events in news articles under the assumption that the more frequently an event is referenced, the more important it is. Prior work in multiple timeline summarization (e.g., ; Mansouri et al., 2023; Yu et al., 2021) has largely relied on surface-based detection, representation, and clustering of event mentions, whereas our method provides a deeper, structurally informed approach that more accurately captures event orderings and dependencies. By improving event representation and linking through our structured timeline extraction, TLEX could enhance event clustering and ranking for TLS applications.

With regard to approaches that seek to extract timelines from TimeML annotated texts, we can divide these methods into two categories: (1) ML-based approaches that start from raw text, or raw text annotated only with events and time expressions (i.e., no links); and (2) temporal reasoning on fully TimeML annotated document.

ML-based methods

For the first category, researchers build a model to parse documents, predict TLINKs, and infer the interval-based order. Mani et al. (2006) demonstrated a machine learning method to annotate events and partially order them using a qualitative temporal graph generated using Allen's temporal relations. However, this model only predicted three relations—BEFORE, AFTER, and SIMULTANEOUS—and thus excludes large portions of TimeML, and only achieves roughly 75% ordering accuracy. Do et al. (2012) generated the same three relations to obtain a full ordering, achieving a similar accuracy of 73%. In addition to the imperfect performance of these systems, in both cases the methods only consider intervals, rather than start and end points, and so lose much detailed temporal information. Finally, Kolomiyets et al. (2012) presented two models for extracting timelines from qualitative temporal graphs, one based on shift-reduce parsing, and one based on graph parsing. Both approaches take a sequence of event words as input and produce a tree structure. Achieving 70% accuracy, their model generated six temporal relations—BEFORE, AFTER, INCLUDES, IS_INCLUDED, IDENTITY, OVERLAP—again, only a subset of the full

possible set of temporal relations. Similarly, Zhang and Xue (2018) built a parser to extract dependency trees, which can easily be transformed into a timeline. They used LSTM in their pipeline and achieved 76% accuracy. Later, Ocal and Finlayson (2020) showed that temporal dependency trees omit temporal information. Using CNNs and LSTMs, Leeuwenberg and Moens (2020) demonstrated a deep learning model to predict events' start time-point, duration, and end time-point. Their model extracts three types of TLINKs between events (BEFORE, AFTER, and OVERLAPS), achieving 77% accuracy. Mathur et al. (2022) built a temporal parser to extract a temporal dependency graph and maintain the temporal order of the events using contextual features such as BERT. They achieved 77% accuracy on four TLINKs—BEFORE, AFTER, INCLUDES, and OVERLAPS. As can be seen these ML-based approaches have only been applied to partial TimeML annotations (3–6/25 TimeML links) with imperfect performance (70–77% accuracy).

More recently, researchers explored LLM-based approaches to extract timelines from TimeML annotations. Hasegawa et al. (2024) developed TimeSET, a timeline construction evaluation dataset and proposed an evaluation framework comparing LLMs to construct timelines from single documents, finding that NLI formulation with Flan-T5 (3B) Chung et al. (2024) achieved the strongest performance with a median F1 score of approximately 0.5 across prompt templates on three TLINKs—BEFORE, AFTER, and SIMULTANEOUS. In the clinical domain, Wang and Weiss (2025) developed an LLM-based framework for extracting relative timelines from PubMed Open Access case reports, finding that O1-preview achieved moderate event recall (0.80) but high temporal concordance (0.95) on two TLINKs—BEFORE and AFTER. And finally, Wei et al. (2025) introduced a multi-level benchmark for temporal reasoning across 11 fine-grained sub-tasks and 3 real-world scenarios (knowledge-intensive, dynamic news events, and multi-session dialogues), evaluating LLMs on temporal relation types BEFORE, AFTER, and SIMULTANEOUS, with the best-performing system (o3-mini on TIME-LITE) achieving accuracies ranging from 29.90% to 93.33% across different task types, demonstrating that temporal reasoning remains challenging for current LLMs despite test-time scaling improvements.

Temporal reasoning-based methods

In the second category—i.e., temporal reasoning-based methods—Verhagen et al. (2006) introduced Tango, a TimeML parser tool to parse the TimeML annotated documents and create a TimeML graph. Using the TIMEX values, Tango displays the graph in a timeline form, where each section of the timeline contains a TIMEX and all the events connected to the TIMEX, however, it doesn't provide the global order of events. Although Tango checks the consistency of the TimeML graphs using path consistency, it didn't report any inconsistencies for their test corpus (the TimeBank corpus). Similar to Tango, Verhagen (2007) introduced TBOX, a TimeML parser. TBOX also generates a TimeML graph from a TimeML annotation, but it further removes the temporal closure links to display a simplified TimeML graph. TBOX displays each event in a box shape and places each box based on the temporal relation to present the timeline (e.g., if event A is BEFORE event B, then box_A would be on the left of box_B). However, this representation is problematic, considering temporal indeterminacy is already high in TimeML annotations (Verhagen, 2007). Although

these two tools use all 14 types of TLINKs, they ignore SLINKs and ALINKs. Therefore, many events would not be correctly displayed in the timeline structure.

Additionally, Ning et al. (2018) introduced a multi-axis annotation framework aimed at categorizing intention, opinion, hypothetical, generic, and negation events into distinct axes to distinguish them from static and recurring events. This innovative annotation approach led to the development of the MultiAxis Temporal Relations for Start-points (MATRES) corpus.

Summarizing these approaches for timeline extraction, we see that on the one hand, QSTR approaches transform Allen's interval algebra graphs into constraint graphs and solve them using constraint satisfaction techniques. Although these approaches provide a solution for solving qualitative temporal graphs, their methods cannot be applied directly to TimeML graphs because of subordinating relations, and they also do not detect or represent indeterminacy, nor do they help with correcting inconsistencies. ML-based approaches, on the other hand, have been limited by the number of relations considered and provide inexact solutions, presumably because of the noise inherent in a statistical solution. Finally, the TimeML parsers do not provide global orders for all events since they ignore aspectual and subordinating events.

TLEX addresses both of these sets of deficiencies by drawing on both QSTR and NLP to provide a technique for extracting *exact* timelines from *full* TimeML graphs. Unlike prior approaches, TLEX provides a timeline extraction method using *all* temporal relations, including aspectual and subordinating relations. It also exposes indeterminacy in ordering, which is a natural consequence of the ambiguity of natural language, and also clearly identifies the sources of inconsistency to enable manual correction. Expanding Ning et al. (2018)'s comprehensive multi-axis annotation framework, TLEX creates a trunk-and-branch timeline structure that also effectively segregates counter-factive and negative evidential events from real-life events.

3 TLEX algorithms

In this section, we present TLEX, our novel approach for extracting exact timelines from TimeML temporal graphs. Unlike previous methods, TLEX provides a complete, end-to-end solution that ensures theoretical correctness while handling all temporal, aspectual, and subordinating relations. We describe the core algorithmic steps of TLEX, from partitioning the temporal graph into subgraphs to transforming it into a point algebra representation, checking for consistency, generating timelines, and detecting indeterminate sections. Each step preserves the integrity of the original annotations while enabling the construction of a trunk-and-branch timeline structure. Through a combination of constraint satisfaction techniques and formal guarantees, our approach produces an exact solution, unlike prior work, which provided only partial, statistical solutions. The following subsections provide a detailed breakdown of each stage of the algorithm, along with proofs of correctness and complexity analysis.

3.1 Overview of our approach

3.1.1 Problem statement

Recall that our task is to, by taking all possible temporal relations (including aspectual and subordination relations) into consideration, extract from a TimeML annotated text a complete set of timelines that are consistent with the TimeML temporal graph, identify which portions of the timelines are indeterminate, and organize those timelines into a trunk-and-branch structure.

Without loss of generality, we may view the input TimeML temporal graph to TLEX as a connected,[6] *directed* graph $G_{TM} = (V_{TM}, E_{TM})$, where the vertex set V_{TM} are events and time expressions, and there is a directed edge from vertex u to vertex v if there is a TimeML link from u to v (note that there can be more than one link, hence multiple relations corresponding to a single edge of G). That is, $E_{TM} = \{(u, v) : u, v \in V_{TM} \text{ and there is a tlink, slink, or alink from u tov}\}$.[7] Let $n = |V_{TM}|$ be the number of intervals in the input TimeML temporal graph and $m = |E_{TM}|$ be the number of edges of G_{TM}. As G_{TM} is connected, $m \geq n - 1$; and since there can be at most one link of each type between an (ordered) pair of temporal intervals, the number of links in the TimeML temporal graph is at most $3m$.

3.1.2 Structure of the algorithm

TLEX consists of the following five steps:

1. PARTITIONING, in which the input TimeML temporal graph is partitioned into subgraphs connected with only temporal and aspectual links;
2. TRANSFORMING, in which each TimeML temporal subgraph is transformed into a PA constraint subgraph with time-points as vertices and basic temporal relations ($<$, $=$) as edge labels;
3. CONSISTENCY CHECKING, in which the consistency of each PA constraint subgraph is checked, a maximal list of inconsistent cycles is output for manual correction and only consistent PA constraint subgraphs are passed on to the next step;
4. TIMELINEGENERATION, in which the *minimum* timeline for all the time-points in each PA constraint subgraph is generated;
5. INDETERMINACYDETECTION, in which each pair of time-points in the generated timeline can be checked whether there is any ordering indeterminacy,[8] between them.

[6] Because if the TimeML temporal graph is not connected, then we can simply run TLEX on each of its connected components separately.

[7] A simple way to implement this would be to assign a 10-bit vector for each edge as follows. First use **0** vector to denote no relation exists for each of the three types of links. The type of TLINK relation of an edge can thus be represented with $\lceil \log_2(14 + 1) \rceil = 4$ bits; similarly, SLINK relation needs $\lceil \log_2(5 + 1) \rceil = 3$ bits and ALINK relation needs $\lceil \log_2(6 + 1) \rceil = 3$ bits.

[8] More precisely, by *indeterminacy* we mean the possibility of generating some other timeline in which the ordering of the two time-points is reversed.

