Temporal Graph Databases

Alejandro Vaisman

Instituto Tecnológico de Buenos Aires, Argentina
avaisman@itba.edu.ar

Twelfth European Big Data Management & Analytics Summer School (eBISS 2024), Padova, Italy

Agenda

- Introduction and motivation
- Temporal Graph Databases
- Implementation
- Temporal Graphs in Sensor Networks
- Conclusion

Agenda

- **Introduction and motivation**
  - Why graph databases?
  - Graph database models
    - Property graphs
    - RDF graphs
  - Temporal databases
  - Temporal graph databases

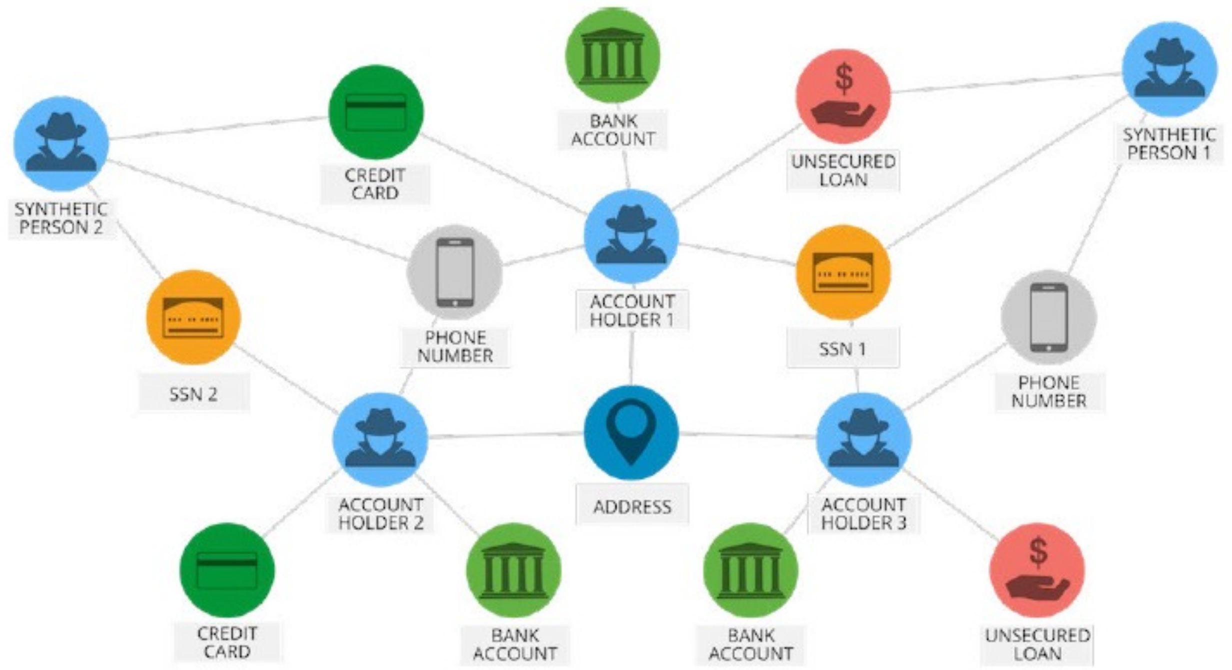
Agenda

- **Introduction and motivation**
  - Why graph databases?
  - Graph database models
    - Property graphs
    - RDF graphs
  - Temporal databases
  - Temporal graph databases

Main characteristic of (big) data

A world of interrelated information

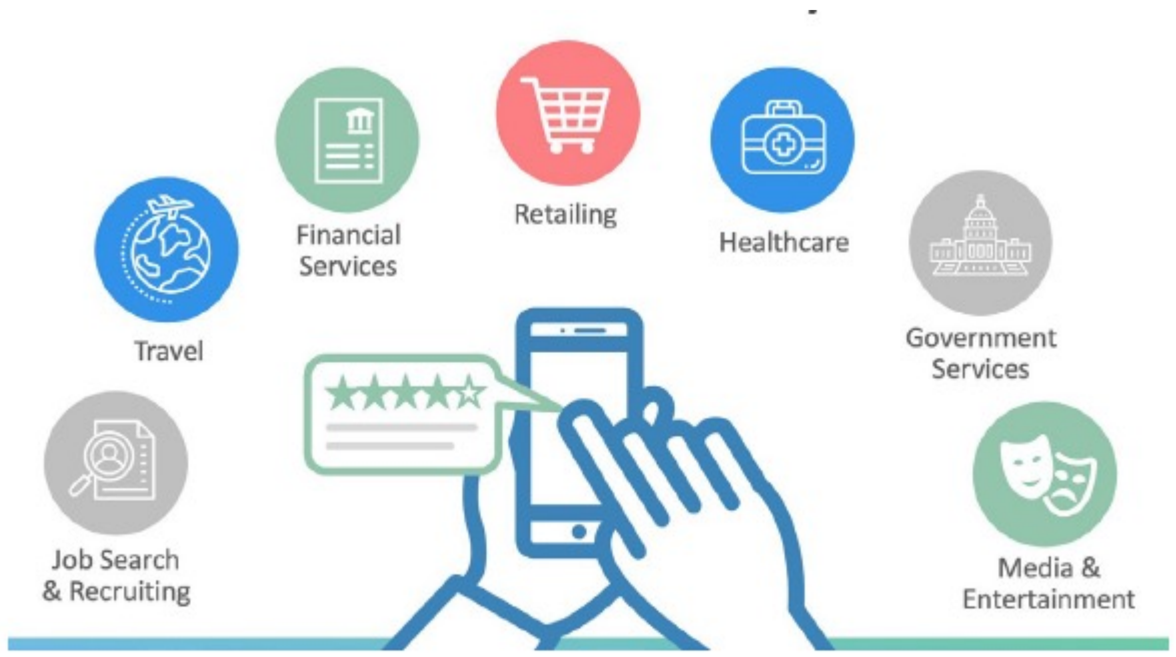
Fraud detection



https://neo4j.com/whitepapers/top-ten-use-cases-graph-database-technology/

A world of interrelated information

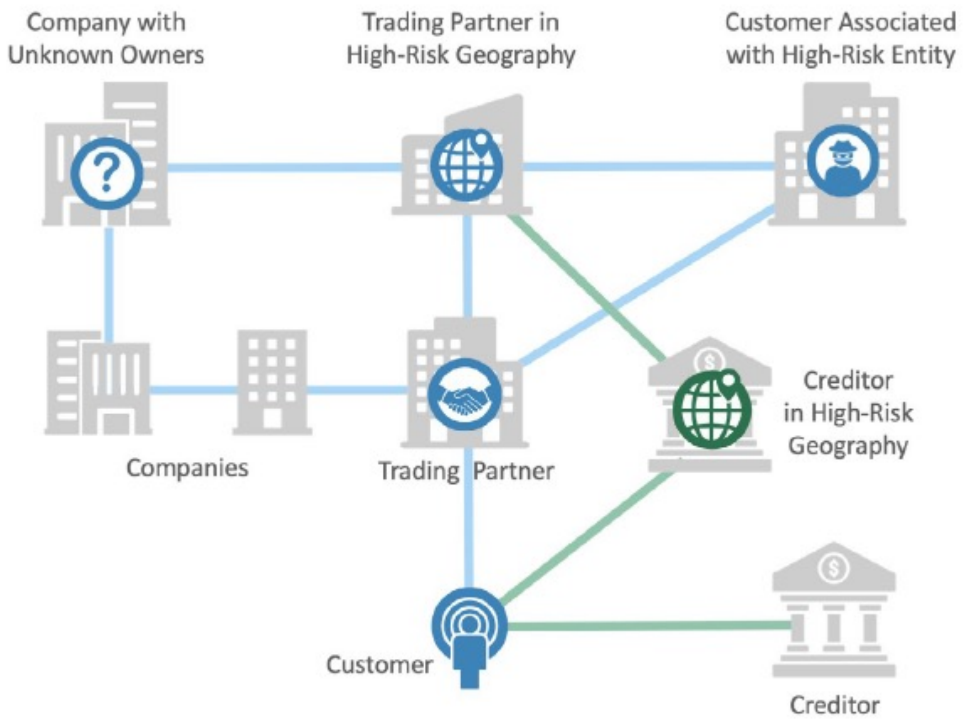
Real-time recommendation



https://neo4j.com/whitepapers/top-ten-use-cases-graph-database-technology/

A world of interrelated information

Anti-money laundering

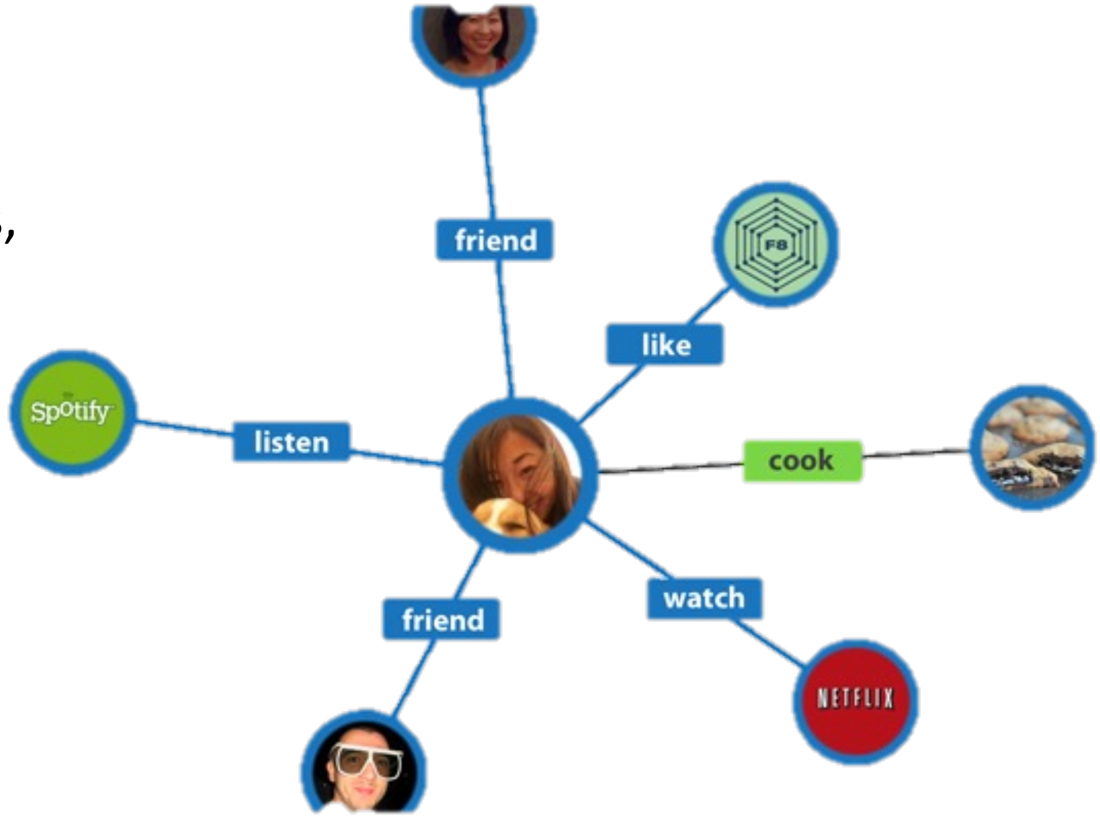


Company with Unknown Owners
Trading Partner in High-Risk Geography
Customer Associated with High-Risk Entity
Companies
Trading Partner
Creditor in High-Risk Geography
Customer
Creditor

https://neo4j.com/whitepapers/top-ten-use-cases-graph-database-technology/

Modeling data connectedness

- A social network
- Persons, friendships, photos, locations, apps, pages, ads, interests, age range, etc.



Problems & Questions

How do we deal with these data?

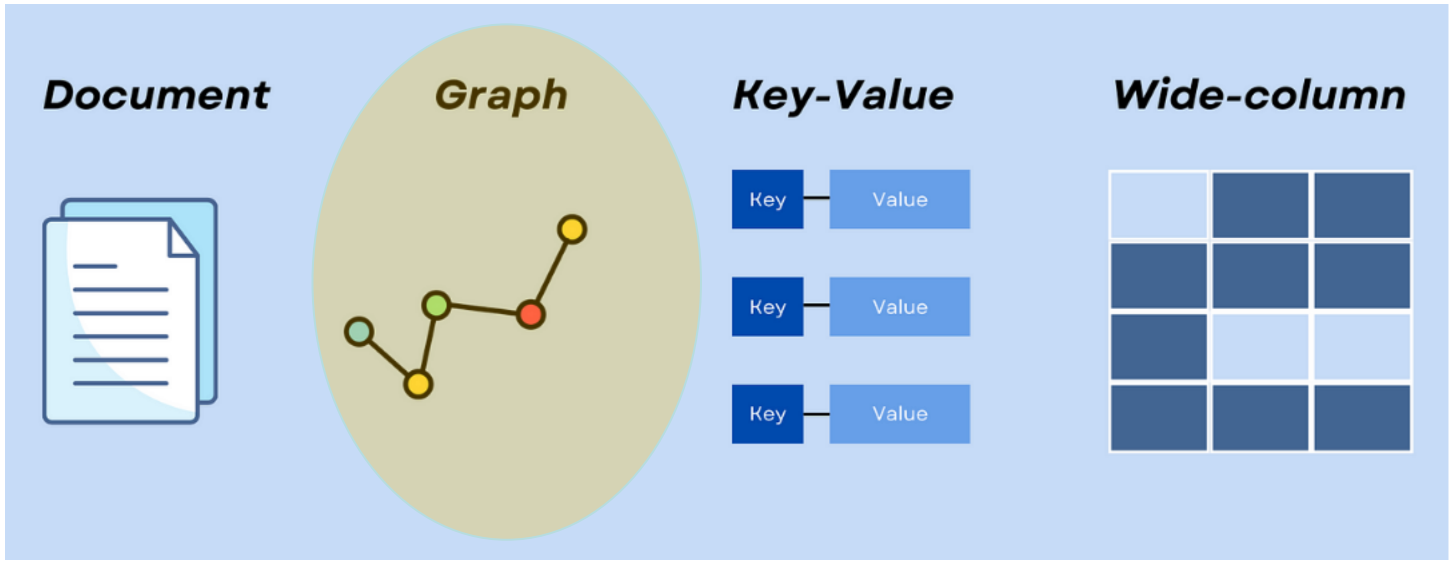
Is traditional DB technology enough?