Figure 3 presents the overall structure of the TLEX algorithm. At the center, it highlights the three main components: Extracting Timeline (combining Steps 1, 2, and 4), Consistency Checking (Step 3), and Indeterminacy Detection (Step 5). On the left, the figure shows the step-by-step process of the Extracting Timeline component: first partitioning the TimeML graph into subgraphs (Step 1), then transforming each subgraph into a PA constraint graph (Step 2), and finally extracting the corresponding timeline (Step 4). The upper right portion illustrates the consistency checking process (Step 3), which ensures that the PA graph is fully consistent; if inconsistencies are found, Step 3 returns the offending constraint links, which can be projected back onto the TimeML graph, with inconsistent links shown in red. The lower right portion depicts Indeterminacy Detection (Step 5), which identifies regions of the graph where the ordering between nodes is indeterminate—for example, where the order between events 2 and 3 cannot be fixed—and shows how this indeterminacy is represented in the resulting timeline.

In the following sections we provide detailed descriptions and analysis of each step, using the following text as a running example:

(9) After working$_1$ in the morning$_2$ John came back$_3$ home. He wished$_4$ that he didn't go$_5$ to work, so he could be happy$_6$. Later, his phone rang$_7$ and he answered$_8$ it. As soon as he picked up$_9$, Paul started$_{10}$ complaining$_{11}$. He was quickly bored$_{12}$, but realized$_{13}$ that if he hung up$_{14}$, Paul would be mad$_{15}$ for the entire weekend$_{16}$. So he continued$_{17}$ to listen$_{18}$. Then he remembered$_{19}$ he ignored$_{20}$ Paul's text in the morning$_2$.

In this example, each event is underlined and given a numerical subscript for reference. One thing notable about this example text is that events 14 and 15 did not happen in the "real world" of the story: because John did not hang up$_{14}$ the phone, Paul didn't get mad$_{15}$. These two events are related to the "real world" timeline by subordinating links and should be isolated onto their own "subordinated" timeline.

We informally define a "main" or "real world" timeline as a timeline that contains the events and times that actually happen in the "world" described in the text. Similarly, we informally define "subordinated" timelines as containing events that did not occur in the world described in the text. As noted, in the example text, the events hang up$_{14}$ and would be mad$_{15}$ belong to a subordinated timeline. It is possible that there are multiple main timelines in a text; we treat this possibility formally in Sect. 4.3.

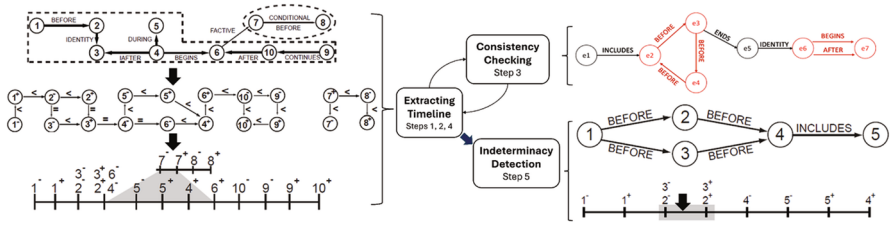

Fig. 3 Overview of the TLEX algorithm, showing the Extracting Timeline component (Steps 1, 2, and 4), Consistency Checking (Step 3), and Indeterminacy Detection (Step 5)

3.1.3 Depth first search (DFS)

We will need the standard depth first search (DFS) algorithm to explore a directed graph and detect cycles in the graph. For completeness, we provide the notation and the pseudo-code of the DFS version presented in Section 20.3 of Cormen et al. (2022), omitting the *time* field as we do not use it.

Specifically, on an input directed graph $G = (V, E)$ stored as an adjacency list, DFS initially colors all vertices as WHITE. When DFS begins to explore the neighborhood of a vertex u (by recursively calling DFS-VISIT on each vertex v which is in u's adjacency list $G.Adj[u]$ and whose color is still WHITE), it changes u's color to GRAY. When DFS finishes examining u's neighbors, it changes u's color to BLACK and the color of u will never be changed again after that. Apart from color, each vertex also has a field π, which is used to store its parent node in the DFS forest constructed in the process of graph exploration. It is known that DFS runs in linear time, namely $O(n + m)$ where $n = |V|$ is the number of vertices and $m = |E|$ is the number of edges of the graph G.

3.2 Partitioning

In this step, TLEX partitions the TimeML graph into subgraphs that are connected only with temporal and aspectual relations. The reason is that temporal and aspectual relations represent temporal information about the text, while subordinating relations do not.

To do this, we first transform the input graph $G_{TM} = (V_{TM}, E_{TM})$ into an *undirected* graph $G'_{TM} = (V_{TM}, E'_{TM})$ by keeping the same vertex set and adding an (undirected) edge (u, v) between two vertices u and v if there is a temporal or aspectual relation from u to v, or a temporal or aspectual relation from v to u. That is, we ignore all SLINK relations and view TLINK and ALINK relations as undirected edges. This can clearly be done in $O(n + m)$ time. Next, we run DFS as illustrated in Algorithm 1 on $G'_{TM} = (V_{TM}, E'_{TM})$ and partition it into connected subgraphs. Since the transformed graph $G'_{TM} = (V_{TM}, E'_{TM})$ has $|E'_{TM}| \leq |E'_{TM}|$, the DFS runs in $O(n + m)$ and hence the total running time of PARTITIONING is also $O(n + m)$. We denote the resulting subgraphs as $G^{(1)}_{TM}, \ldots, G^{(k)}_{TM}$.

Input: directed graph $G = (V, E)$
Output: a DFS forest
1: **for all** $v \in V(G)$ **do**
2: $u.color = \text{WHITE}$
3: $u.\pi = \text{NIL}$
4: **end for**
5: **for all** $v \in V(G)$ **do**
6: **if** $u.color == \text{WHITE}$ **then**
7: DFS-VISIT(G, u)
8: **end if**
9: **end for**

DFS-Visit(G, u)

10: $u.color = \text{GRAY}$
11: **for all** $v \in G.Adj[u]$ **do**
12: **if** $v.color == \text{WHITE}$ **then**
13: $v.\pi = u$
14: DFS-VISIT(G, v)
15: **end if**
16: **end for**
17: $u.color = \text{BLACK}$

Algorithm 1 DFS(G)

Figure 4 illustrates PARTITIONING on the TimeML graph corresponding to Example (9). Subgraphs are separated with dashed lines. Since there are no temporal or aspectual links between event pairs 5 and 6 and 14, 15 and 16 with the rest of the graph (but the graph is otherwise connected), TLEX partitions the graph into three temporally connected subgraphs: the main, or "real world" subgraph (1–2–3–4–7–8-9–10–11–12–13–17–18–19-20) and the subordinated subgraphs (5–6 and 14–15-16). TLEX will subsequently extract a separate timeline from each of these subgraphs.

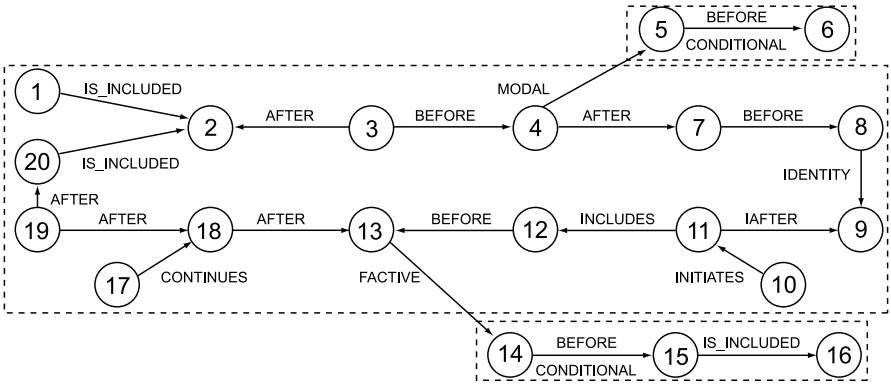

Fig. 4 Visualization of the TimeML graph from the example. Numbers correspond to the events in the text, and arrows correspond to the temporal, aspectual, or subordinating links. The two temporally and aspectually connected subgraphs are separated by dashed lines, and links on the "real world" timeline are bolded. This is the first step of the algorithm, where the temporal graph is initially structured before transformation. It is directly connected to Figs. 5 and 9, which illustrate the subsequent transformation into a PA constraint graph and the final extracted timeline

After the partitioning, TLEX stores the *connecting points* between the main subgraphs and subordinated subgraphs (these points can be easily found by scanning the subordinating links: if a subordinating link (u, v) connecting one vertex from a main subgraph and one vertex from a subordinated subgraph, then both vertices u and v are connecting points). Connecting points will be used later to build the trunk-and-branch timeline. The connecting points in Example (9) are 4 & 5 and 13 & 14.

3.3 Transforming

In TimeML, DATE, DURATION, and SET temporal expressions are regarded as temporal intervals, with starting and ending time points.[9] TIME temporal expressions represent a specific moment in time, such as *9:46 pm* or *1 am*. Unlike intervals, they don't have a duration and simply pinpoint a single time-point in PA (Barták et al., 2014). We transform each temporal *intervals* and time stamps of TimeML subgraph $G_{TM}^{(i)}$ into a point algebra (PA) constraint graph $G_{PA}^{(i)} = (V_{PA}^{(i)}, E_{PA}^{(i)})$, for every $i = 1, \ldots, k$.

Let $G_{TM}^{(i)} = (V_{TM}^{(i)}, E_{TM}^{(i)})$ be any of the interval TimeML subgraphs. Note that we still view $G_{TM}^{(i)}$ as a *directed* graph following the same definition as the original graph G_{TM}; that is, there is a directed edge from vertex u to vertex v in $G_{TM}^{(i)}$ if and only if there is a TLINK relation or an ALINK relation between them.

Recall that each vertex $v \in V^{(i)}$ actually represents a temporal interval I_v, so we first replace each interval v with two time-points, the start time-point t_v^- and the end time-point t_v^+ of the interval I_v. More formally, we let

$$V_{PA}^{(i)} = \{t_v^- : v \in V_{TM}^{(i)}\} \cup \{t_v^+ : v \in V_{TM}^{(i)}\}.$$

Now we construct the edges of $G_{PA}^{(i)}$ (i.e., basic PA constraints) as follows. First, we add the constraint $t_v^- < t_v^+$ for every $v \in V^{(i)}$. Using the fact that each TLINK or ALINK can be represented as a simple conjunction of the basic temporal constraints *less than* (<) and *equals* (=) as shown in Table 1, we next, for each directed edge in $E_{TM}^{(i)}$, add its corresponding basic PA constraints to $E_{PA}^{(i)}$. More formally, if we use $\phi(u, v, t_u^+, t_u^-, t_v^+, t_v^-)$ to denote the set of PA constraints, as specified in Table 1, between $\{t_u^+, t_u^-\}$ and $\{t_v^+, t_v^-\}$ which are imposed by the TimeML constraint of an edge (u, v) in $G_{PA}^{(i)}$, then

$$E_{PA}^{(i)} = \left(\cup_{v \in V_{TM}^{(i)}} \{t_v^- < t_v^+\}\right) \bigcup \left(\cup_{(u,v) \in E_{TM}^{(i)}} \phi(u, v, t_u^+, t_u^-, t_v^+, t_v^-)\right).$$

[9] Note that REINITIATES occurs between a continuous event and a reinitiating trigger word (such as **resuming** or **renewing**). If the event is not continuous, then there will be two different event instances instead of one, and the second event instance will be initiated by the trigger word.

It is easy to check that the blowup of PA constraint graph size is at most a constant factor, namely $|V_{PA}^{(i)}| = 2|V_{TM}^{(i)}|$ and $|E_{PA}^{(i)}| \leq |V_{TM}^{(i)}| + 2|E_{TM}^{(i)}|$. In the following we write

$$n_i := |V_{PA}^{(i)}|$$

for the number of vertices in the i^{th} PA constraint subgraph and

$$m_i := |E_{PA}^{(i)}|$$

for the number of edges in the ith PA constraint subgraph. Note that $\sum_{i=1}^{k} n_i = 2n$ and $\sum_{i=1}^{k} m_i \leq n + 2m$. It follows that TRANSFORMING takes linear time $O(n + m)$ to compute. While certain temporal details like verb tense and specific linguistic signals are lost during the transformation of the TimeML graph to the PA graph, the information pertaining to the global order of events and times remains intact.

Figure 5 illustrates the transformed PA constraint graph from the TimeML temporal graph shown in Fig. 4.

3.4 Consistency checking

Our inconsistency detection process applies to all TLINK and ALINK relations, ensuring that aspectual dependencies are accounted for when checking the coherence of the temporal graph. Unlike previous methods that primarily focused on TLINK consistency, TLEX identifies cycles and contradictions across both temporal and aspectual links. However, our approach does not examine inconsistencies in event mention coreference or missing temporal links, as these require external knowledge beyond the TimeML annotation.

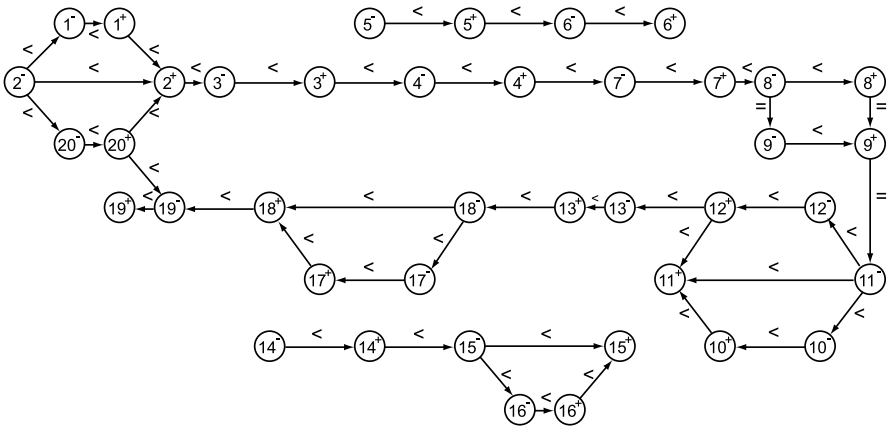

Fig. 5 The three PA constraint subgraphs transformed from the three temporally and aspectually connected subgraphs in Fig. 4. These are produced by replacing each node I with its start and end time-points I^- and I^+, and replacing each temporal or aspectual link with the set of basic temporal relations shown in Table 1

A *timeline* for a PA constraint graph $G_{PA} = (V_{PA}, E_{PA})$ is a function L that maps each time-point in V_{PA} to a (not necessarily unique) point on the real axis such that all time-point constraints imposed by E_{PA} are satisfied. A PA constraint graph $G_{PA} = (V_{PA}, E_{PA})$ is *consistent* if there exists a timeline L for G. Therefore we need to check the consistency of each PA constraint subgraph before passing it on for timeline generation.

It was shown by van Beek (1992) that a PA constraint graph is consistent if and only if it does not have any of the following three types of inconsistent cycles.

Definition 3.4.1 (*Inconsistent Cycle*) An inconsistent cycle is a cycle in a PA constraint graph with one of the following types (for simplicity of notation, we use "$\neq$" as shorthand for the relation "<" or ">" and use "$\leq$" as shorthand for the relation "=" or "<") :

(i) $v = \cdots = w \neq v$;
(ii) $v \leq \cdots \leq w \leq \cdots \leq v \neq w$;
(iii) $v < \cdots < w < \cdots < v$, where all but one of the < can be $\leq$ or = (Note that this case also covers the case of self-loops, $v < v$).

An example of an inconsistent cycle is shown in Fig. 6.

In the third step of TLEX, namely CONSISTENCY CHECKING, given a PA constraint subgraph $G^{(i)}_{PA} = (V^{(i)}_{PA}, E^{(i)}_{PA})$, our task is to check if $G^{(i)}_{PA}$ is consistent; moreover if $G^{(i)}_{PA}$ is inconsistent, output a *Maximal List of Inconsistent Cycles* (MLIC) which includes the set of all inconsistent cycles detected in our checking procedure. This list of TimeML links that correspond to these points can then be provided to human annotators for manual corrections. Although there has been some research in the field of qualitative spatial reasoning focused on automatically correcting inconsistencies in point algebra constraint graphs in a minimum or approximate manner (Condotta et al., 2016; Iwata & Yoshida, 2013), these approaches cannot be applied to TimeML graphs. Inconsistencies in TimeML graphs usually come from incorrect or incomplete annotations; more rarely, there may be fundamental inconsistencies in the meaning of the language. Therefore, errors need to be fixed by reference to the original text. It is possible that one could develop other knowledge- or inference-based methods for automatically correcting these inconsistencies, but we will leave the exploration of this for future work. This means that any inconsistencies discovered by TLEX need to be manually corrected before proceeding. We discuss more specific cases of inconsistencies in selected corpora in Sect. 4.

CONSISTENCY CHECKING comprises the following three parts.

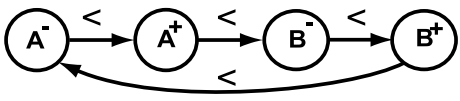

Fig. 6 Example of an inconsistent cycle generated by A BEFORE B and B BEFORE A

3.4.1 Merging equal time-points into "compound" time-points

In this part, we merge time-points that are inter-connected by *equals* (=) relations. Specifically, we keep only the = relations in the PA constraint subgraph and view it as an *undirected* graph. We then run DFS on the modified graph to partition the vertices in $V_{PA}^{(i)}$ into connected components $V_1, \ldots, V_\ell$ (Note that some V_i may contain a single vertex; indeed that should be the case for most V_i's when the PA constraint graph is generated from a real-world text corpus). Time-points in each V_i are inter-connected by = relations and hence they should be equal to each other. We call each such V_i a "compound" time-point. Since running DFS on (a subgraph of) each PA constraint subgraph takes time at most linear in its size, the total running time for processing all PA constraint subgraphs takes $O(m + n)$ time.

In the example, PA constraint graph in Fig. 5, 8^- and 9^- will be merged into one "compound" time-point, $8^+, 9^+$ and 11^- will be merged into another "compound" time-point, and the remaining time-points will be singleton "compound" time-points.

3.4.2 Building "compound" PA constraint graph, detecting type (i) and type (ii) inconsistency

We now build a "compound" PA constraint graph $\tilde{G}_{PA}^{(i)} = (\tilde{V}_{PA}^{(i)}, \tilde{E}_{PA}^{(i)})$: the vertex set $\tilde{V}_{PA}^{(i)}$ is just the set of "compound" time-points created in the previous part, and the labels of the edges will be of *less than* (<) relation only and can be generated by scanning the set of *less than* (<) labeled edges in $E_{PA}^{(i)}$. More specifically, we first associate a new field "compound index" (an integer between 1 and ℓ) to each vertex in $V_{PA}^{(i)}$, indicating to which "compound' time-point the corresponding time point belongs. We then create an $\ell \times \ell$ Boolean-valued table T whose (i, j)-entry stores whether "compound" time-point i is *less than* "compound" time-point j. We now go through all *less than* (<) labeled edges in $E_{PA}^{(i)}$ and update the entries of T accordingly. During this process, we also detect inconsistency cycles as follows:

> *detecting type (i) inconsistency*: If there is a constraint $u < v$ and u, v both belong to the same "compound" time-point, then a type (i) inconsistency cycle is found, and we add all time-points in this "compound" time-point to the MLIC;
> *detecting type (ii) inconsistency*: For every constraint $u < v$ where u belongs to "compound" time-point i and v belongs to "compound" time-point j, we update the (i, j)-entry of T to TRUE. We then check the (j, i)-entry of T and a type (ii) inconsistency cycle is detected if that entry is also TRUE. If so, we add all time-points in both "compound" time-point i and "compound" time-point j to the MLIC.

Note that now the "compound" PA constraint graph has $\ell \leq n_i$ vertices, at most m_i directed edges and all edges are label with *less than* (<) relations. As we need to build the table T, the running time is $O(n^2 + m)$.