We must address:

- Connectedness
- Unstructured data
- High Volumes
- Real-time

NoSQL technologies

Types of NoSQL databases

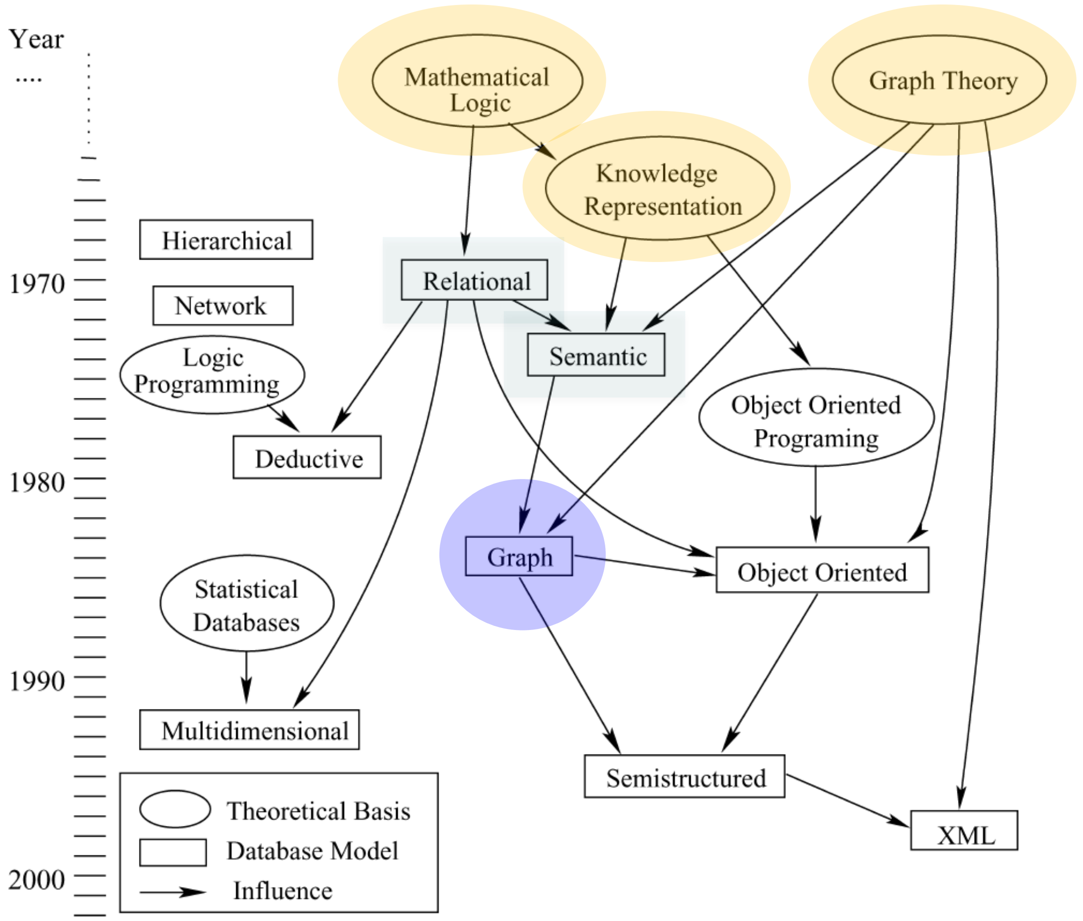


mongoDB neo4j redis cassandra

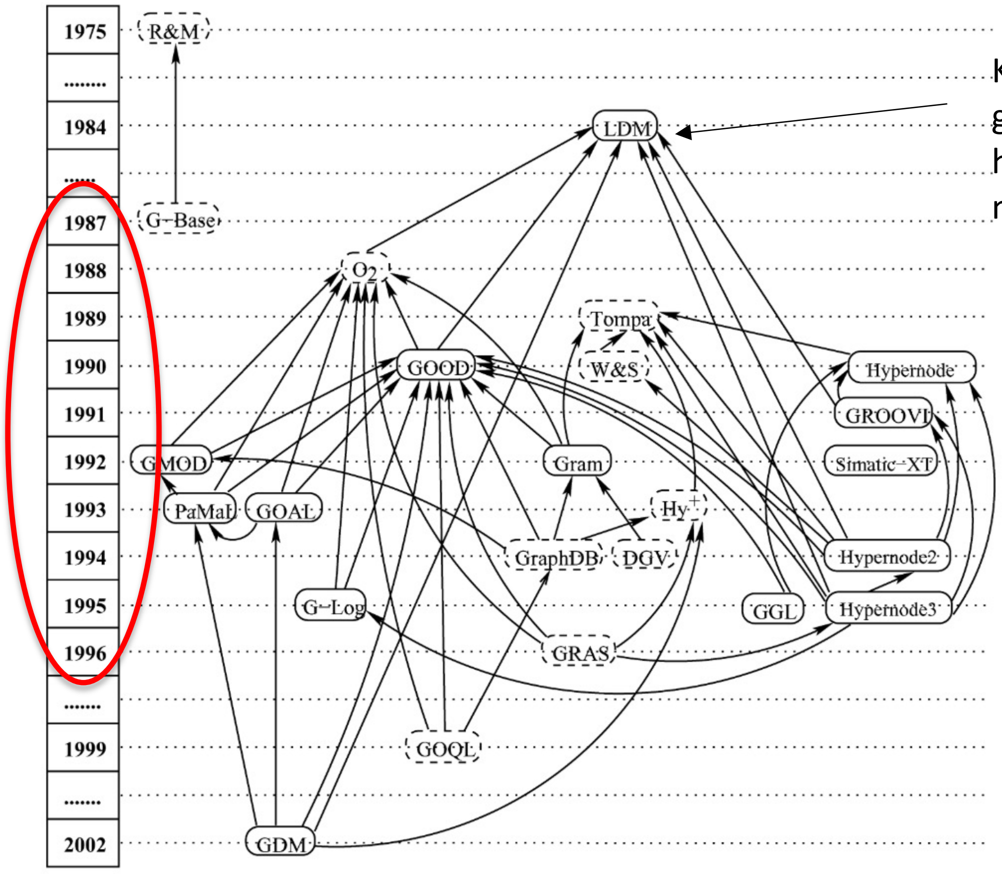
Agenda

- **Introduction and motivation**
  - Why graph databases?
  - Graph database models
    - Property graphs
    - RDF graphs
  - Temporal databases
  - Temporal graph databases

A history of database models (A. Mendelzon)



The Golden age of GDB Models



Kuper and Vardi 1984: generalize the relational, hierarchical and network models

Graph database models*

- **Database model:** three components: a set of data structure types, a set of operators or inference rules, and a set of integrity rules (Codd, 1980)

- *Data and/or the schema represented by graphs,* hypergraphs, hypernodes
  Node -> entity, edge -> relationship between entities, property -> feature

- *Data manipulation expressed by graph transformations*, or operations on graph features: paths, neighborhoods, subgraphs, patterns, connectivity, graph statistics (e.g., diameter, centrality, etc.)

- *Integrity constraints enforce data consistency*, schema-instance consistency, identity & referential integrity, functional dependencies. E.g.: labels w/ unique names, constraints on nodes, domain and range of properties

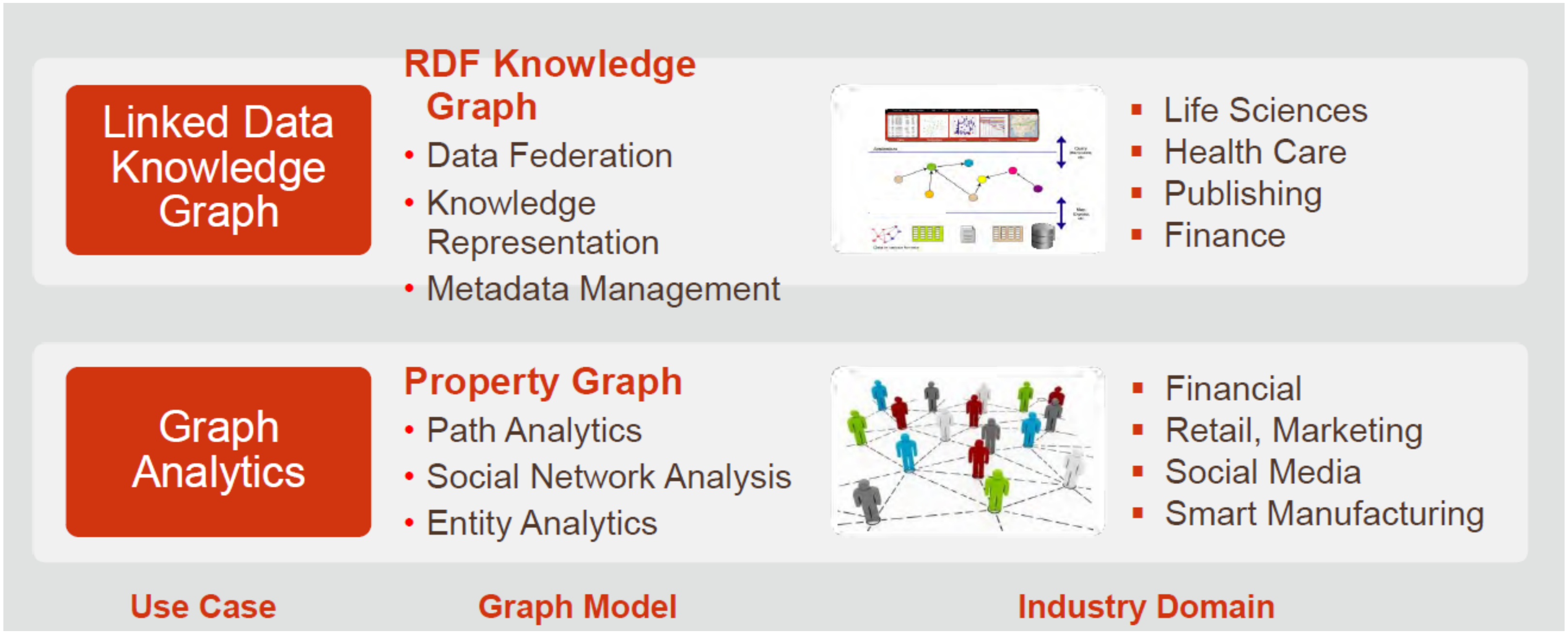
* C. Gutiérrez, R. Angles. A Survey on Graph Database Models ACM Computing Surveys, 2008

Knowledge graphs*

- Many (sometimes conflicting) definitions,   technical and general
- Knowledge graph: a graph of data (or data graph) intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent relations between these entities
- The graph of data conforms to a graph-based data model (e.g., a directed edge-labelled graph, a property graph, etc)
- Knowledge: something that is known, which may be accumulated from external sources, or extracted from the KG itself

* Hogan et al. Knowledge Graphs, ArXiv:2003.02320v6, 2021.

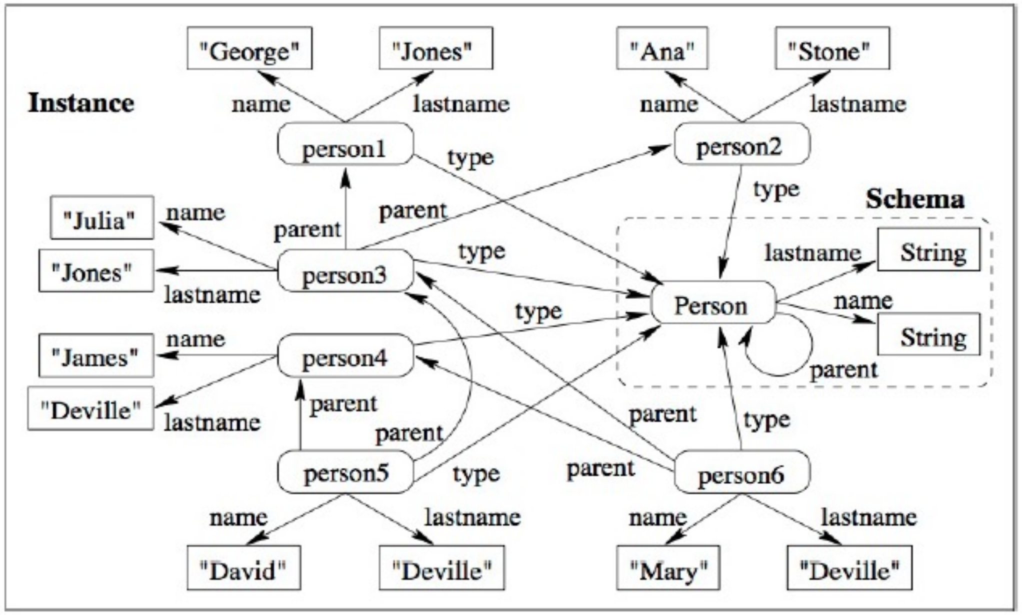
Two graph data models

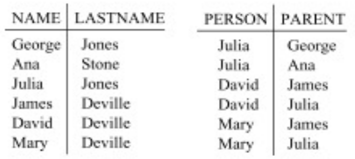


Use Case	Graph Model		Industry Domain
Linked Data Knowledge Graph	**RDF Knowledge Graph** • Data Federation • Knowledge Representation • Metadata Management		▪ Life Sciences ▪ Health Care ▪ Publishing ▪ Finance
Graph Analytics	**Property Graph** • Path Analytics • Social Network Analysis • Entity Analytics		▪ Financial ▪ Retail, Marketing ▪ Social Media ▪ Smart Manufacturing

- Perry, M. Introduction to RDF Graph for Oracle Database 19c. Architecture and Overview (2019)