3.4.3 Cycle-finding in "compound" PA constraint graph—detecting type (iii) inconsistency

To detect type (iii) inconsistency in $\tilde{G}_{PA}^{(i)} = (\tilde{V}_{PA}^{(i)}, \tilde{E}_{PA}^{(i)})$, we apply the well-known cycle-detection method of DFS. Run the DFS described in Algorithm 1 on $\tilde{G}_{PA}^{(i)}$ with the following slight modifications: if when vertex *u* explores its neighboring vertices in Line 9 and finds that *v*'s color is GRAY, then a cycle is detected. Proof of the correctness of this method is well-known, but for completeness we provide a sketch here. Let $v_1 < v_2 < \cdots < v_p < v_1$ be a cycle in $\tilde{G}_{PA}^{(i)}$ and without loss of generality v_1 is the first vertex to be discovered by DFS. By the White-path Theorem (Cormen et al., 2022, Theorem 20.9), v_p is a descendant of v_1 in the DFS forest generated by the algorithm; it follows that when v_p is discovered in Line 12, the color of v_1 is still GRAY, hence the cycle will surely be detected. On the other hand, if a vertex *u* finds that *v*'s color is GRAY when exploring its neighbors, then *v* is an ancestor of *u*, meaning that there is a directed path from *v* to *u*. But *v* is a neighbor of *u* implies there is an edge from *u* to *v*. Therefore there must be a cycle containing both *u* and *v*.

To detect all possible type (iii) inconsistency cycles, each time a cycle is detected we remove edge (*u*, *v*) from $\tilde{G}_{PA}^{(i)}$ to break the cycle, add the cycle to the MLIC (the cycle can be found by tracing back recursively the parent vertex $u.\pi$ starting from *u* until we reach vertex *v*), and keep running the DFS algorithm till it finishes. The running time of this part is $O(m + n)$

Summary of CONSISTENCY CHECKING.

From our analysis in each of the three parts, the overall running time of CONSISTENCY CHECKING is clearly $O(m + n^2)$. Note that, if the algorithm does not detect any inconsistency in the "compound" PA constraint subgraph $\tilde{G}_{PA}^{(i)}$ and passes it on to next step, then the "compound" PA constraint subgraph satisfies

1. $\tilde{G}_{PA}^{(i)}$ is a directed acyclic graph (DAG);
2. All the constraints are *less than* (<) relations; that is, if (*u*, *v*) is an edge in $\tilde{G}_{PA}^{(i)}$ then for any timeline *L* satisfying $\tilde{G}_{PA}^{(i)}$, $L(u) < L(v)$.

3.5 Timeline generation

3.5.1 Normal form timeline

The two properties listed at the end of last section of the "compound" PA constraint subgraph allow ordering *linearly*, namely topological sort, all the time-points involved. Recall that a *topological sort* of a directed acyclic graph $G = (V, E)$ is a linear ordering of all its vertices such that if *G* contains an edge (*u*, *v*), then *u* appears before *v* in the ordering. However, our goal in TIMELINE GENERATION is slightly different: for every "compound" PA constraint subgraph $\tilde{G}_{PA}^{(i)}$, we want to generate a

shortest timeline for time-points in $\tilde{G}^{(i)}_{PA}$, for which we provide a more formal definition now. We prioritize generating the shortest timeline among potentially multiple solutions to establish a reference point for subsequent indeterminacy calculations. This allows for an equitable distribution of indeterminacy scores across all timeline segments, as detailed in Sect. 3.6. To keep notation simple, in the rest of this section we use a general PA constraint graph $G = (V, E)$ instead of $\tilde{G}^{(i)}_{PA}$ as the input to Timeline Generation.

First, in order to make notation simple and eliminate unnecessary ambiguities, we require the values of a timeline to be positive integers, hence the notion of *normal form timelines*. Let $\mathbb{N} = \{1, 2, \ldots\}$ denote the set of natural numbers.

Definition 3.5.1 (*Timeline and normal form timeline*) Let $G = (V, E)$ be a PA constraint graph. A function $L : V \to \mathbb{R}$, which maps every time-point in V to a real number, is called a *timeline* of $V(G)$ if all the temporal constraints imposed by E are satisfied. A timeline is said to be in *normal form* if the range of the timeline L is further restricted to be $\mathbb{N}$. The *length* of a normal form timeline is the maximum value of the timeline $\ell(L) = \max_{u \in V} L(u)$. A timeline L is called a *minimal normal form timeline* of G if for any normal form timeline L' of $V(G)$, $\ell(L) \leq \ell(L')$. Finally, a timeline L is called a *minimum normal form timeline* if for any normal form timeline L' of $V(G)$ and every vertex u in V, $L(u) \leq L'(u)$.

It follows immediately from the definitions that if a timeline L is a minimum normal form timeline then it is also a minimal normal form timeline. However, the converse is in general not true. The motivation behind our definitions is the following. It is possible that there are multiple normal form timelines of a PA constraint graph that all have the minimum possible length. In order to eliminate such ambiguity, we further require the timeline should map every time-point to its minimum possible value in any consistent timeline. Indeed, one can show that for any PA constraint graph G, the minimum normal form timeline for G is unique[10] and hence it is justified to call a timeline *the* minimum normal form timeline for G.

Consider the example shown in Fig. 7, one possible normal form timeline (after transforming the original Allen algebra on temporal intervals into PA on time-points) L_1 is to map interval I_1 to [1, 2], interval I_2 to [3, 4], interval I_3 to [5, 6], interval I_4 to [7, 8], and interval I_5 to [9, 10], with $\ell(L_1) = 10$. Another normal form timeline

[10] The uniqueness follows by inspecting the Greedy Kahn's algorithm (Algorithm 2) and proving the unique optimality of its output timeline by an induction on time-points that are added to set S at iteration t.

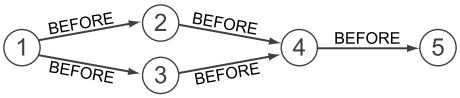

Fig. 7 A TimeML graph with an indeterminacy. While the order between interval I_4 and interval I_5 is fixed, the relative order between interval I_2 and interval I_3 is indeterminate

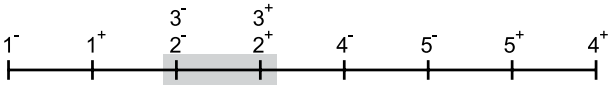

Fig. 8 The minimum timeline extracted from the TimeML graph in Fig. 7, with the indeterminate section highlighted

L_2, as shown in Fig. 8, maps interval I_1 to [1, 2], both interval I_2 and interval I_3 to [3, 4], interval I_4 to [5, 6], and interval I_5 to [7, 8], with $\ell(L_2) = 8$. It is not hard to be convinced that L_2 is the minimum normal form timeline for the original TimeML graph.

TLEX assumes, in the absence of additional information, that the minimum normal form timeline is the best.

3.5.2 Minimum normal form timeline generation algorithm

3.5.2.1 Kahn's topological sort algorithm We first review the classic Kahn's algorithm (Kahn, 1962) for topological sort as it is the basis of our minimum normal form timeline generation algorithm. It is helpful to view topological sorting vertices in V as a scheduling problem (for example, taking courses to complete an undergraduate degree), and think edges in E as scheduling constraints (if there is an edge from u to v, it means course u is a prerequisite for course v). Kahn's topological sort algorithm for an acyclic directed graph (DAG) $G = (V, E)$ is to repeatedly find a vertex of in-degree 0, output it, and remove it and all of its outgoing edges from the graph. The key observation of the algorithm is that, as long as the in-degree of a vertex becomes 0, it is free, and we can safely schedule it. Once it is scheduled, the constraints it imposes on other vertices can be removed, and this potentially frees more vertices to schedule. The correctness of the algorithm follows from the fact that there always exists some vertex of in-degree 0 in a DAG.[11] Note that removing a vertex of in-degree 0 and deleting all its outgoing edges from a DAG can never create new cycles; therefore, the algorithm will proceed until all vertices are put in a topological ordering. The running time of Kahn's algorithm is $O(m + n)$ by the following analysis. First, we maintain a queue which is set to be empty initially for the vertices of in-degree 0. Second, in the preprocessing stage, by scanning all edges in E, we can compute the in-degree of every vertex, and place all in-degree 0 vertices in the queue; this takes $O(m + n)$ time. Last, after removing a vertex u of in-degree 0 from the queue, we go through u's outgoing edges, delete them from E, and update the in-degrees of u's out-neighbors, and insert any vertex into the queue if its in-degree

[11] To see this, suppose this is not the case. Then every vertex has at least one incoming edge. Now starting from any vertex u of G, repeatedly trace along (in the opposite direction) any of its incoming edges. Since every vertex has at least one incoming edge, this process will keep running. But graph G is finite, therefore the path of this process must eventually visit some vertex it has visited before. That is, there must exist a cycle in this path. Reversing the direction of each edge in this path gives rise to a cycle in the original graph G, thus a contradiction.

becomes 0. Note that we need to process each edge only once and process each vertex $O(1)$ times in the whole algorithm, so this part also takes $O(m+n)$ time.

3.5.2.2 Greedy Kahn's algorithm for minimum normal timeline Note that the Kahn's algorithm described above already provides a normal form timeline for time-points in V. What we do next is to slightly modify the algorithm so that it outputs the minimum normal form timeline.

Since a normal form timeline can only schedule time-points at $t = 1, 2, \ldots$, we partition our timeline construction into stages accordingly: at stage *i*, the algorithm determines which time-points the timeline L should map to value *i*. A simple idea to construct a *minimum* normal form time is, therefore to *greedily* schedule as many time-points as allowed by the PA constraint graph. What is the maximal set of time-points that can be scheduled first, i.e. mapped to value 1? Clearly, since all time-points with in-degree 0 do not have any constraint on them, thus can all be mapped to value 1. Following the idea of Kahn's algorithm, after mapping these time-points to 1, we delete their out-going edges (this will create some new time-points of in-degree 0 as argued in the preceding paragraph) and map all new in-degree 0 time-points to value 2. We keep repeating this process until all time-points have been scheduled.

Input: A consistent PA constraint graph $G = (V, E)$
Output: A minimum normal form timeline $L : V \to \mathbb{N}$

```
1:  t = 0
2:  for all v ∈ V(G) do
3:      d_in(v) = 0
4:  end for
5:  S = ∅
6:  for all (u, v) ∈ E(G) do
7:      d_in(v) = d_in(v) + 1                ▷ compute the in-degree of every time-point
8:  end for
9:  while V ≠ ∅ do
10:     t = t + 1
11:     S = {u ∈ V : d_in(u) = 0}            ▷ set S contains all currently "free" time-points
12:     for all u ∈ S do
13:         remove vertex u from V
14:         L(u) = t                         ▷ greedily schedule all time-points that are "free"
15:         delete all outgoing edges of u   ▷ remove all constraints u imposes on other
    time-points
16:         remove u from S
17:     end for
18: end while
```

Algorithm 2 GreedyKahn's algorithm

The pseudo-code of our greedy Kahn's algorithm is listed in Algorithm 2. It is easy to see, following the same argument as that of the original Kahn's algorithm, the running time of Greedy Kahn's algorithm is $O(m+n)$. The correctness of the algorithm follows from the following lemma.

Lemma 3.5.1 *For any consistent PA constraint graph $G = (V, E)$, Greedy Kahn's Algorithm generates the minimum normal form timeline for G.*

Proof Let L be the normal form timeline generated by Greedy Kahn's Algorithm and let L' be any other normal form timeline. We claim that for any $u \in V$, $L(u) \leq L'(u)$, hence by definition L is the minimum normal form timeline.

Note that Greedy Kahn's Algorithm partitions the time-points in V into ℓ disjoint sets, $V = \dot{\bigcup}_{i=1}^{\ell} S_i$, where $\dot{\bigcup}$ stands for disjoint set union and $S_i := \{u \in V : L(u) = i\}$ for every $1 \leq i \leq \ell$. We prove the claim by induction on the index number i of $\{S_i\}$ to which a time-point belongs. The claim is certainly true for every time-point in S_1. Now for the inductive step, suppose the claim is true for all time-points in $S_1 \cup \cdots \cup S_k$. That is, any other timeline L' must have $L'(u) \geq L(u)$ for every time-point u in $S_1 \cup \cdots \cup S_k$. Let v be a time-point in S_{k+1}. Why can not v be in S_k? Because there must exist another (or more) time-point w such that $(w, v) \in E$ and $w \in S_k$. So if L' satisfies that $L'(v) = k' \leq k$, then we must also have $L'(w) < k' \leq k = L(w)$, a contradiction. This completes the inductive step of the proof and hence the proof of the lemma. □

Note that van Beek (1992) developed an efficient algorithm that finds a consistent scenario (i.e. timeline) for a PA graph by first decomposing the original graph into strongly connected components (SCCs) and then applying topological sort on the simplified PA graph. However, our application of the greedy Kahn's algorithm in solving PA graphs seems to be new.

The timeline for the running example is shown in Fig. 9. As before, there are two temporally and aspectually connected subgraphs, where the connection between them (the gray bar) represents the factive subordinating link. As can be seen, this is a trunk-and-branch multi-timeline structure, where the trunk is (in this case) the "main" timeline and the branch is the subordinated timeline.

As we have noted, one major advantage of tlex is its ability to properly handle subordinating relations. Prior approaches to timeline extraction ignored subordinating relations. If they were alternatively treated as some sort of temporal relation, machine-learning-based techniques e.g., Verhagen et al. (2005), Bethard (2013) and Mirza and Tonelli (2016) would likely insert events 8 and 9 between or near time-points 13^- and 13^+. That would indicate events 14 and 15 happened after events 1–12 but before events 17–20, while in the story world, these events did not occur at all. This is clearly incorrect.

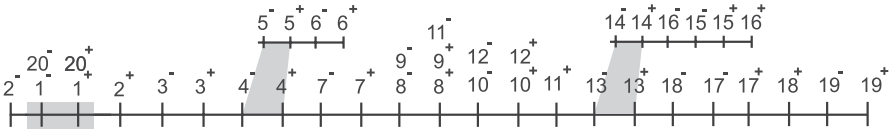

Fig. 9 Visualization of the timeline extracted from Fig. 5. The three subgraphs are arranged into a main timeline and subordinated timelines connected by a grey branch. The indeterminate sections (20 & 1) are highlighted

3.6 Indeterminacy detection

Indeterminacy in TLEX refers to portions of the timeline where multiple valid orderings exist due to ambiguous or underspecified TimeML relations. Our method detects such cases by analyzing the temporal constraints imposed by TLINKS and ALINKS, highlighting where event ordering is not uniquely determined. This feature distinguishes TLEX from previous timeline extraction methods, which assume a single fixed ordering without explicitly marking indeterminate regions.

In most cases, TimeML graphs lack enough information to uniquely specify the full ordering; a simple example is shown in Fig. 7. For that TimeML graph, only the orderings among the first event and the last two events in the timeline are uniquely determined; namely $1^- < 1^+ < 2$ and $3 < 4^- < 4^+ < 5^- < 5^+$. On the other hand, the ordering between events 2 and event 3 is indeterminate. In fact, there are 13 different possible scenarios, as shown in Fig. 10.

As another example, consider the following two events in Example (9): John worked in the morning and John ignored the text in the morning. Both events happened in the morning; however, we do not have enough information to determine the order of these two events. It is possible he ignored the text before he started working and it is possible that he ignored it while he was working. Using the DFS algorithm to be developed in Lemma 3.6.1, we can highlight the temporal indeterminacy as shown in Fig. 9.

Therefore, in general, we would like to know whether there is any indeterminacy between two time-points in a timeline. Specifically, for any two time-points u and v

(1) $1^- < 1^+ < 2^- < 2^+ < 3^- < 3^+ < 4^- < 4^+ < 5^- < 5^+$
(2) $1^- < 1^+ < 3^- < 3^+ < 2^- < 2^+ < 4^- < 4^+ < 5^- < 5^+$
(3) $1^- < 1^+ < 2^- < 2^+ = 3^- < 3^+ < 4^- < 4^+ < 5^- < 5^+$
(4) $1^- < 1^+ < 3^- < 3^+ = 2^- < 2^+ < 4^- < 4^+ < 5^- < 5^+$
(5) $1^- < 1^+ < 2^- = 3^- < 2^+ < 3^+ < 4^- < 4^+ < 5^- < 5^+$
(6) $1^- < 1^+ < 2^- = 3^- < 3^+ < 2^+ < 4^- < 4^+ < 5^- < 5^+$
(7) $1^- < 1^+ < 3^- < 2^- < 2^+ = 3^+ < 4^- < 4^+ < 5^- < 5^+$
(8) $1^- < 1^+ < 2^- < 3^- < 2^+ = 3^+ < 4^- < 4^+ < 5^- < 5^+$
(9) $1^- < 1^+ < 2^- < 3^- < 3^+ < 2^+ < 4^- < 4^+ < 5^- < 5^+$
(10) $1^- < 1^+ < 3^- < 2^- < 2^+ < 3^+ < 4^- < 4^+ < 5^- < 5^+$
(11) $1^- < 1^+ < 2^- = 3^- < 2^+ = 3^+ < 4^- < 4^+ < 5^- < 5^+$
(12) $1^- < 1^+ < 2^- < 3^- < 2^+ < 3^+ < 4^- < 4^+ < 5^- < 5^+$
(13) $1^- < 1^+ < 3^- < 2^- < 3^+ < 2^+ < 4^- < 4^+ < 5^- < 5^+$

Fig. 10 All possible solutions for the graph shown in Fig. 7, with indeterminate orderings highlighted

for which $u < v$ in a given minimum normal form timeline[12] output by Algorithm 2, is there another timeline in which $u = v$ or $v < u$? As part of TLEX, we have an efficient subroutine to check this.

Lemma 3.6.1 *Let L be the minimum normal form timeline output by Algorithm 2 in Step 4 of TLEX, and let u and v be two time-points in L with $u < v$. Then there is an algorithm with running time $O(m + n)$ which checks whether there is another consistent timeline in which $u = v$ or $v < u$.*

Proof Observe that there exists a timeline L' in which $v = u$ or $v < u$ if and only if there is no directed path from *u* to *v* in the PA constraint graph, as otherwise, *u* must precede *v* in any consistent timeline. But such a fact can be checked by running the DFS algorithm depicted in Algorithm 1 starting from vertex *u*. Note that the difference between our current algorithm and Algorithm 1 is that here we only run DFS-VISIT(G, u) (instead of exhausting all vertices in the graph) and terminate once the the DFS tree rooted at *u* is completely generated. If the DFS tree does not contain *v*, then there is no such path and there are other consistent timelines in which *u* and *v* mapped to the same time or *v* precedes *u*. Clearly, such a check takes at most the running time of the DFS algorithm, which is $O(m + n)$. □

Note that, in the worst case that we perform such indeterminacy checking for all pairs of time-points, the total running time of Step 5 of TLEX is at most $\frac{n(n-1)}{2} \cdot O(m + n) = O(mn^2 + n^3)$.

3.7 End-to-end main theorem

Informally, we call a timeline the "main" timeline if it contains events and times that actually happen in the "world" described in the text. Because we do not currently have an automatic way of identifying main timelines, we will rely on information external to TLEX to identify the main timelines. For this, it is sufficient for a person to identify at least one event or time that occurs on every disjoint main timeline for the text.

Before stating our main theorem, we need several definitions.