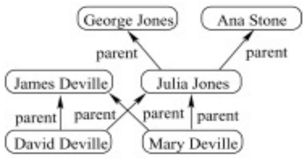
RDF graph data model





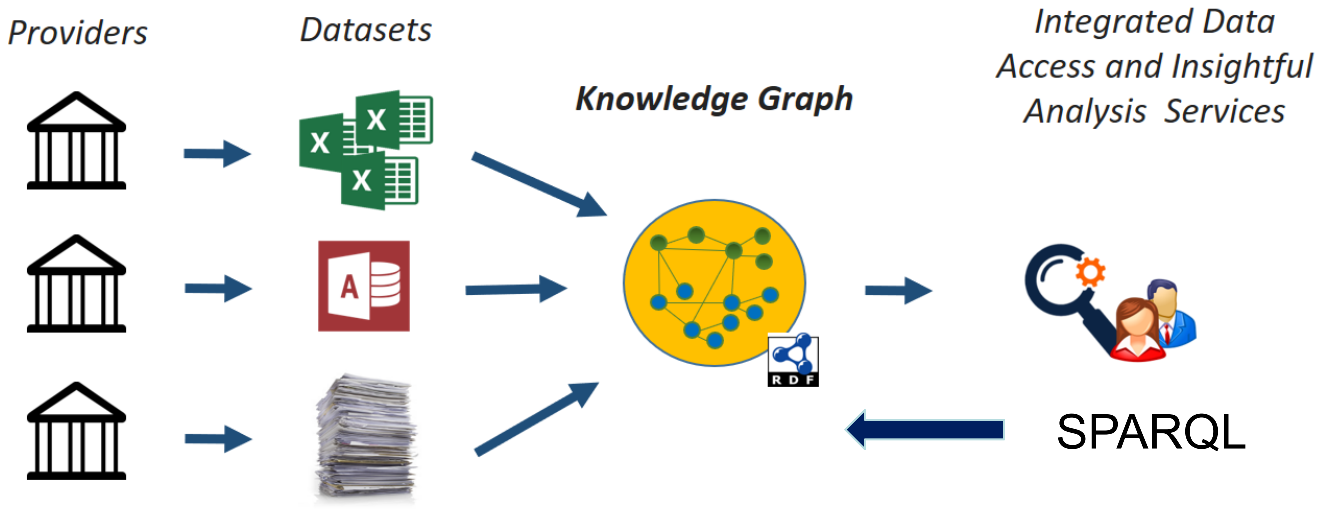
NAME	LASTNAME
George	Jones
Ana	Stone
Julia	Jones
James	Deville
David	Deville
Mary	Deville

PERSON	PARENT
Julia	George
Julia	Ana
David	James
David	Julia
Mary	James
Mary	Julia



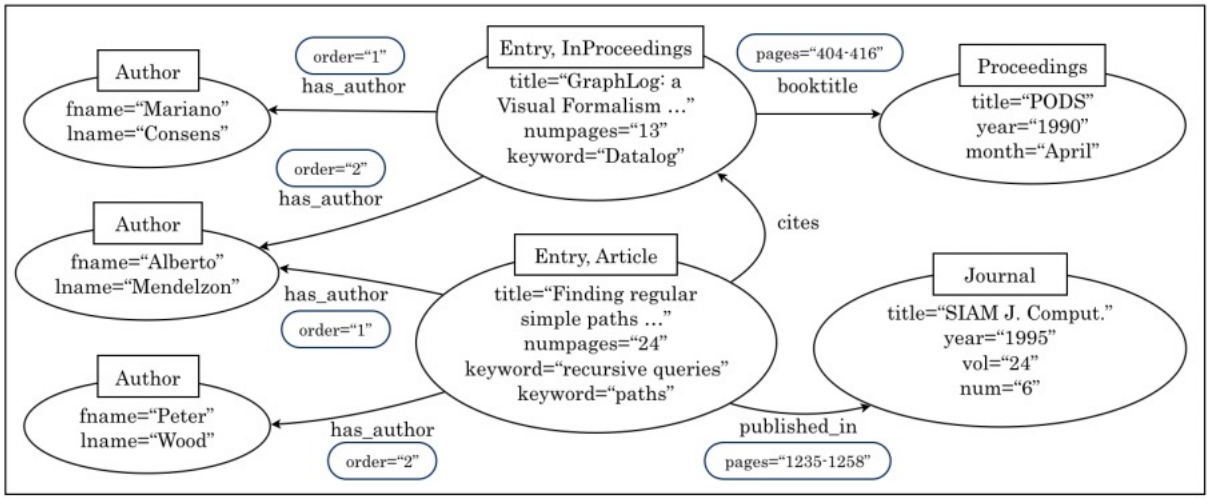
- Originally deviced to represent metadata
- Represents resources and relations between resources
- An RDF graph: a collection of (subject, predicate, object) triples
- Schema and instances represented using the same formalism

RDF Knowledge Graphs



* Papadaki et al. A Brief Survey of Methods for Analytics over RDF Knowledge Graphs, 2023

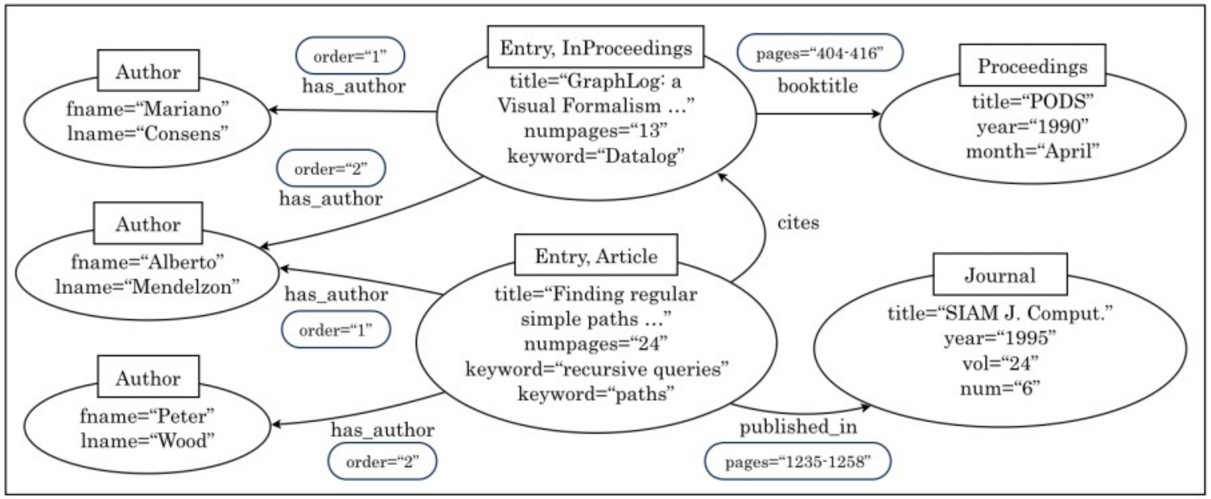
The property graph data model*



- Informally, a directed labelled multigraph where each node or edge associated with a set (possibly empty) of property-value pairs
- A node represents an entity, an edge represents a relationship between entities, a property represents a specific feature of an entity or relationship

* R. Angles. The Property Graph Database Model. AMW 2018, Cali, Colombia

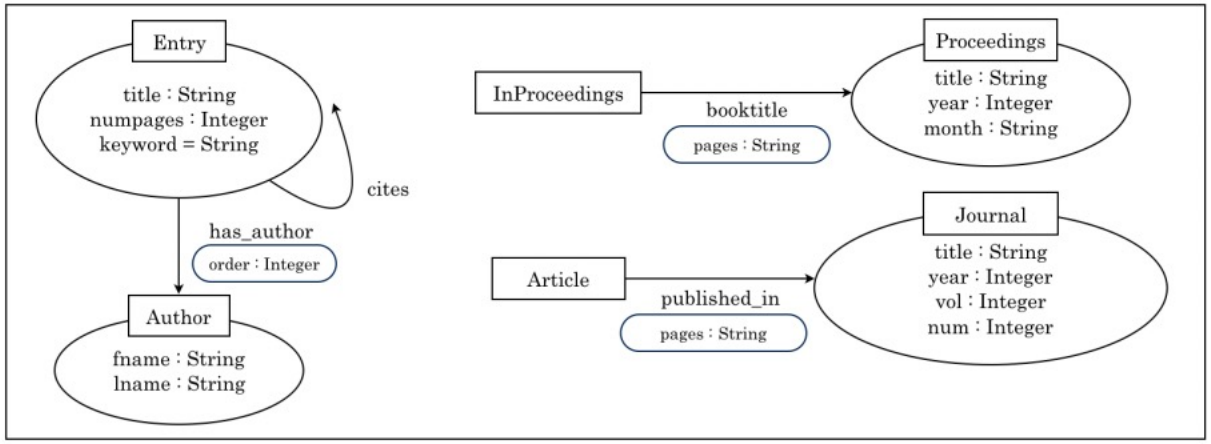
The property graph data model*



- First formal definition (Angles, 2018)

1. N is a finite set of nodes (also called vertices);
2. E is a finite set of edges such that E has no elements in common with N;
3. $\rho : E \rightarrow (N \times N)$ is a total function that associates each edge in E with a pair of nodes in N (i.e., ρ is the usual incidence function in graph theory);
4. $\lambda : (N \cup E) \rightarrow \mathrm{SET}^+(\boldsymbol{L})$ is a partial function that associates a node/edge with a set of labels from L (i.e., λ is a labeling function for nodes and edges);
5. $\sigma : (N \cup E) \times \boldsymbol{P} \rightarrow \mathrm{SET}^+(\boldsymbol{V})$ is a partial function that associates nodes/edges with properties, and for each property it assigns a set of values from $\boldsymbol{V}$.