Definition 3.7.1 (*Trunk-and-Branch Timeline*) Let $G_{TM} = (V_{TM}, E_{TM})$ be a consistent TimeML graph with temporally and aspectually connected subgraphs $\{G^{(1)}_{TM}, \ldots, G^{(k)}_{TM}\}$ and corresponding timelines $T_L = \{L_1, \ldots, L_k\}$, and let

[12] We only need to check the case that one time-point is in front of another time-point, since if these two time-points are mapped to the same time *t* in the minimum normal timeline *L*, then the problem has a trivial solution. To see this, consider the following two possibilities. First, *u* and *v* belong to the same "compound" time-point. This means that there is an equality constraint between them, therefore these two time-points must always be mapped to the same time in any consistent timeline. Second, *u* and *v* do not belong to the same "compound" time-point. Since they are mapped to the same time *t* in the minimum normal timeline, there does not exist any (direct or indirect) constraint between *u* and *v*. Hence we can construct a timeline in which $u < v$ and another timeline in which $v < u$.

$T_M \subset T_L$ be the set of main timelines. A *trunk-and-branch timeline* for G_{TM} is a tuple $TBT(G_{TM}) = \langle G_{TBT}, T_M \rangle$, where G_{TBT} is the graph obtained from concatenating[13] all timelines in T_L together and combined with the set of subordinating TimeML links between timelines; namely, $V(G_{TBT}) = V_{TM} = \cup_{i=1}^{k} V(G_{TM}^{(i)})$, $E(G_{TBT}) = \left(\bigcup_{i=1}^{k} E(L_i)\right) \bigcup E_S$, and E_S is the set of subordinating links between timelines.

Definition 3.7.2 (*Minimum normal form trunk-and-branch timeline*) Let $TBT(G_{TM}) = \langle G_{TBT}, T_M \rangle$ be a trunk-and-branch timeline for a consistent TimeML graph G_{TM} with temporally connected subgraphs $\{G_{TM}^{(1)}, \ldots, G_{TM}^{(k)}\}$ and corresponding timelines $T_L = \{L_1, \ldots, L_k\}$. We say $TBT(G_{TM})$ is a *normal form trunk-and-branch timeline* if every timeline in T_L is a normal form timeline. Furthermore, we say $TBT(G_{TM})$ is the *minimum normal form trunk-and-branch timeline* if every timeline in T_L is a minimum normal form timeline.

By combining the algorithms and running time analysis in Sects. 3.2, 3.3, 3.4, 3.5 and 3.6, we finally arrive at the following end-to-end main theorem of TLEX:

Theorem 3.7.1 *On an input of TimeML graph $G_{TM} = (V_{TM}, E_{TM})$, with temporally connected subgraphs $\{G_{TM}^{(1)}, \ldots, G_{TM}^{(k)}\}$. Let $V_M \subset V_{TM}$ be a subset of events or times which are identified as on "main timelines". Then TLEX produces one of the two following outputs:*

- if all $\{G_{TM}^{(i)}\}_{i \in [k]}$ are consistent, *TLEX* outputs
  1. the minimum normal-form trunk-and-branch timeline $TBT(G_{TM}) = \langle G_{TBT}, T_M \rangle$ for G_{TM}, and every $i \in [k]$ such that $V(L_i) \cap V_M \neq \emptyset$ is identified as a main timeline; and
  2. a list of indeterminacy tables $\{ID_1, \ldots, ID_k\}$ such that table ID_i stores whether there is indeterminacy between each pair of time-points in timeline L_i.
- If some $G_{TM}^{(i)}$ is inconsistent, *TLEX* outputs a Maximal List of Inconsistent Cycles (MLIC) for each inconsistent temporally connected subgraph.

Moreover, the running time of *TLEX* is at most $O(n^2 m + n^3)$, where *n* is the number of events or time expressions and *m* is the number of TimeML links in the input TimeML graph.

[13] The *concatenation* of two normal form timelines L_1 and L_2, of graph $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ respectively, is defined in a natural way as follows. Suppose, without loss of generality, that timeline L_1 precedes timeline L_2, and the length of L_1 is ℓ. Then the concatenated timeline between L_1 and L_2, denoted $L = L_1 \circ L_2$ with $L : V_1(G_1) \cup V_2(G_2) \to \mathbb{N}$, is defined to be $L(v) = L_1(v)$ if $v \in V_1(G_1)$ and $L(v) = \ell + L_2(v)$ if $v \in V_2(G_2)$. More generally, if $L_1, L_2, \ldots, L_k$ is a sequence of normal form timelines arranged in temporally-increasing order, then their concatenation is defined to be $L = L_1 \circ L_2 \circ \cdots \circ L_k$.

Table 3 Details of the inconsistencies of the four corpora

Corpus	Total files	Inconsistent files (TLINKs only)	Inconsistent files (TLINKs & ALINKs)
Timebank	183	30	65
N2 Corpus	67	10	11
ProppLearner	15	9	11
MEANTIME	120	36	36[a]
Total	385	85	123

[a]The MEANTIME corpus does not have ALINKs

Table 4 Details of the timelines extracted from the four corpora

Corpus	Length of main timeline			No. of subordinated branches/text			No. of indeterminate sections/text		
	Min	Avg.	Max	Min	Avg.	Max	Min	Avg.	Max
TimeBank	4	8.1	18	1	16.0	142	0	5.3	11
N2 Corpus	6	24.7	72	1	19.3	106	0	10.2	21
ProppLearner	128	180.5	290	33	82.7	134	0	28.8	63
MEANTIME	4	7.3	13	0	2.0	7	0	3.8	7
Total	4	17.5	290	0	14.8	142	0	6.6	63

4 Experimental evaluation

We now discuss our experimental evaluations and show that TLEX performs as expected on real data. We evaluated TLEX on the four corpora described previously in Table 2. As described in Sect. 3.4, we ran CONSISTENCY CHECKING [14] across all inconsistent texts and manually corrected those TimeML graphs with reference to the original text. To emphasize the importance of ALINKs, we performed inconsistency detection on TimeML graphs with and without ALINKs. Table 3 shows the number of inconsistent files in corpora. Without ALINKs, TLEX determined that 30 texts across the four corpora had inconsistent TimeML graphs. Importantly, these inconsistencies are exactly the same 30 as those found by Derczynski and Gaizauskas (2010), which checks the consistency of the TimeML graphs by transforming them into PA graphs and applying their closure algorithm. But more importantly, the number of inconsistent texts increases significantly when ALINKs are included. To correct these inconsistencies, we removed self-loops automatically and manually corrected larger cycles. All manual corrections were by reference to the original texts. We produced fully consistent TimeML graphs for all 385 texts. In some cases, we corrected inconsistencies by removing an incorrect link in the original TimeML graph; in other cases by changing an incorrectly labeled link. The details of the corrections to TimeBank are reported elsewhere (Ocal et al., 2022).

After correction, we extracted the minimum normal form timeline for each text using the TLEX TIMELINEGENERATION step and identified their indeterminate sections using TLEX INDETERMINACY DETECTION. Table 4 shows statistics on the length of the

[14] Experimental evaluation showed that the inconsistency detection algorithm successfully detected all inconsistencies. In other work we used the detected inconsistencies to manually correct TimeBank, a subset of the corrections provided in this paper (Ocal et al., 2022).

minimum main timelines (i.e., the number of time steps in the minimum solution), the number of subordinated timelines, and the number of indeterminate sections.

Our experiments took less than a minute for the entire 385 TimeML annotated files from four corpora on a current, standard consumer laptop (2.4 GHz 4-core Intel i7 3630QM with 8 GB of RAM).

4.1 Sampling evaluation

As we do not have gold-standard annotated timelines against which to compare, we cannot directly compute recall, precision, or F_1 for the output of TLEX. Therefore we performed a sampling evaluation to check the extracted timelines, where we selected aspects of timelines at random to check against the original texts. We used *Simple Random Sampling* (SRS) (Saunders et al., 2009, p. 222) wherein every member of a population has an equally likely chance of being selected. In SRS, we check the correctness of a specific feature of a set of *n* members randomly selected from a population with size *N* to obtain an estimate of the correctness of that feature over all the data. The formula for calculating sample size for a finite population is:

$$n = \frac{n_0}{1 + \frac{n_0 - 1}{N}} \quad \text{where} \quad n_0 = \frac{Z^2}{4c^2}$$

Where *c* is the confidence interval, *Z* is the *Z*-score for the confidence level, and *N* is the population size. For all experiments, we used a *c* of $0.02(2\%)$ and a confidence level of $0.95(95\%)$, which corresponds to a *Z*-score of 1.96.

Evaluation experiments were performed by the first author and an independent annotator that we hired. Both the annotator and the first author performed SRS separately, and we calculated the inter-annotator agreement (IAA) score based on their results. We evaluated our extracted timelines via simple random sampling, checking five features: (1) the ordering of time-points with respect to each other, (2) the number of main timelines, (3) the placement of events on main versus subordinated timelines, (4) the connecting point of subordinated timelines, and (5) whether indeterminate sections are truly indeterminate with respect to the underlying TimeML graph.

4.2 Time-point ordering

Time-point ordering was the first feature we evaluated. We randomly picked neighboring pairs of time-points from the TLEX-generated timelines and compared them to the TimeML graph to ensure that the timeline and the graph were consistent. For example, if we examined a pair of neighboring points labeled A^+ and B^-, we examined the paths in the TimeML graph between A and B to determine if this ordering was correct. Using SRS, we selected 2,264 pairs of time-points out of 39,534 possible neighboring assignment pairs, sampled from each corpus in proportion to the total number of time-points in each. Both the independent annotator and the first author evaluated the correctness of the ordering, and found each pair to be correctly ordered with respect to the original text (IAA kappa score $\kappa = 1$). Since the confidence interval is 2%, this results in an estimated accuracy of 98–100% with 95% confidence.

Unlike prior approaches, TLEX achieves a perfect ordering accuracy, modulo sampling error and correctness of the TimeML graph, using all relation types.

4.3 Number of main timelines

Identification of the main timelines requires identification of at least one event or time on each main timeline, which is a laborious process. We approximated this identification, therefore, by identifying as main timelines all timelines that did not have incoming subordination links from another timeline in the trunk-and-branch timeline.

With this identification in hand, we assessed whether multiple main timelines actually corresponded to temporally disjoint and otherwise unrelatable graphs. In texts with multiple main timelines, we define the *breaking pair* between two timelines as the pair of events, times, or an event and time, one from each timeline, which are closest in the text measured by number of intervening words. Because the population size (1,241) was not substantially larger than the sample size (818), we manually checked all breaking pairs in our extracted timelines, and whether they actually indicated disjoint timelines. Both the annotator and the first author observed that all of the breaking pairs corresponded with true disjoint breaks between timelines in the texts (IAA kappa score $\kappa = 1$).