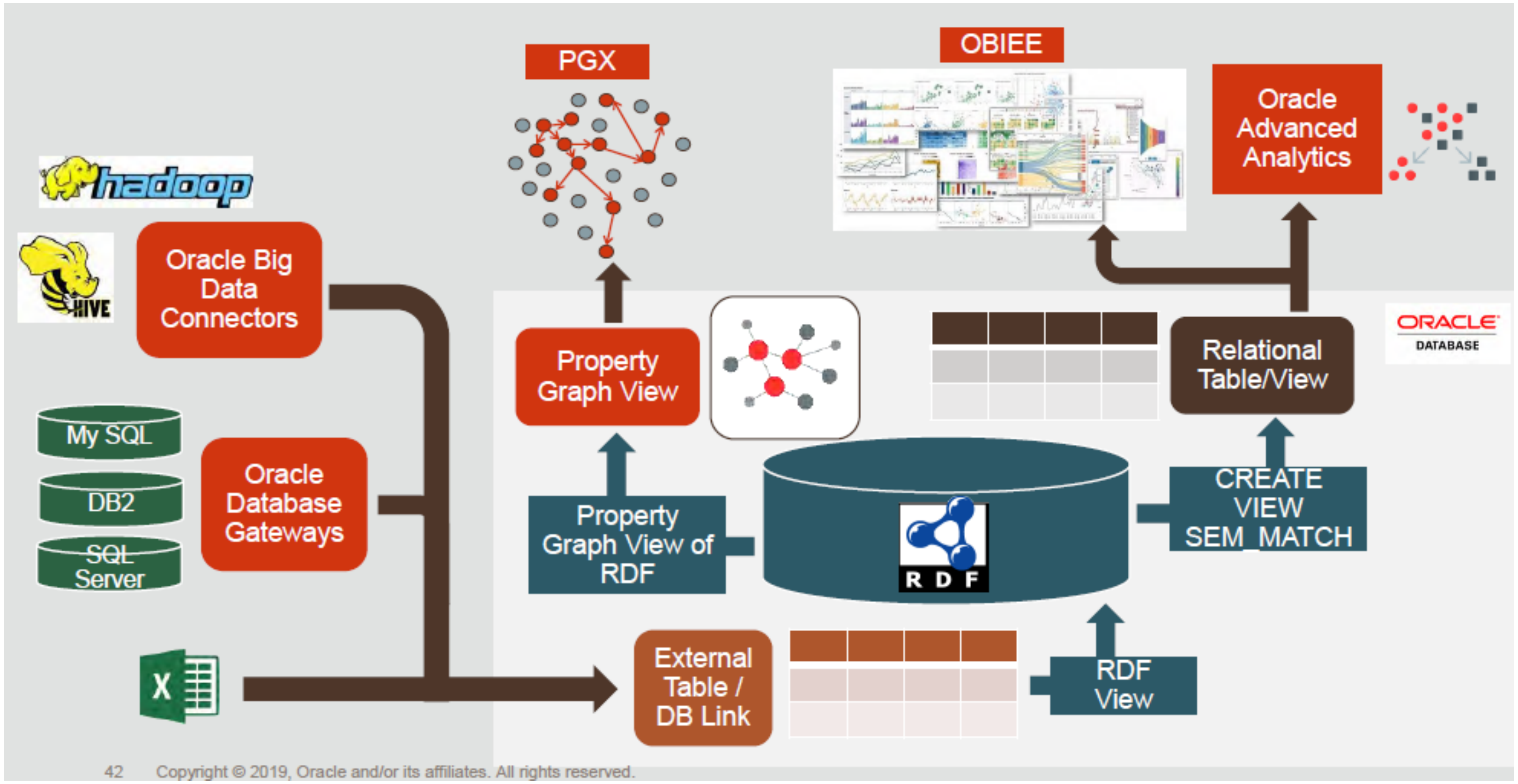
The property graph data model*



- Schema

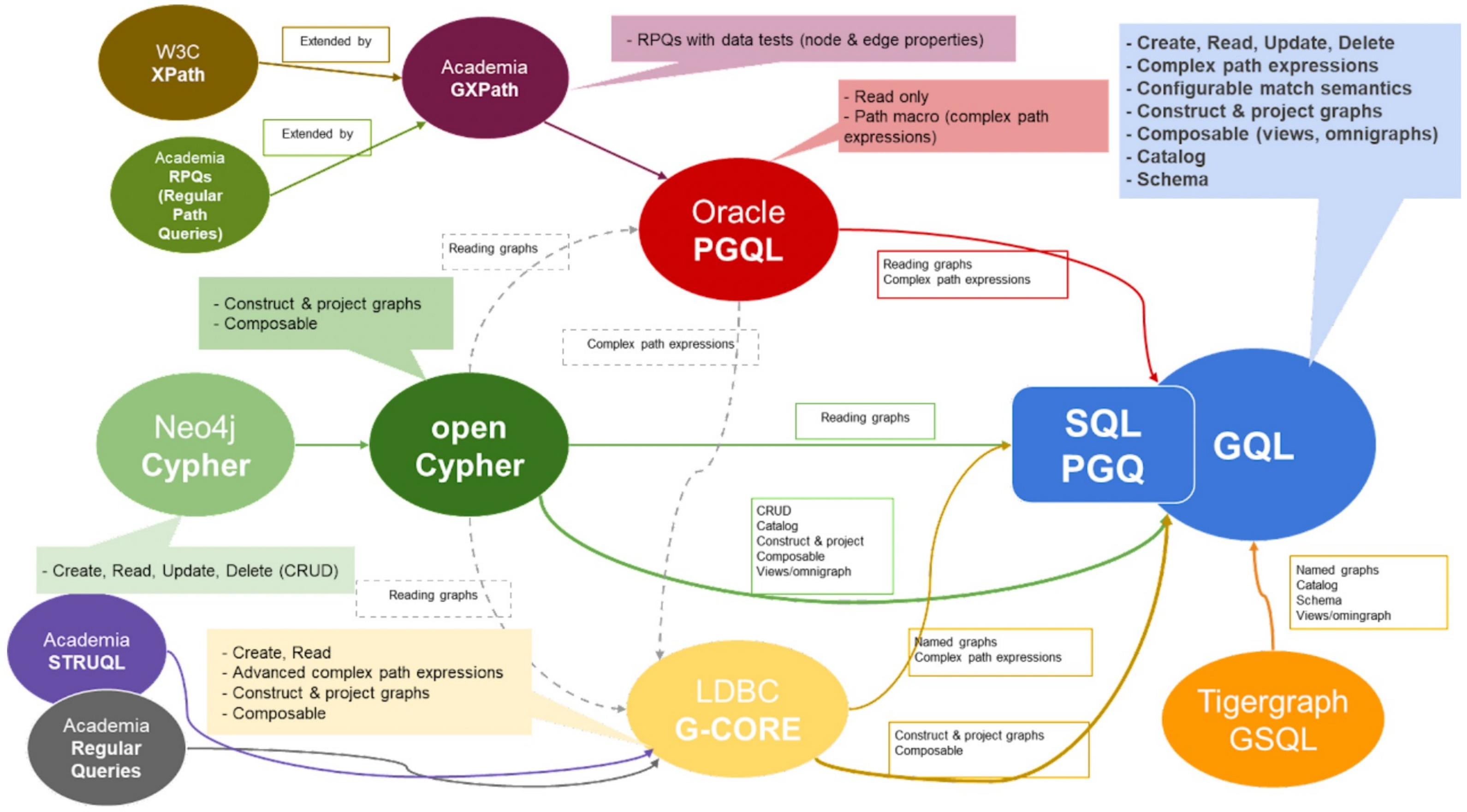
1. *$T_N \subset \boldsymbol{L}$ is a finite set of labels representing node types;*
2. *$T_E \subset \boldsymbol{L}$ is a finite set of labels representing edge types, satisfying that T_E and T_N are disjoint;*
3. *$\beta : (T_N \cup T_E) \times \boldsymbol{P} \to \boldsymbol{T}$ is a partial function that defines the properties for node and edge types, and the datatypes of the corresponding values;*
4. *$\delta : (T_N, T_N) \to \mathrm{SET}^+(T_E)$ is a partial function that defines the edge types allowed between a given pair of node types.*

Using both models for analytics (example)



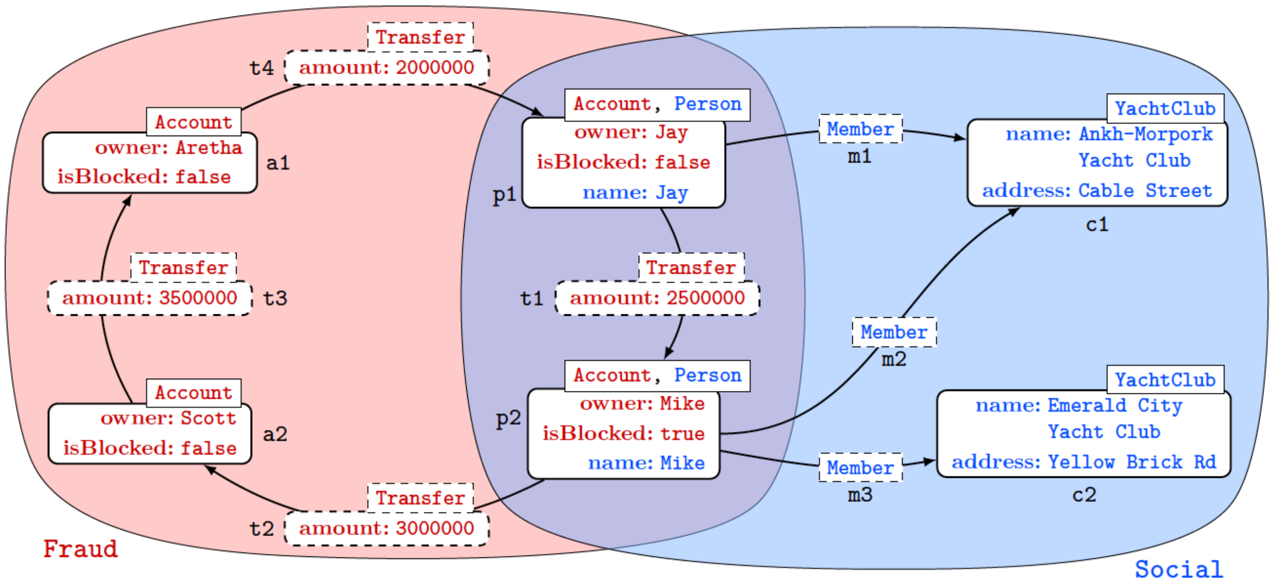
- Perry, M. Introduction to RDF Graph for Oracle Database 19c. Architecture and Overview (2019)

Graph query languages*



- https://www.gqlstandards.org/what-is-a-gql-standard

GQL *



```
USE Fraud
MATCH (x) -[z:Transfer WHERE z.amount>1000000]-> (y WHERE y.isBlocked=true)
RETURN x.owner AS sender, y.owner AS recipient
```

* Francis et al.. A Researcher's Digest of GQL, ICDT 2023

Agenda

- **Introduction and motivation**
  - Why graph databases?
  - Graph database models
    - Property graphs
    - RDF graphs
  - Temporal databases
  - Temporal graph databases

Temporal Databases

- Represent and manage time-varying information
- Several ways to interpret the time frame
  - Valid time (VT): Time when a record is valid in the real world
    - E.g., captures when a salary was paid to an employee
    - Supplied by the user
  - Transaction time (TT): Time when a fact is stored in the DB
    - Begins at the time when a record is inserted or updated, and ends when the record is deleted or updated
    - Generated by the database system
  - Bitemporal time (BT): Valid and transaction times combined
- Lifespan (LS): Time when an object or relationship exists,
  - e.g., duration of a project
- Granularity represents the minimal division of the timeline

Temporal Databases

- DBMSs provide limited support for dealing with time-varying data
- Many of them only provide data types for encoding dates or timestamps
- SQL standard: temporal support, partially implemented in most DBMSs
- SQL must be used for querying time-varying data, not an easy task
- Example: a temporal database:

Employee

SSN	FirstName	LastName	BirthDate	Address

Salary

SSN	Amount	FromDate	ToDate

Affiliation

SSN	DNumber	FromDate	ToDate

WorksOn

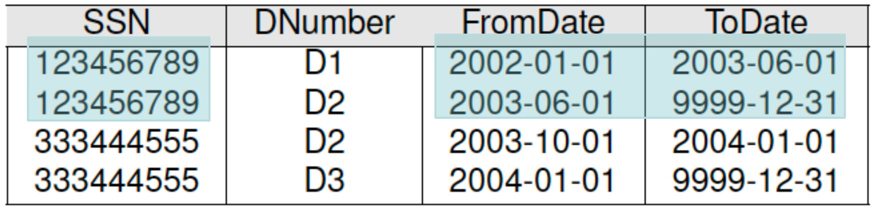
SSN	PNumber	FromDate	ToDate

Controls

DNumber	PNumber	FromDate	ToDate

- FromDate and ToDate: Indicate when the information in a row

Operations in Temporal Databases



SSN	DNumber	FromDate	ToDate
123456789	D1	2002-01-01	2003-06-01
123456789	D2	2003-06-01	9999-12-31
333444555	D2	2003-10-01	2004-01-01
333444555	D3	2004-01-01	9999-12-31

- *Given the table Affiliation obtain the periods of time when an employee has worked for the company, independently of the department*
- This is called a temporal projection
- Not easy to express in SQL
  - Note: first and last two rows are **value equivalent**, equal on all their columns except for FromDate and ToDate
  - Result must be coalesced: combining several value-equivalent rows into one provided that their time periods overlap

Operations in Temporal Databases

- *Given the table Affiliation obtain the periods of time when an employee has worked for the company, independently of the department*
- Non coalesced result

SSN	FromDate	ToDate
123456789	2002-01-01	2003-06-01
123456789	2003-06-01	9999-12-31
333444555	2003-10-01	2004-01-01
333444555	2004-01-01	9999-12-31

- Coalesced result

SSN	FromDate	ToDate
123456789	2002-01-01	9999-12-31
333444555	2003-10-01	9999-12-31

- Coalescing is an expensive operation, requires expert SQL knowledge

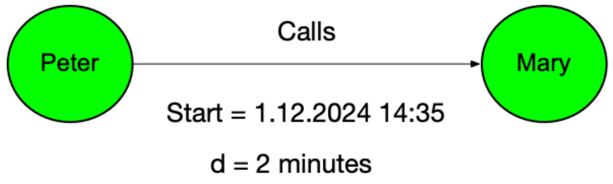
Agenda

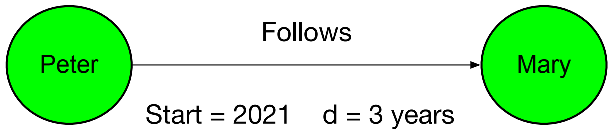
- **Introduction and motivation**
  - Why graph databases?
  - Graph database models
    - Property graphs
    - RDF graphs
  - Temporal databases
  - Temporal graph databases

Temporal Graph Databases

- Typically, graphs assumed to be static (non-temporal)
- Changes may occur in a property graph as the world they represent evolves across time

1. *Phone call network*. Each vertex can represent a person (or a phone #), an edge (u, v, t, d) tells that u called v at time t, with duration d; new nodes and edges are added frequently, and the properties of u or v may change over time
2. *Social networks*. Each vertex models a person, organization, etc.; an edge (u, v, t, d) represents a relationship between u and v (e.g., u *follows* v, u *is a friend of* v) at time t which lasts d





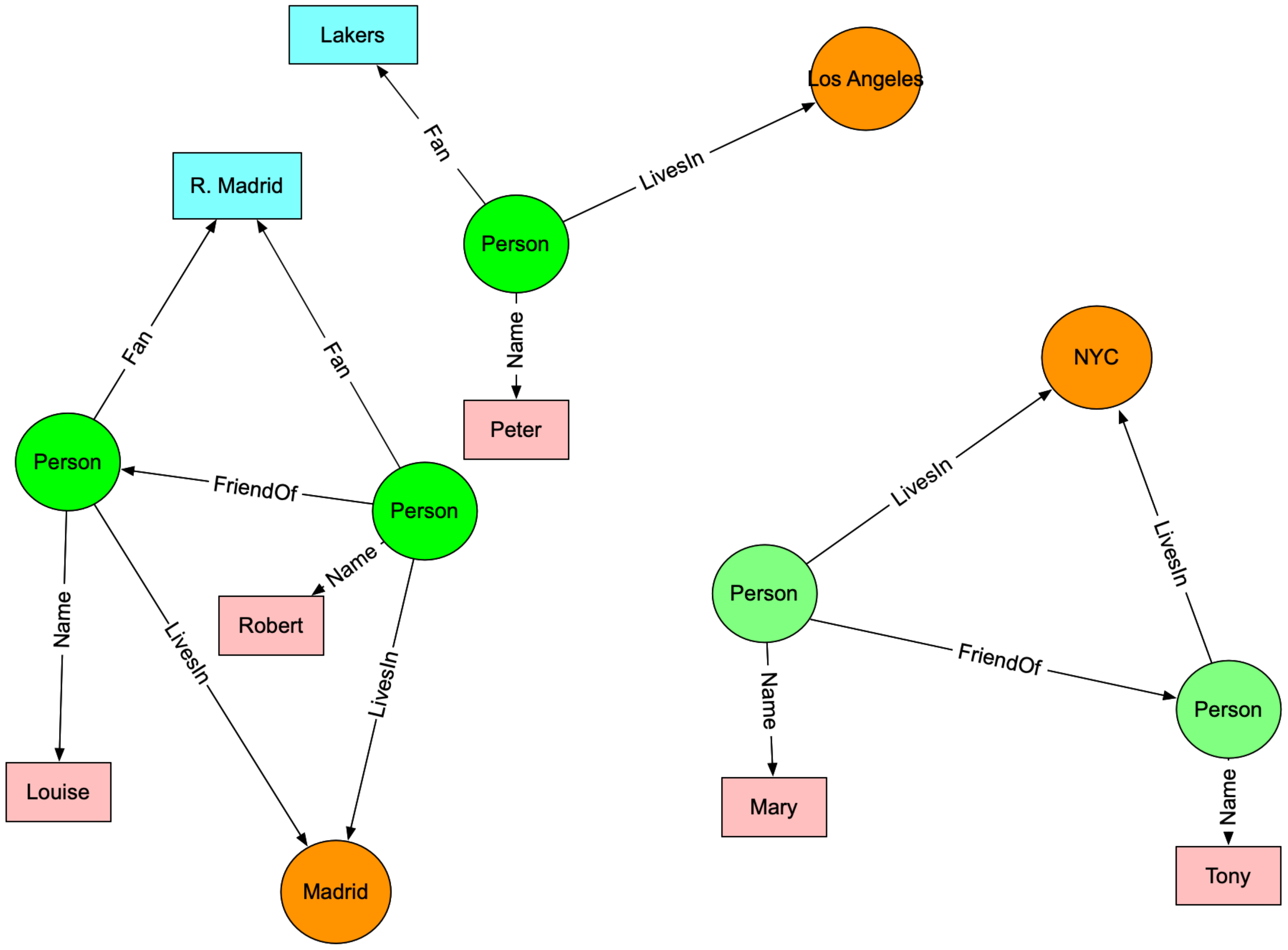
Temporal Graph Databases

3. *Transportation networks*. Each vertex represents a location, an edge (u, v, t, d) a road segment from u to v, existing since time t, and whose interval of existence is d
4. *Travel schedules*. Each vertex in a graph represents a location, and an edge (u, v, t, d) is a trip (flight, bus, etc.) from u to v departing at time t, whose duration is d

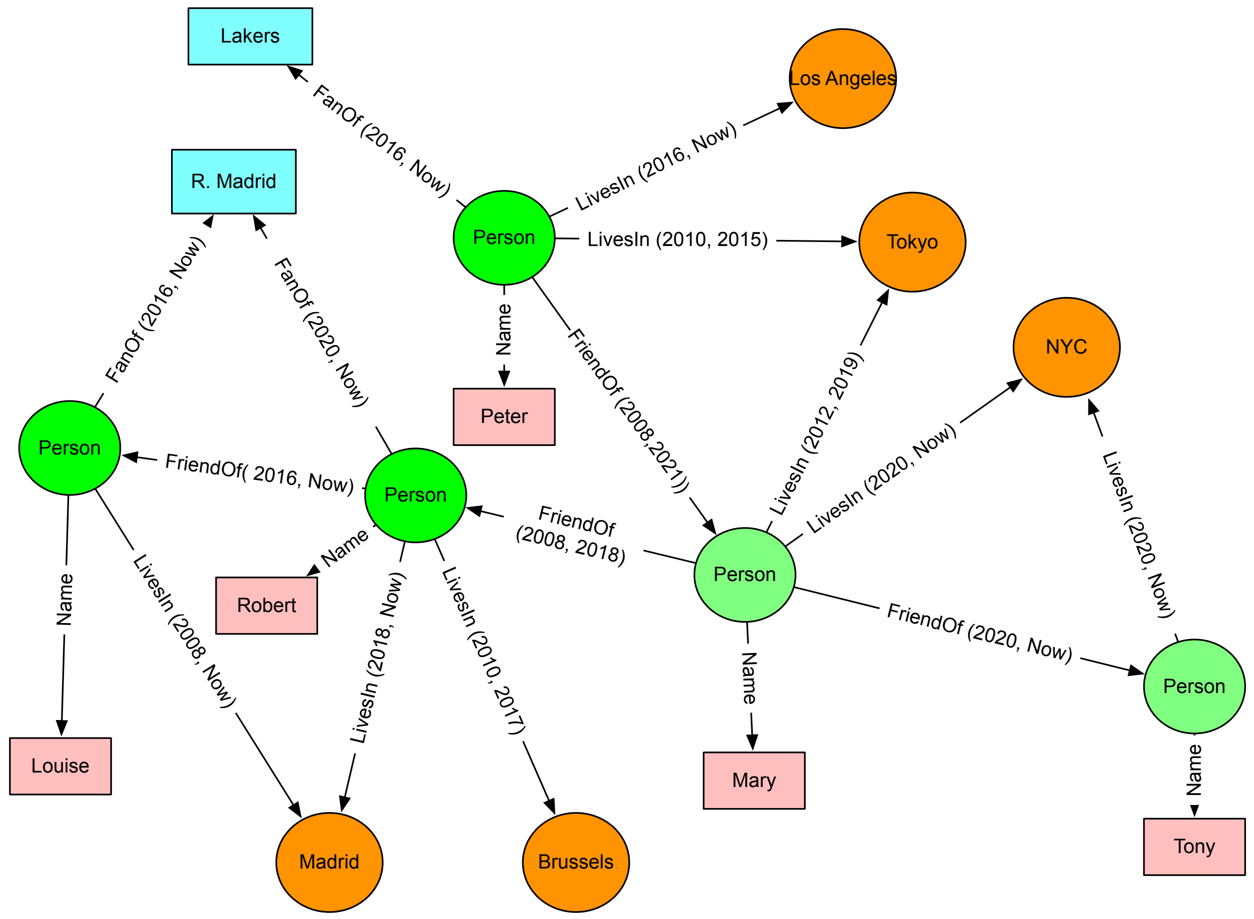
Temporal graph literature is limited

- Addresses mostly cases 1 (without changes in properties) and 4
- Cases 2 and 3 require an approach over PGs along the lines of the temporal database theory
- We study how temporal databases concepts can be applied to graph databases, to model, store, and query temporal graphs

A Graph Database



A (Simplified) Temporal Graph Database



Agenda

- Introduction and motivation
- **Temporal Graph Databases**
- Implementation
- Temporal Graphs in Transportation Networks
- Conclusion

Agenda

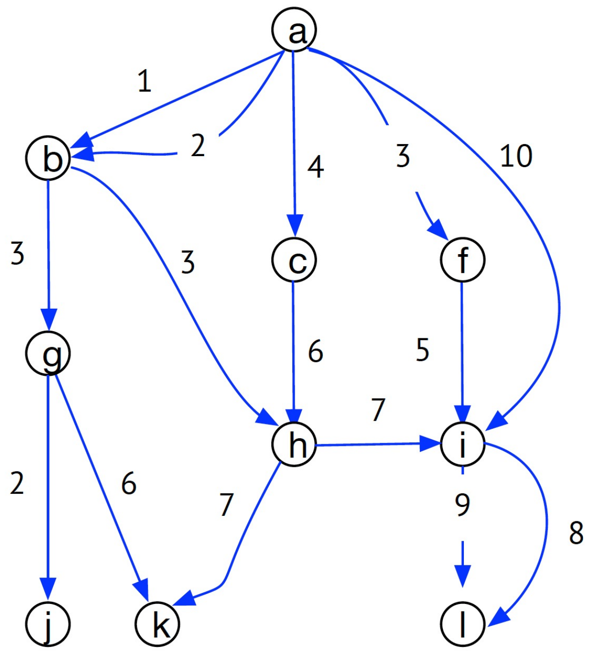
- **Temporal Graph Databases**
  - Temporal graph data models
  - An abstract data model
  - The T-GQL query language

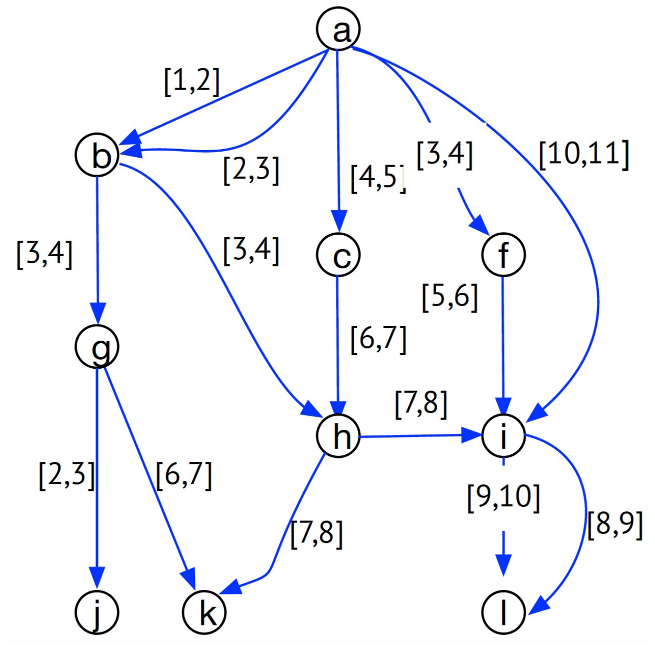
Agenda

- **Temporal Graph Databases**
  - Temporal graph data models
  - An abstract data model
  - The T-GQL query language

Temporal graph data models

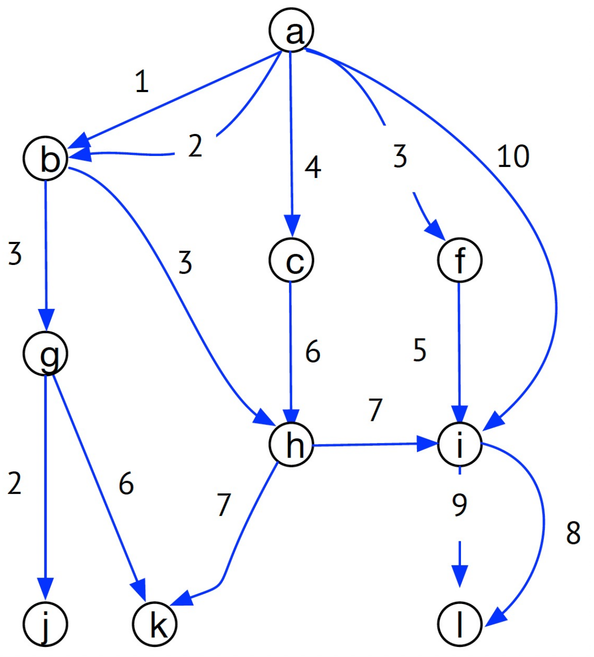
- We classify data models in the literature of temporal graphs as:
  - Duration-labeled temporal graphs (DLTG)
  - Interval-labeled temporal graphs (ILTG)
  - Snapshot-based temporal graphs (SBTG)





Temporal graph data models

- We classify data models in the literature of temporal graphs as:
  - Duration-labeled temporal graphs (DLTG)

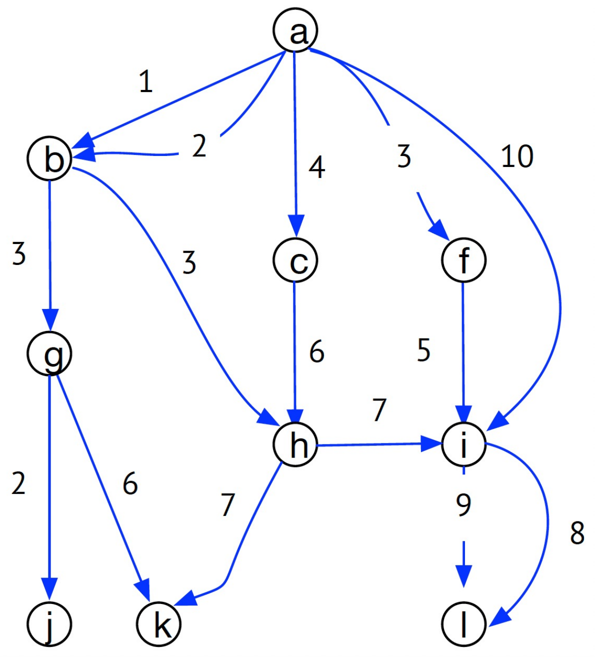


- Studied by Wu et al. *
- A node represented as a string (nodes not annotated with properties), and the edges labeled with a value representing the duration of the relationship (in the figure, duration λ = 1
- Each edge e = (u, v, t, λ) represents a relationship from a vertex u to another vertex v starting at time t with a duration λ

* Wu et al. Path problems in temporal graphs. Proic. VLDB, 2014, Hangzhou, China. http://www.vldb.org/pvldb/vol7/p721-wu.

Temporal graph data models

- We classify data models in the literature of temporal graphs as:
  - Duration-labeled temporal graphs (DLTG)



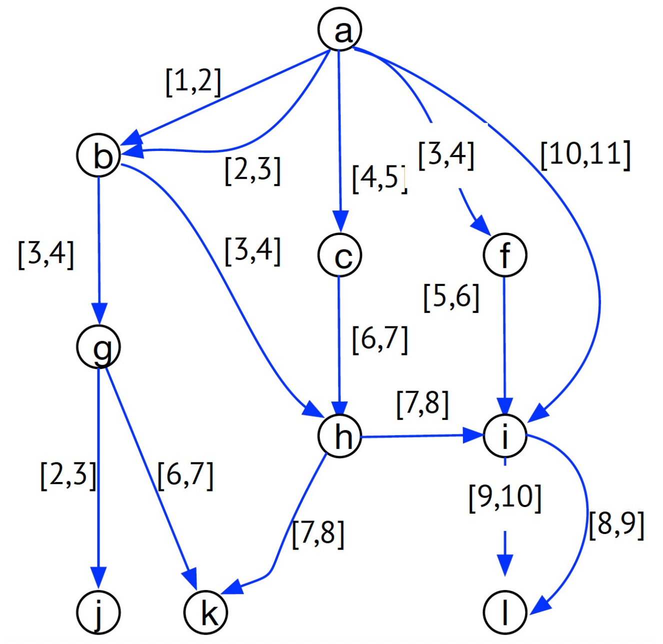
- Four different forms of 'shortest' paths called minimum temporal paths:

1. Earliest-arrival path path: earliest arrival time from a source x to a target y
2. Latest-departure path: latest departure time starting from x in order to reach y at a given time
3. Fastest path: goes from x to y in the minimum elapsed time
4. Shortest path: shortest from x to y in terms of number of hops

* Wu et al. Path problems in temporal graphs. Proic. VLDB, 2014, Hangzhou, China. http://www.vldb.org/pvldb/vol7/p721-wu.

Temporal graph data models

- We classify data models in the literature of temporal graphs as:
  - Interval-labeled temporal graphs (ILTG): A temporal label is defined over the database objects

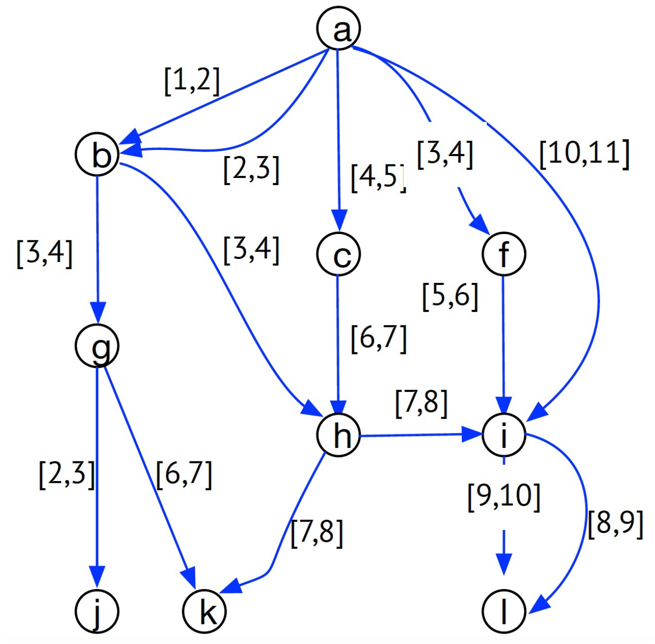


- Label over database objects
- Studied in Campos et al.*
- Defined as a graph where each edge e = (u, v, I) represents a relationship from a vertex u to another vertex v, valid during a closed-open interval I= [t_s,t_e)
- Also nodes and properties are annotated with their validity intervals

* Campos, A. et al Towards temporal graph databases. Proceedings of the AMW 2016, Panama City, Panama.