For some cases, we observed that sometimes a text had multiple timelines which were disjoint, but should have been collected together into a single main timeline because they were temporally relatable. This is because the annotation of the text is missing a relation between one group of events or times and another group of events or times, but in both cases those events occur in the "real world" of the text. In other words, the TimeML graph was disconnected in a way that was inconsistent with the actual semantics of the text, meaning the TimeML annotation was incorrect. We observed this situation for 181 texts, which is another dimension along which the TimeML annotations for these corpora might be improved. In our other work, we investigated the disconnectivity in TimeML graphs and showed that these disconnectivities mainly occur by a missing link between two subgraphs due to a manual annotation error (Ocal et al., 2022). We corrected these erroneous disconnections in other work (Ocal et al., 2022).

4.4 Time-point placement

We evaluated whether time-points were correctly placed on main versus subordinated timelines, using our approximation for main timelines as indicated previously. The first author and the annotator checked to see if a time-point placed on a subordinate timeline did not occur in the "real world" described in the text. 11,474 time-points were on subordinated timelines, and we sampled 1,986 time-points across the corpora, again in proportion to their number in each corpus. Again, all samples were correct (IAA kappa score $\kappa = 1$), giving an estimated accuracy of 98–100% with 95% confidence.

4.5 Subordinated connecting points

We checked to confirm that subordinated timelines were connected to the correct time-points on the identified main timelines. There are 5,701 subordinated timelines in our data, and every one was connected to a main timeline with at least one subordinating link. The first author and the annotator checked 1,690 connections, observing that every connection was correctly placed (IAA kappa score $\kappa = 1$), giving an estimated accuracy of 98–100% with 95% confidence.

4.6 Indeterminate sections

Finally, the first author and the annotator checked whether indeterminate sections were truly indeterminate with respect to the TimeML graph. There are 2,541 indeterminate sections, comprising 11,688 pairs of time-points. We randomly selected 1,992 time-point pairs from the indeterminate sections. In all cases the pairs were truly indeterminate (IAA kappa score $\kappa = 1$), meaning an estimated accuracy of 98–100% with a 95% confidence.

Our evaluation demonstrates that TLEX successfully detects inconsistencies across TLINK and ALINK relations, providing a broader coverage than prior methods that focused solely on TLINK consistency. Moreover, the identification of indeterminate sections adds a new layer of interpretability to extracted timelines, making it possible to distinguish between fully ordered sequences and ambiguous event structures. These features make TLEX a more comprehensive tool for analyzing temporal structures in narrative texts.

5 Future work

Our work naturally suggests several interesting directions of future research.

Our investigation showed that manually annotated TimeML texts often contain errors that result in temporal inconsistencies; automatic methods for generating TimeML annotations are currently even more noisy and error-prone. Because of this, a logical next step is to develop methods that automatically suggest corrections to the maximal list of inconsistent cycles identified in the CONSISTENCY CHECKING step. Both rule-based and supervised machine learning-based approaches could be developed to address this problem.

Finally, it is interesting to investigate the performance of TLEX on automatically generated TimeML graphs. Indeed, TLEX may be applied in this way to measure the quality of the timelines extracted, which provides a good indication of how useful automatic TimeML extraction is in practice.

6 Contributions

In this work, we developed TLEX (TimeLine EXtraction), an exact, end-to-end solution which extracts timelines from TimeML annotated texts. The novel features of TLEX include identifying specific relations that are involved in an inconsistency (which can then be manually corrected) and determining sections of the timelines that have indeterminate order, which is critical for downstream tasks such as aligning events from different timelines. We provide efficient algorithms which implement the tasks within TLEX and conduct experimental evaluations by applying TLEX to 385 TimeML annotated texts from four corpora. We provide a reference implementation of TLEX, the extracted timelines for all texts, and the manual corrections to the temporal graphs of the inconsistent texts.

Acknowledgements We gratefully acknowledge valuable discussions with Jared Hummer, Antonela Radas, Victor Yarlott, Karine Megerdoomian, Akul Singh, and Emmanuel Garcia. Contributions to the implementation of the TLEX algorithm in Java and Python were made by FIU KFSCIS Senior Project team members Christopher Eberhard, Stephan Belizaire, Pablo Maldonado, Luis Robaina, Adrian Silva, Victoria Fernandez, Ronald Pena, Ismael Clark, Felipe Arce, Kevin Fontela, Raul Garcia, Leandro Estevez, Carlos Pimentel, Hector Borges, Ivan Parra Sanz, Tony Erazo, Sage Pages, Gerardo Parra, and Franklin Bello Romero. This work was partially funded by research grants TO-134841, TO-135998, and TO-139837 to Dr. Finlayson from MITRE Corporation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of MITRE Corporation. This work was also partially supported by ONR Award N00014-17-1-2983 to Dr. Finlayson.

Author contributions All authors contributed equally to all aspects of this work, including the development of the research idea, experimental design, data acquisition and analysis, and the writing and revision of the manuscript. All authors approved the final version.

Data availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare no conflict of interest.

References

Allan, J., Gupta, R., & Khandelwal, V. (2001). Temporal summaries of new topics. In *Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 10–18).

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM, 26*(11), 832–843. https://doi.org/10.1145/182.358434

Barták, R., Morris, R. A., & Venable, K. B. (2014). *An introduction to constraint-based temporal reasoning*. Morgan & Claypool Publishers.

Beek, P. (1992). Reasoning about qualitative temporal information. *Artificial Intelligence, 58*(1–3), 297–326.

Bethard, S. (2013). ClearTK-TimeML: A minimalist approach to TempEval 2013. In *Second joint conference on lexical and computational semantics (* SEM), Vol. 2: Proceedings of the 7th international workshop on semantic evaluation (SemEval 2013)* (pp. 10–14).

Chambers, N., Cassidy, T., McDowell, B., & Bethard, S. (2014). Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics, 2*, 273–284.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., et al. (2024). Scaling instruction-finetuned language models. *Journal of Machine Learning Research, 25*(70), 1–53.

Condotta, J.-F., Nouaouri, I., & Sioutis, M. (2016). A SAT approach for maximizing satisfiability in qualitative spatial and temporal constraint networks. In *Proceedings of the 15th international conference on principles of knowledge representation and reasoning, KR'16* (pp. 432–442). AAAI Press.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to algorithms*. MIT.

Dechter, R., Meiri, I., & Pearl, J. (1991). Temporal constraint networks. *Artificial Intelligence, 49*(1–3), 61–95.

Derczynski, L., & Gaizauskas, R. J. (2010). Analysing temporally annotated corpora with CAVaT. In *Proceedings of the 7th conference on international language resources and evaluation (LREC'10)*. Valetta, Malta.

Do, Q. X., Lu, W., & Roth, D. (2012). Joint inference for event timeline construction. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL'12)* (pp. 677–687).

Ferro, L., Gerber, L., Mani, I., Sundheim, B., & Wilson, G. (2001). TIDES temporal annotation guidelines, ver. 1.0.2. http://www.timeml.org/terqas/readings/MTRAnnotationGuide_v1_02.pdf

Finlayson, M., Halverson, J., & Corman, S. (2014). The N2 corpus: A semantically annotated collection of islamist extremist stories. In *Proceedings of the 9th international conference on language resources and evaluation (LREC'14)*, Reykjavik, Iceland.

Finlayson, M. A. (2017). Propplearner: Deeply annotating a corpus of Russian folktales to enable the machine learning of a Russian formalist theory. *Digital Scholarship in the Humanities, 32*(2), 284–300. https://doi.org/10.1093/llc/fqv067

Freksa, C. (1992). Temporal reasoning based on semi-intervals. *Artificial Intelligence, 54*(1–2), 199–227.

Gantner, Z., Westphal, M., & Wolfl, S. (2008). GQR—A fast reasoner for binary qualitative constraint calculi. In *AAAI workshop on spatial and temporal reasoning*.

Gereveni, A., & Schubert, L. (1995). Efficient algorithms for qualitative reasoning about time. *Artificial Intelligence, 74*(2), 207–248. https://doi.org/10.1016/0004-3702(94)00016-T

Ghalandari, D. G., & Ifrim, G. (2020). Examining the state-of-the-art in news timeline summarization. In *Proceedings of the 58th annual meeting on Association for Computational Linguistics* (pp. 1322–1334).

Hasegawa, K., Kandukuri, N., Holm, S., Yamakawa, Y., & Mitamura, T. (2024). Formulation comparison for timeline construction using llms. arXiv preprint. arXiv:2403.00990

Iwata, Y., & Yoshida, Y. (2013). Exact and approximation algorithms for the maximum constraint satisfaction problem over the point algebra. In N. Portier & T. Wilke (Eds.), *30th International symposium on theoretical aspects of computer science (STACS 2013)*. Leibniz International Proceedings in Informatics (LIPIcs) (Vol. 20, pp. 127–138). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. https://doi.org/10.4230/LIPIcs.STACS.2013.127 . http://drops.dagstuhl.de/opus/volltexte/2013/3928.

Kahn, A. B. (1962). Topological sorting of large networks. *Communications of the ACM, 5*(11), 558–562.

Kolomiyets, O., Bethard, S., & Moens, M.-F. (2012). Extracting narrative timelines as temporal dependency structures. In *Proceedings of the 50th annual meeting of the Association for Computational Linguistics (ACL'12)* (pp. 88–97).

Kreutzmann, A., & Wolter, D. (2014). Qualitative spatial and temporal reasoning with and/or linear programming. In *Proceedings of the 21st European conference on artificial intelligence. ECAI'14* (pp. 495–500). IOS Press. https://doi.org/10.3233/978-1-61499-419-0-495

Leeuwenberg, A., & Moens, M.-F. (2019). A survey on temporal reasoning for temporal information extraction from text. *Journal of Artificial Intelligence Research, 66*, 341–380.

Leeuwenberg, A., & Moens, M.-F. (2020). Towards extracting absolute event timelines from English clinical reports. *IEEE/ACM Transactions on Audio, Speech, and Language Processing, 28*, 2710–2719.

Ligozat, G. (2013). *Qualitative spatial and temporal reasoning*. Wiley.

Liu, S., Zhou, M. X., Pan, S., Song, Y., Qian, W., Cai, W., & Lian, X. (2012). TIARA: Interactive, topic-based visual text summarization and analysis. *ACM Transactions on Intelligent Systems and Technology, 3*(2), 25–12528. https://doi.org/10.1145/2089094.2089101

Llorens, H., Chambers, N., UzZaman, N., Mostafazadeh, N., Allen, J., & Pustejovsky, J. (2015). Semeval-2015 task 5: Qa tempeval-evaluating temporal information understanding with question answering. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)* (pp. 792–800).

Mani, I., Verhagen, M., Wellner, B., Lee, C. M., & Pustejovsky, J. (2006). Machine learning of temporal relations. In *Proceedings of the 21st international conference on computational linguistics and the 44th annual meeting of the Association for Computational Linguistics (ICCL-ACL'06)* (pp. 753–760). Sydney, Australia.

Mansouri, B., Campos, R., & Jatowt, A. (2023). Towards timeline generation with abstract meaning representation. In *Companion proceedings of the ACM web conference 2023. WWW '23 Companion* (pp. 1204–1207). Association for Computing Machinery. https://doi.org/10.1145/3543873.3587670.

Mathur, P., Morariu, V., Kaynig-Fittkau, V., Gu, J., Dernoncourt, F., Tran, Q., Nenkova, A., Manocha, D., & Jain, R. (2022). DocTime: A document-level temporal dependency graph parser. In *Proceedings of the 2022 conference of the North American Chapter of the Association for Computational Linguistics: Human language technologies* (pp. 993–1009). Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.naacl-main.73 . https://aclanthology.org/2022.naacl-main.73

Minard, A., Speranza, M., Urizar, R., Altuna, B., Erp, M., Schoen, A., & Son, C. (2016). MEANTIME, the newsreader multilingual event and time corpus. In *Proceedings of the tenth international conference on language resources and evaluation (LREC 2016)*. Portorož, Slovenia.

Mirza, P., & Tonelli, S. (2016). Catena: Causal and temporal relation extraction from natural language texts. In *Proceedings of COLING 2016, the 26th international conference on computational linguistics: Technical papers* (pp. 64–75).

Navarro-Colorado, B., & Saquete, E. (2016). Cross-document event ordering through temporal, lexical and distributional knowledge. *Knowledge-Based Systems, 110*, 244–254. https://doi.org/10.1016/j.knosys.2016.07.032

Nebel, B., & Bürckert, H.-J. (1995). Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra. *Journal of ACM, 42*(1), 43–66. https://doi.org/10.1145/200836.200848

Ning, Q., Wu, H., & Roth, D. (2018). A multi-axis annotation scheme for event temporal relations. In *Proceedings of the 56th annual meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1318–1328). Association for Computational Linguistics. https://doi.org/10.18653/v1/P18-1122 . https://aclanthology.org/P18-1122

Ocal, M., & Finlayson, M. (2020). Evaluating information loss in temporal dependency trees. In *Proceedings of the 12th language resources and evaluation conference* (pp. 2148–2156). European Language Resources Association. https://www.aclweb.org/anthology/2020.lrec-1.263

Ocal, M., Perez, A., Radas, A., & Finlayson, M. (2022). Holistic evaluation of automatic timeml annotators. In *Proceedings of the thirteenth language resources and evaluation conference* (pp. 1444–1453).

Ocal, M., Radas, A., Hummer, J., Megerdoomian, K., & Finlayson, M. (2022). A comprehensive evaluation and correction of the TimeBank corpus. In *Proceedings of the thirteenth language resources and evaluation conference*, Marseille, France (pp. 2919–2927). https://aclanthology.org/2022.lrec-1.313.

Ocal, M., Singh, A., Hummer, J., Radas, A., & Finlayson, M. (2023). jtlex: a java library for timeline extraction. In *Proceedings of the 17th conference of the European chapter of the Association for Computational Linguistics: System demonstrations* (pp. 27–34).

Ostuni, D., Raffaele, A., Rizzi, R., & Zavatteri, M. (2021). Faster and better simple temporal problems. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 35, pp. 11913–11920).

Pustejovsky, J., Castaño, J., Ingria, R., Saurí, R., Gaizauskas, R., Setzer, A., & Katz, G. (2003). TimeML: robust specification of event and temporal expressions in text. In *Fifth international workshop on computational semantics (IWCS-5)* (pp. 1–11).

Pustejovsky, J., Hanks, P., Saurí, R., See, A., Gaizauskas, R., Setzer, A., Radev, D., Sundheim, B., Day, D., Ferro, L., Lazo, M. (2003). The TimeBank corpus. In *Proceedings of corpus linguistics conference*, pp. 647–656. Lancaster, UK

Saquete, E., Martínez-Barco, P., Muñoz, R., & Vicedo, J. L. (2004). Splitting complex temporal questions for question answering systems. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics. ACL '04*. Association for Computational Linguistics. https://doi.org/10.3115/1218955.1219027

Saunders, M. N. K., Lewis, P., & Thornhill, A. (2009). *Research methods for business students* (5th ed.). Prentice Hall.

Sauri, R., Littman, J., Gaizauskas, R., Setzer, A., & Pustejovsky, J. (2006). TimeML Annotation Guidelines, Version 1.2.1. https://catalog.ldc.upenn.edu/docs/LDC2006T08/timeml_annguide_1.2.1.pdf.

Setzer, A. (2001). *Temporal information in newswire articles: An annotation scheme and corpus study* [PhD thesis], University of Sheffield.

Singh, A., Hummer, J., Ocal, M., & Finlayson, M. (2024). pytlex: A python library for timeline extraction. In *Proceedings of the 18th conference of the European chapter of the Association for Computational Linguistics: System demonstrations* (pp. 27–34).

Sioutis, M., & Condotta, J.-F. (2014). Tackling large qualitative spatial networks of scale-free-like structure. In (*Proceedings of the 8th Hellenic Conference on AI (SETN 2014)* (Vol. 8445, pp. 178–191). Ioannina, Greece. https://doi.org/10.1007/978-3-319-07064-3_15

Swan, R., & Allan, J. (2000). Automatic generation of overview timelines. In *Proceedings of the 23rd annual international ACM SIGIR conference on research and development in information retrieval* (pp. 49–56).

Valdes-Perez, R. E. (1987). The satisfiability of temporal constraint networks., In: Proceedings of the Sixth National Conference on Artificial Intelligence, pp. 256–260.

Valdes-Perez, R. E. (1986). *Spatio-temporal reasoning and linear inequalities*. Technical report, MIT Artificial Intelligence Laboratory.

Verhagen, M. (2007). Drawing TimeML relations with TBox. In *Annotating, extracting and reasoning about time and events* (pp. 7–28). Springer.

Verhagen, M., Knippen, R., Mani, I., & Pustejovsky, J. (2006). Annotation of temporal relations with Tango. In *LREC* (pp. 2249–2252).

Verhagen, M., Mani, I., Sauri, R., Knippen, R., Jang, S. B., Littman, J., Rumshisky, A., Phillips, J., & Pustejovsky, J. (2005). Automating temporal annotation with TARSQI. In *Proceedings of the ACL 2005 on interactive poster and demonstration sessions* (pp. 81–84). Ann Arbor, MI.

Vilain, M., Kautz, H., & Beek, P. (1990). Constraint propagation algorithms for temporal reasoning: A revised report. In D. S. Weld & J. D. Kleer (Eds.), *Readings in qualitative reasoning about physical systems* (pp. 373–381). Morgan Kaufmann.

Vilain, M. B., & Kautz, H. A. (1986). Constraint propagation algorithms for temporal reasoning. *AAAI, 86*, 377–382.

Wallgrün, J. O., Frommberger, L., Wolter, D., Dylla, F., & Freksa, C. (2006). Qualitative spatial representation and reasoning in the Sparq-toolbox. In *Proceedings of the international conference spatial cognition*, Bremen, Germany (pp. 39–58). https://doi.org/10.1007/978-3-540-75666-8_3

Wang, J., & Weiss, J. C. (2025). A large-language model framework for relative timeline extraction from pubmed case reports. *AMIA Summits on Translational Science Proceedings, 2025*, 598.

Wei, S., Li, W., Song, F., Luo, W., Zhuang, T., Tan, H., Guo, Z., & Wang, H. (2025). Time: A multi-level benchmark for temporal reasoning of LLMS in real-world scenarios. arXiv preprint. arXiv:2505.12891

Yu, Y., Jatowt, A., Doucet, A., Sugiyama, K., & Yoshikawa, M. (2021). Multi-timeline summarization (MTLS): Improving timeline summarization by generating multiple summaries. In *Proceedings of the 59th annual meeting of the Association for Computational Linguistics and the 11th international joint conference on natural language processing (Vol. 1: Long papers)* (pp. 377–387).

Zhang, Y., & Xue, N. (2018). Neural ranking models for temporal dependency structure parsing. *CoRR*. arXiv:1809.00370..

Authors and Affiliations

Mustafa Ocal[1] · Ning Xie[2] · Mark A. Finlayson[3]

✉ Mustafa Ocal
mocal@fiu.edu

Ning Xie
nxie@cis.fiu.edu

Mark A. Finlayson
markaf@fiu.edu

[1] Knight Foundation School of Computing and Information Sciences, Florida International University, 11200 S.W. 8th Street, CASE Building, Room 319, Miami, FL 33199, USA

[2] Knight Foundation School of Computing and Information Sciences, Florida International University, 11200 S.W. 8th Street, CASE Building, Room 363, Miami, FL 33199, USA

[3] Knight Foundation School of Computing and Information Sciences, Florida International University, 11200 S.W. 8th Street, CASE Building, Room 362, Miami, FL 33199, USA