Temporal graph data models

- We classify data models in the literature of temporal graphs as:
  - Snapshot temporal graphs*: defined as a sequence of snapshots



- A temporal graph G[ti, tj] in a time interval [ti , tj], is a sequence {Gt_i, Gt_{i+1}, …, Gt_j } of graph snapshots

* K. Semertzidis and E. Pitoura, “Top-k durable graph pattern queries on temporal graphs,” IEEE Trans. Knowl. Data Eng.,

Temporal graph data models

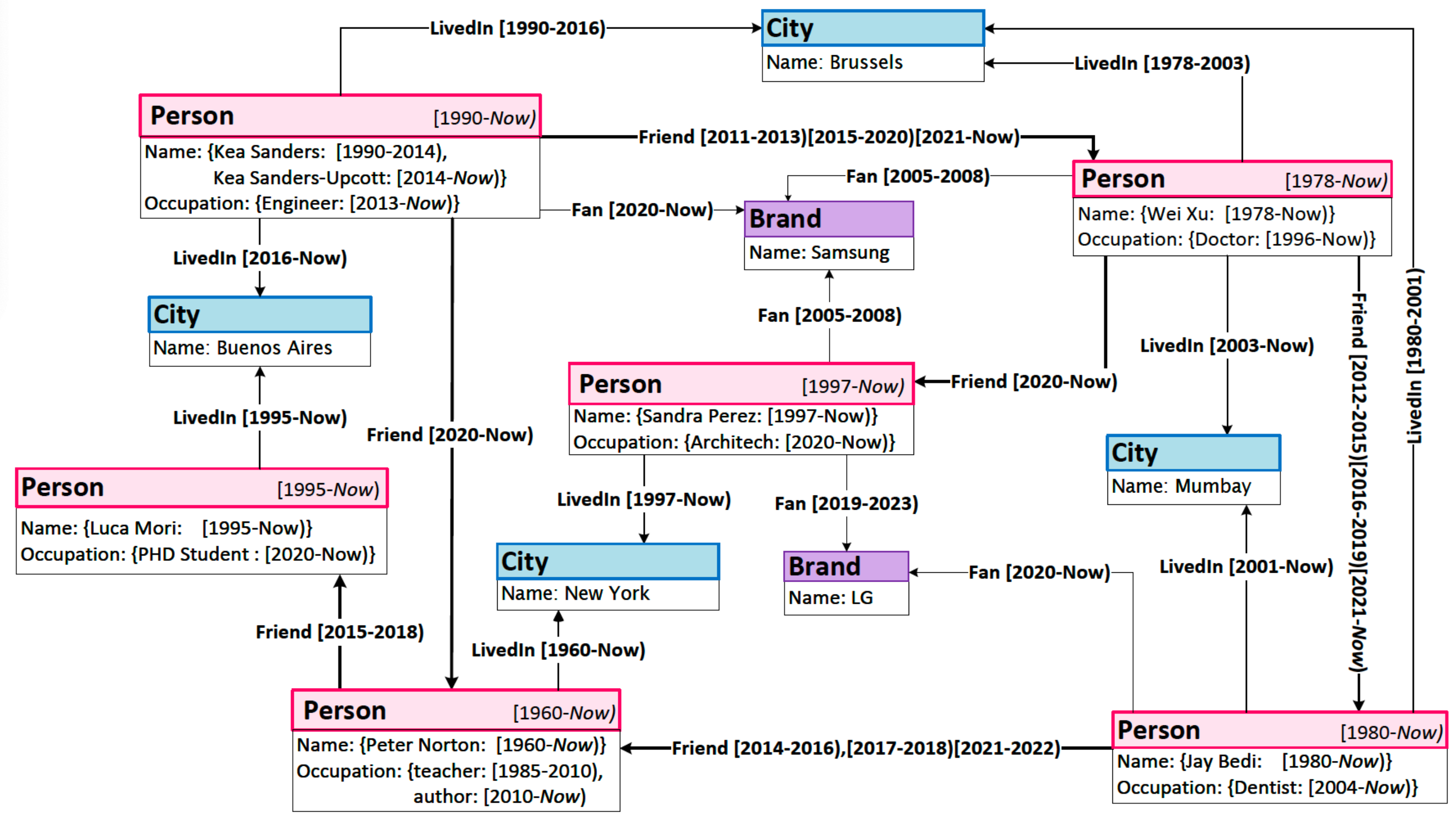
- Other work:

- Byun et al. ChronoGraph: Enabling temporal graph traversals for efficient information difusion analysis over time. IEEE Trans. Knowl. Data Eng., 32(3):424--437, 2020. https://doi.org/10.1109/TKDE.2019.2891565
- Byun, J. (2022). Enabling time-centric computation for efficient temporal graph traversals from multiple sources. IEEE Trans. Knowl. Data Eng., 34 (4), 1751–1762. https://doi.org/10.1109/TKDE.2020.3005672
- C. Cattuto, A. Panisson, and M. Quaggiotto, “Representing time dependent graphs in Neo4j.” https://github.com/SocioPatterns/neo4j-dynagraph/wiki/Representing-time-dependent-graphs-in-Neo4j, 2013.
- T. Johnson, Y. Kanza, L. V. S. Lakshmanan, and V. Shkapenyuk, “Nepal: a path query language for communication networks,” NDA@SIGMOD 2016

Agenda

- **Temporal Graph Databases**
  - Temporal graph data models
  - An abstract data model
  - The T-GQL query language

An abstract model for temporal networks


City
Name: Brussels
LivedIn [1990-2016]
LivedIn [1978-2003]
Person [1990-*Now*)
Name: {Kea Sanders: [1990-2014),
Kea Sanders-Upcott: [2014-*Now*)}
Occupation: {Engineer: [2013-*Now*)}
Friend [2011-2013)[2015-2020)[2021-Now)
Fan [2005-2008)
Person [1978-*Now*)
Name: {Wei Xu: [1978-Now)}
Occupation: {Doctor: [1996-Now)}
Fan [2020-Now)
Brand
Name: Samsung
LivedIn [2016-Now)
City
Name: Buenos Aires
Fan [2005-2008)
LivedIn [2003-Now)
Friend [2012-2015)[2016-2019)[2021-Now)
LivedIn [1980-2001)
Person [1997-*Now*)
Name: {Sandra Perez: [1997-Now)}
Occupation: {Architech: [2020-Now)}
Friend [2020-Now)
LivedIn [1995-Now)
Friend [2020-Now)
City
Name: Mumbay
Person [1995-*Now*)
Name: {Luca Mori: [1995-Now)}
Occupation: {PHD Student : [2020-Now)}
LivedIn [1997-Now)
Fan [2019-2023)
City
Name: New York
Brand
Name: LG
Fan [2020-Now)
LivedIn [2001-Now)
Friend [2015-2018)
LivedIn [1960-Now)
Person [1960-*Now*)
Name: {Peter Norton: [1960-*Now*)}
Occupation: {teacher: [1985-2010),
author: [2010-*Now*)
Friend [2014-2016),[2017-2018)[2021-2022)
Person [1980-*Now*)
Name: {Jay Bedi: [1980-*Now*)}
Occupation: {Dentist: [2004-*Now*)}

An abstract model for a temporal networks

- **Data structure**
- A directed property graph G(N, E)   N, E, nodes and directed edges
- Nodes are labeled with the type of entity they represent
- Interval represents lifespan
- Attributes: static and temporal
- Names of the relationships are associated with a set of intervals
- A special value *Now* is used to tell that the node is valid at the current time

- Also a set of **constraints**

An abstract model for a temporal networks

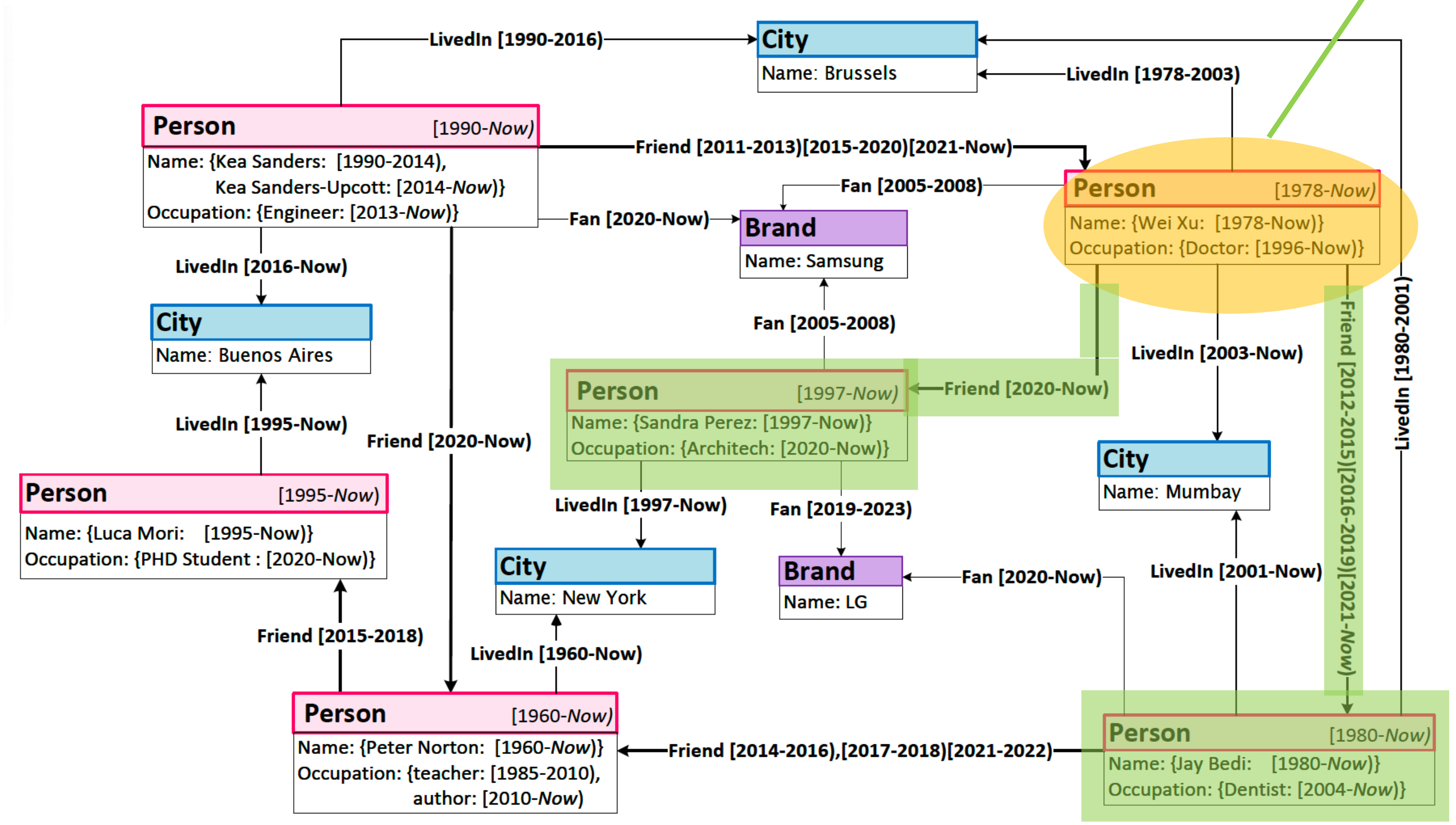
- **Constraints**
- All edges with the same label (i.e, representing the same relationship type), between the same pair of nodes, are coalesced
- The intersection of the nodes’ intervals must include their edge intervals
- A node’s interval includes the union of the intervals of a temporal attribute
- Intervals are time-ordered, maximal, and non-overlapping

Querying the abstract model

- **Query 1** *List the friends of Wei Xu.* Non temporal. Answer: Jay Bedi, Sandra Perez, the persons directly connected to Wei via a *Friend* relation
- **Query 2** *Who where the friends of Wei Xu in 2021?* Answer: Sandra Perez, Jay
- **Query 3** *Where did the people who where friends of Kea between 2011 and 2013 live at that time?* Only the relationship with Wei is valid in [2011-2013). Answer: Mumbai
- **Query 4** *Who were friends of Kea while she was living in Buenos Aires?* Kea has been living in Buenos Aires since 2016, thus, any person that was a friend of Kea at any instant of the interval in [2016-Now). Answer: Wei and Peter
- **Query 5** *Where did Jay live when he and Sandra followed the same brands?* Jay in [2020, Now) and Sandra in [2019,2023) followed LG. Intersection: [2020-2023). Answer: Mumbai

Querying the abstract model

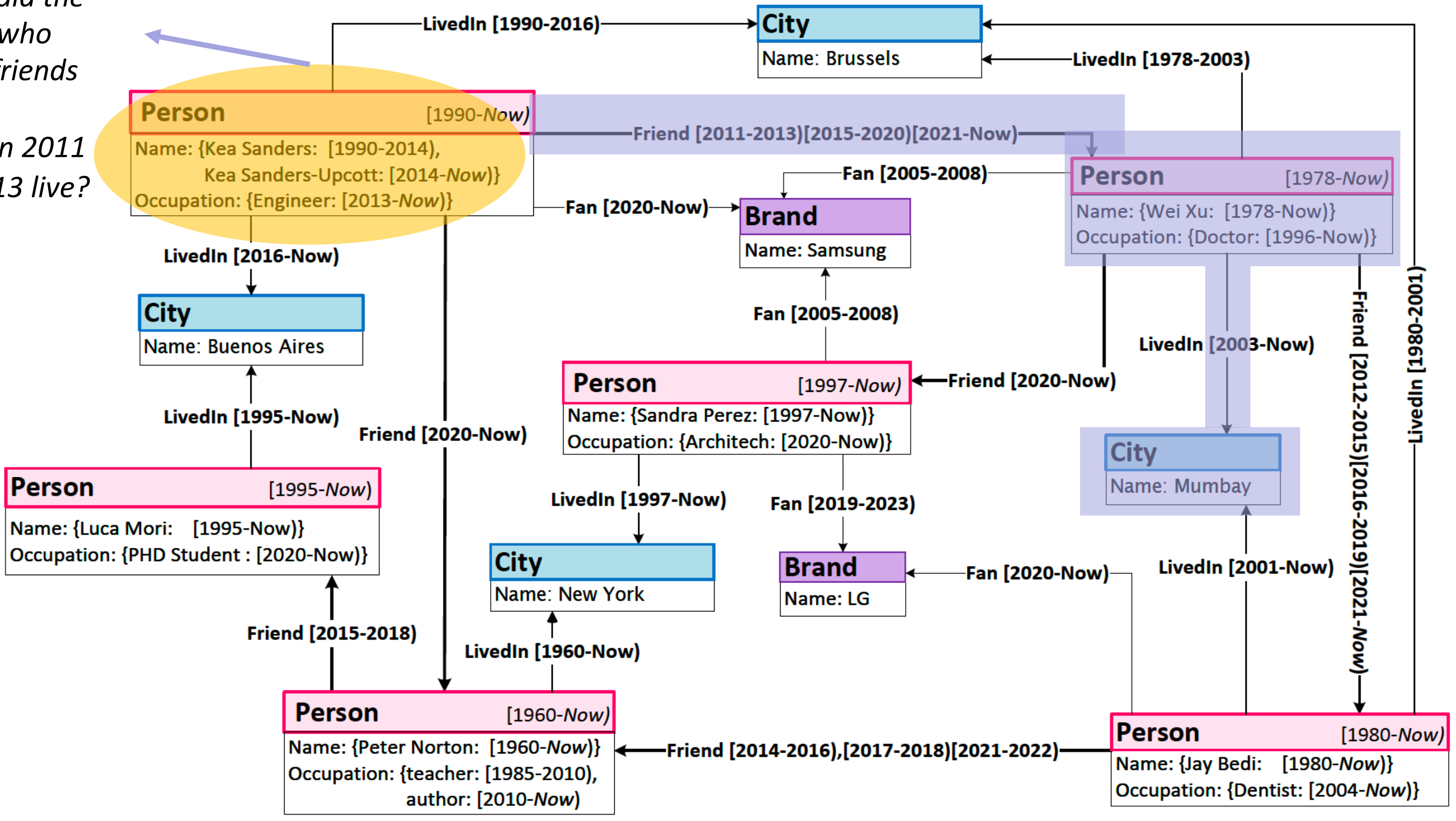
Query 2:
Friends of Wei Xu in 2021



City
Name: Brussels
LivedIn [1990-2016)
LivedIn [1978-2003)
Person [1990-Now)
Name: {Kea Sanders: [1990-2014), Kea Sanders-Upcott: [2014-Now)}
Occupation: {Engineer: [2013-Now)}
Friend [2011-2013)[2015-2020)[2021-Now)
Fan [2005-2008)
Fan [2020-Now)
Brand
Name: Samsung
Person [1978-Now)
Name: {Wei Xu: [1978-Now)}
Occupation: {Doctor: [1996-Now)}
LivedIn [2016-Now)
City
Name: Buenos Aires
Fan [2005-2008)
LivedIn [2003-Now)
Friend [2012-2015)[2016-2019)[2021-Now)
LivedIn [1980-2001)
Person [1997-Now)
Name: {Sandra Perez: [1997-Now)}
Occupation: {Architech: [2020-Now)}
Friend [2020-Now)
LivedIn [1995-Now)
Friend [2020-Now)
City
Name: Mumbay
Person [1995-Now)
Name: {Luca Mori: [1995-Now)}
Occupation: {PHD Student : [2020-Now)}
LivedIn [1997-Now)
Fan [2019-2023)
City
Name: New York
Brand
Name: LG
Fan [2020-Now)
LivedIn [2001-Now)
Friend [2015-2018)
LivedIn [1960-Now)
Person [1960-Now)
Name: {Peter Norton: [1960-Now)}
Occupation: {teacher: [1985-2010), author: [2010-Now)}
Friend [2014-2016),[2017-2018)[2021-2022)
Person [1980-Now)
Name: {Jay Bedi: [1980-Now)}
Occupation: {Dentist: [2004-Now)}

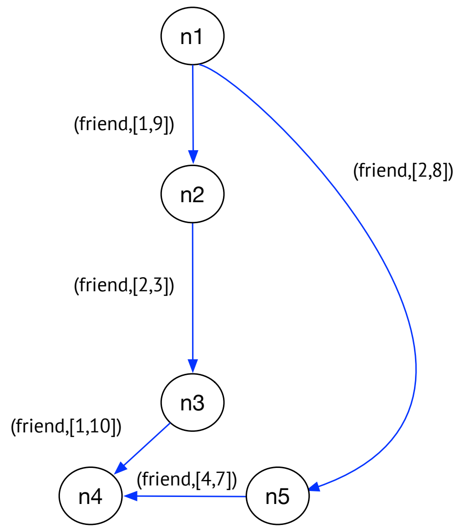
Querying the abstract model

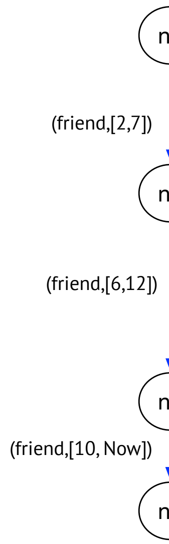
Query 3:
Where did the people who where friends of Kea between 2011 and 2013 live?

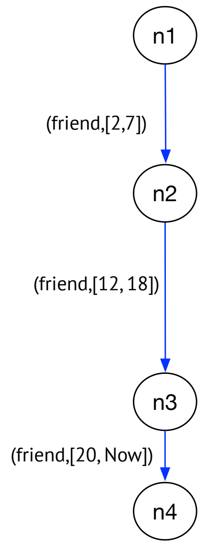


Person [1990-Now)
Name: {Kea Sanders: [1990-2014), Kea Sanders-Upcott: [2014-Now)}
Occupation: {Engineer: [2013-Now)}
LivedIn [1990-2016)
City
Name: Brussels
LivedIn [1978-2003)
Friend [2011-2013][2015-2020][2021-Now)
Fan [2005-2008)
Person [1978-Now)
Name: {Wei Xu: [1978-Now)}
Occupation: {Doctor: [1996-Now)}
Fan [2020-Now)
Brand
Name: Samsung
LivedIn [2016-Now)
City
Name: Buenos Aires
Fan [2005-2008)
LivedIn [2003-Now)
Friend [2012-2015][2016-2019][2021-Now)
LivedIn [1980-2001)
Person [1997-Now)
Name: {Sandra Perez: [1997-Now)}
Occupation: {Architech: [2020-Now)}
Friend [2020-Now)
LivedIn [1995-Now)
Friend [2020-Now)
City
Name: Mumbay
Person [1995-Now)
Name: {Luca Mori: [1995-Now)}
Occupation: {PHD Student : [2020-Now)}
LivedIn [1997-Now)
Fan [2019-2023)
City
Name: New York
Brand
Name: LG
Fan [2020-Now)
LivedIn [2001-Now)
Friend [2015-2018)
LivedIn [1960-Now)
Person [1960-Now)
Name: {Peter Norton: [1960-Now)}
Occupation: {teacher: [1985-2010), author: [2010-Now)}
Friend [2014-2016),[2017-2018][2021-2022)
Person [1980-Now)
Name: {Jay Bedi: [1980-Now)}
Occupation: {Dentist: [2004-Now)}

Temporal paths





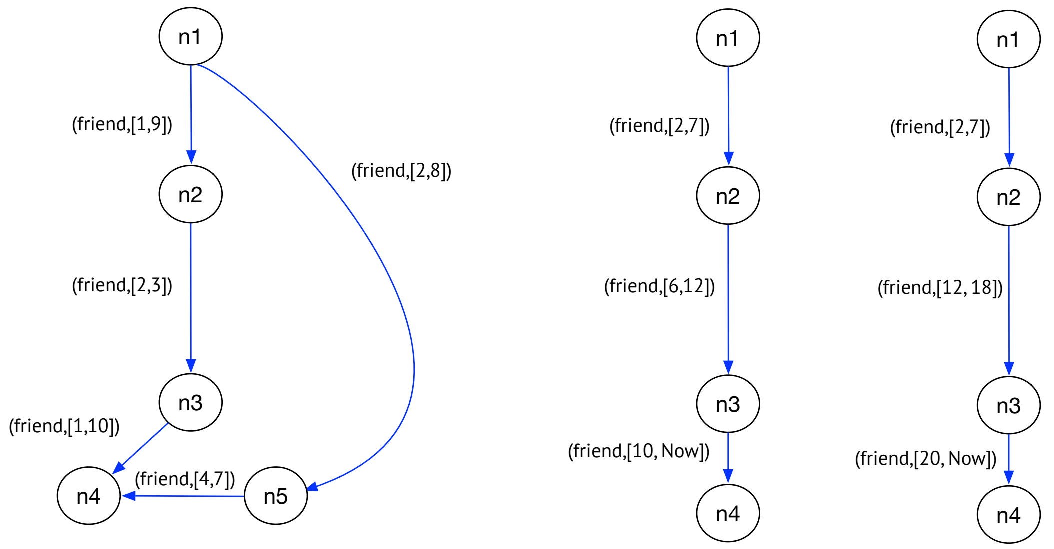


"When have Kea, Wei, Jay, Peter and Luca been friends simultaneously?":
Captured by the notion of *continuous path**

Left: two *Continuous paths*: (n1, n2, n3, n4; friend; [2, 3]); (n1, n5, n4; friend; [4, 7]) ($I = I_{n1} \cap I_{n2} \cap \ldots \cap I_{nk}$)
Center: a *Pairwise Continuous path* (overlapping intervals in the path)
Right: a *Consecutive path* (disjoint consecutive intervals)

* Rizzolo, F. and Vaisman, A., "Temporal XML: Modeling, indexing, and query processing," VLDB Journal, 2008

Temporal paths

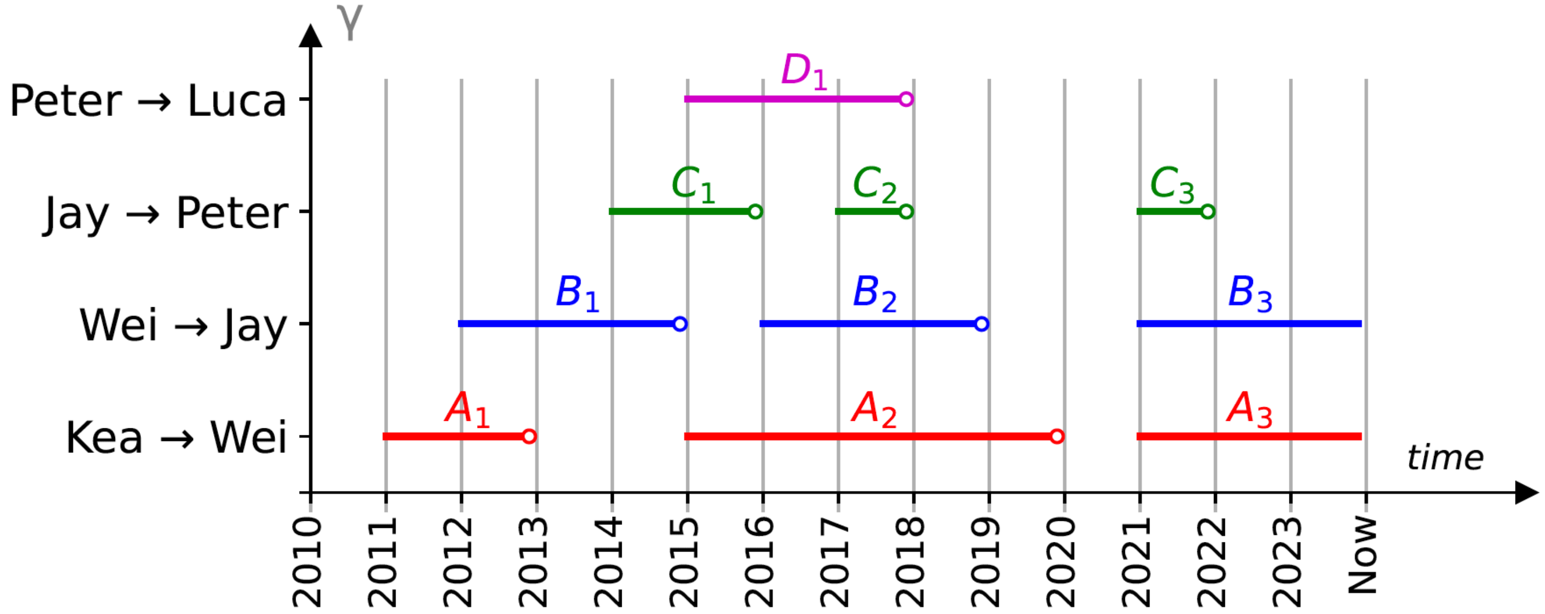


Questions

- Why did we come up with these paths?
- Are there any other interesting paths? How many?
- Can we characterize temporal paths in some way?

Temporal paths

- Let G(N, E) be a TNG. Let (N, r) ⊆ G(N, E) be a graph where N, is a finite set of nodes, r is a relationship (labeled r), r ⊆ N x N.
- Let e be an edge in (N, r), an interval *I* for e is maximal when for any interval I′, we have that I ⊆ *I′ => I = I′*
- Maximal intervals for the *Friend* relationship



Temporal path definition

- (N, r) a temporal network and $\propto$ a binary relation on temporal intervals.
- A *temporal $\propto$-path* in (N, r) is a structure (γ, I), where

  1. γ is a path in (N, r);
  2. *I is a sequence of intervals $I_1, \ldots, I_{k-1}$;*
  3. I_j is a maximal interval for every e_j (n_j, n_{j+1})
  4. $\propto(I_j, I_{j+1})$ holds between every pair of consecutive intervals

- Replacing condition (3) by:

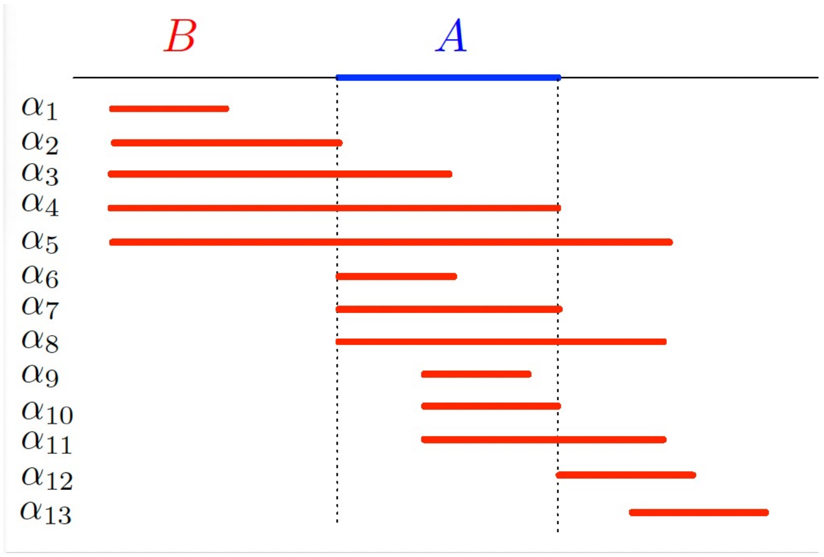
  3. I_j is an interval (not necessarily maximal) for e_j (n_j, n_{j+1}), for $j = 1, \ldots, k - 1$;

  We have a *temporal sub-$\propto$-path* in (N, r)

- Where does this $\propto$ come from?

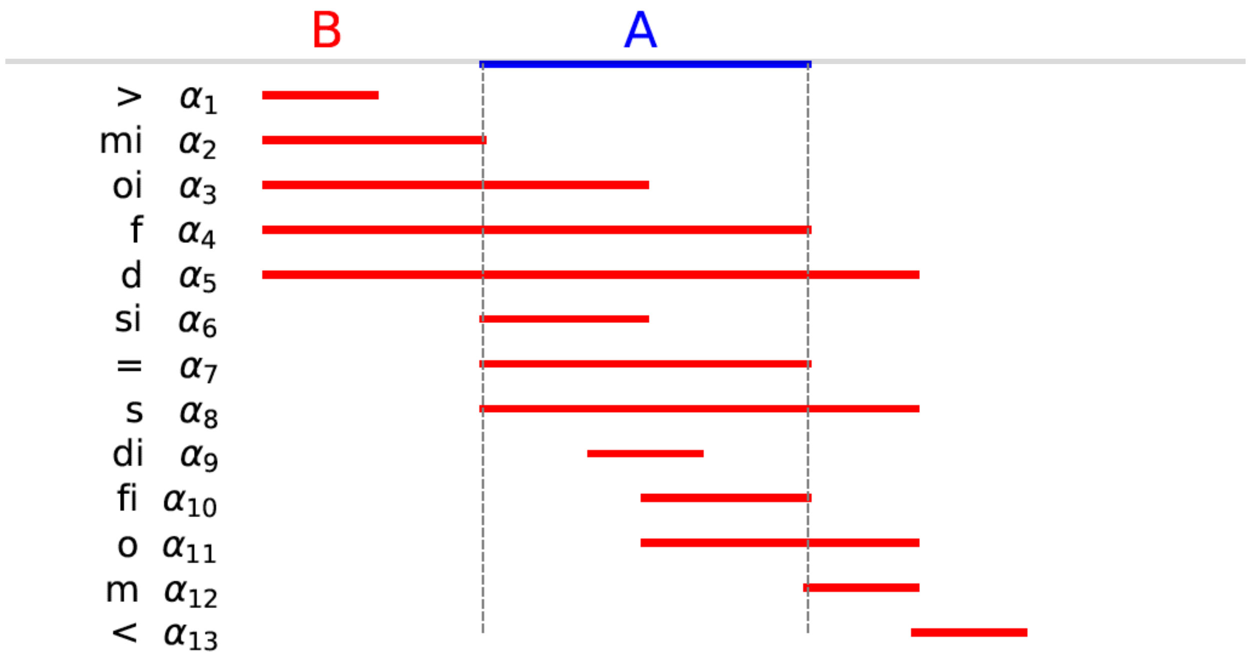
Allen's Algebra

- In 1983, James F. Allen described the possible relationships between two intervals on the real line (the timeline)
- Let A and B be such intervals. We denote their start and end points by s(A), s(B), e(A) and e(B), respectively
- So, we have A=[s(A),e(A)) and B=[s(B),e(B))
- The 13 possible arrangements of A and B, as given by Allen:



Allen’s Algebra

- We write αi(A,B), for i = 1, ..., 13, whenever the intervals A and B are in the relationships as depicted. We call α1, . . . , α13 **the base relations** of the Allen interval algebra.

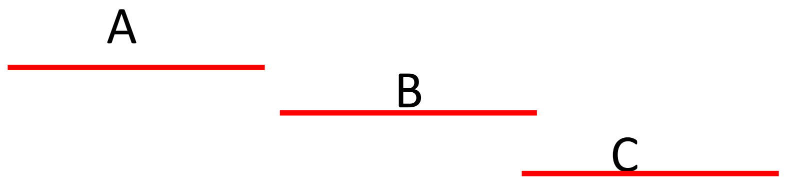


Allen’s Algebra

- Based on these thirteen base relations $\alpha_1, \ldots, \alpha_{13}$, Allen defined an interval algebra which we denote $\mathcal{A}$
- The base relations α1, . . . , α13 are exhaustive and pairwise disjoint, i.e., for any two given intervals, **exactly one of the thirteen relations holds**
- Also, the complement (or negation) of one of the α_i corresponds to the union (or disjunction) of the twelve remaining α_j (j <> i)
  - For example $!\alpha_1 = \alpha_2 \cup \alpha_3 \ldots \cup \alpha_{13}$
- We denote $\alpha_1 \cup \alpha_2 \cup \alpha_3 \cup \alpha_4 \cup \alpha_5(A, B)$, as $\alpha_{1\rightarrow 5}(A, B)$ or $\alpha_{1,2,3,4,5}(A, B)$

Allen's Algebra

- The algebra $\mathcal{A}$ includes the operations inverse (denoted $.^{-1}$), intersection (denoted $\cap$) and composition (denoted $\circ$) :
  - Inverse: $\alpha^{-1}(A, B)$ if and only if $\alpha(B, A)$;
  - Intersection: $\alpha \cap \beta\ (A, B)$ if and only if $\alpha(A, B)$ and $\beta(A, B)$; and
  - Composition: $\alpha \circ \beta\ (A, C)$ if and only if there exists an interval B such that $\alpha(A,B)$ and $\beta(B, C)$



$\alpha_{12}(AB) \circ \alpha_{12}(BC) = \alpha_{13}(A C)$
(α_{12} not transitive)

- The inverse relation, α_i^{-1}, in our notation, verifies that

$$\alpha_i^{-1} = \alpha_{14-i} \quad \text{for } i = 1, 2, \ldots, 13$$

https://ics.uci.edu/~alspaugh/cls/shr/allen.html

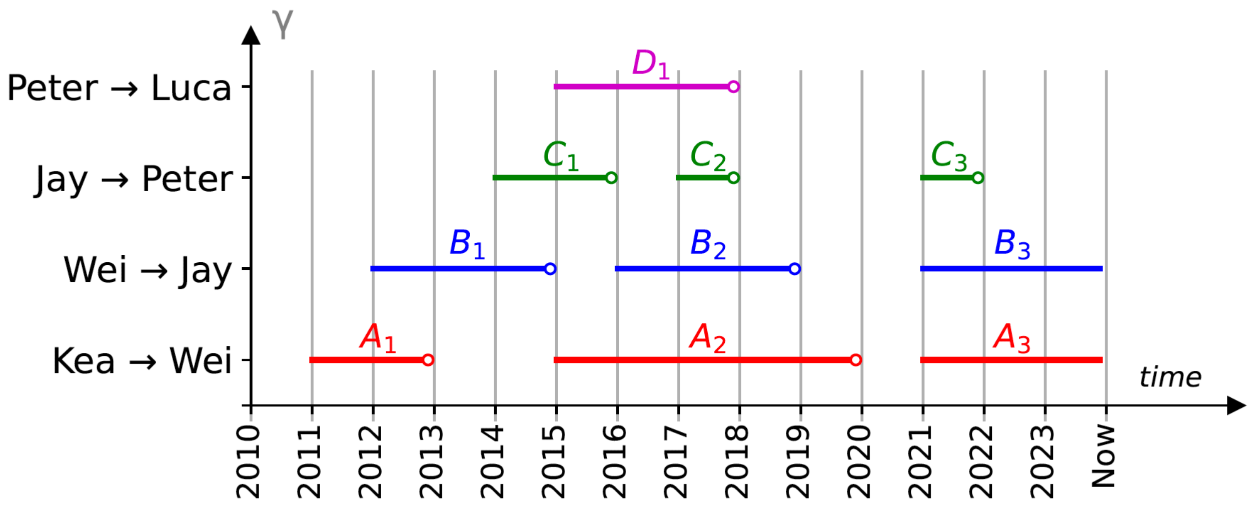
Krokhin, A, Jeavons, P. & Jonsson, P. (2003) : Reasoning about temporal relations : the maximal tractable subalgebras of Allen's interval algebra., Journal of the ACM., 50 (5). pp. 591-640.

Allen's Algebra

- Any element of the Allen algebra $\mathcal{A}$ can be written as a as union of its base relations
- Since compositions and negations can be written as unions and intersections, they can be expressed using negation and union
- Given that there are the thirteen base relations $\alpha_1, \ldots, \alpha_{13}$, $2^{13} = 8192$ unions are possible, and this is in fact the cardinality of $\mathcal{A}$

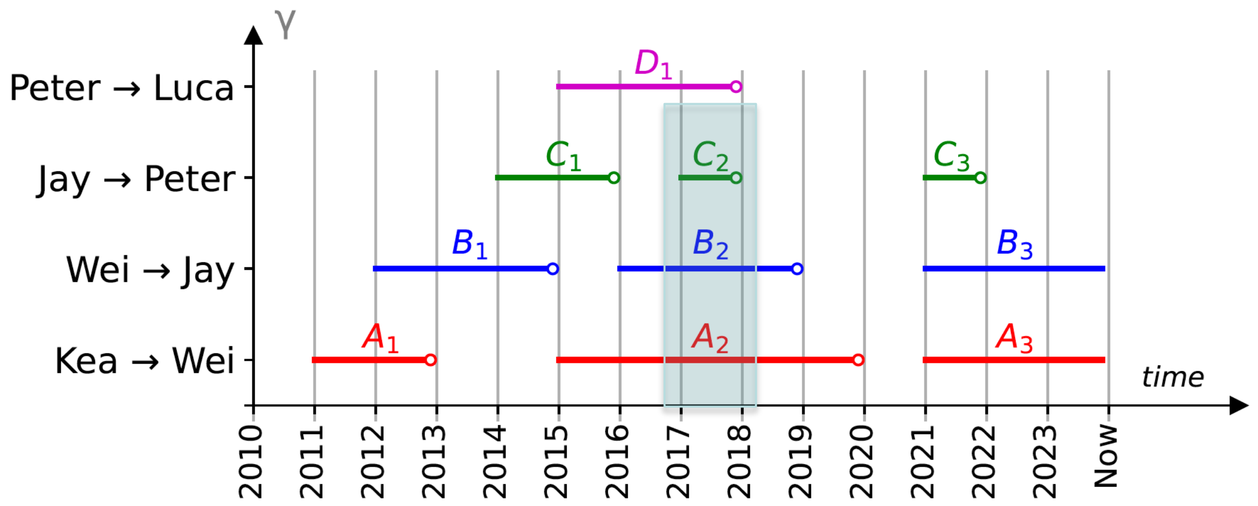
- Question: Are all of combinations useful? How can we prune this set?
- We will study (later) this over a real-world use case

Allen's algebra-based α-paths



- γ (A_2, B_2, C_2) is an α_9-path because $\alpha_9(A_2, B_2)$ and $\alpha_9(B_2, C_2)$ hold
- γ (A_3, B_3, C_2) is an $\alpha_7 \cup \alpha_1$-path because $\alpha_7(A_3, B_3)$ and $\alpha_1(B_3, C_2)$ hold = > an $\alpha_{1,7}$-path

Allen's algebra-based α-paths

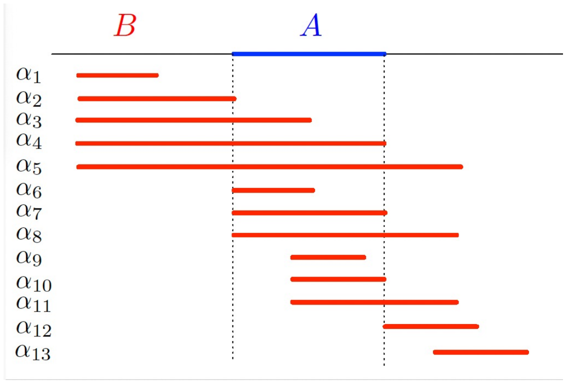


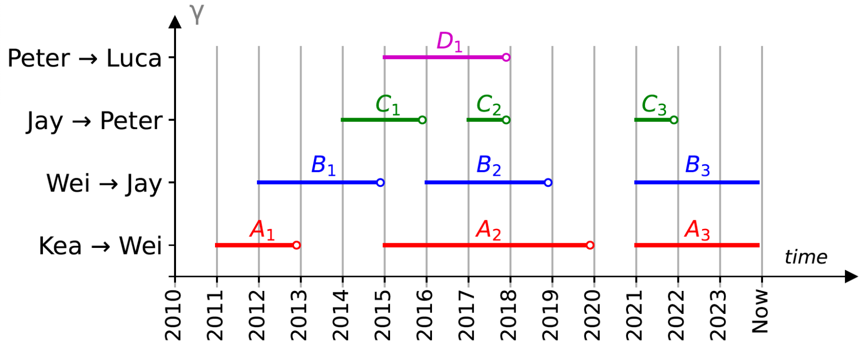
- γ (A_2, B_2, C_2) is an α_9-path because $\alpha_9(A_2, B_2)$ and $\alpha_9(B_2, C_2)$ hold
- γ (A_3, B_3, C_2) is an $\alpha_7 \cup \alpha_1$-path because $\alpha_7(A_3, B_3)$ and $\alpha_1(B_3, C_2)$ hold = > an $\alpha_{1,7}$-path
- If $A = B = C_2$, we have a sub- α_7-path but NOT an α_7-path

Examples of α-paths

- **Query 6.** *List all the α-paths between Kea and Luca during the interval [2015-2021)*

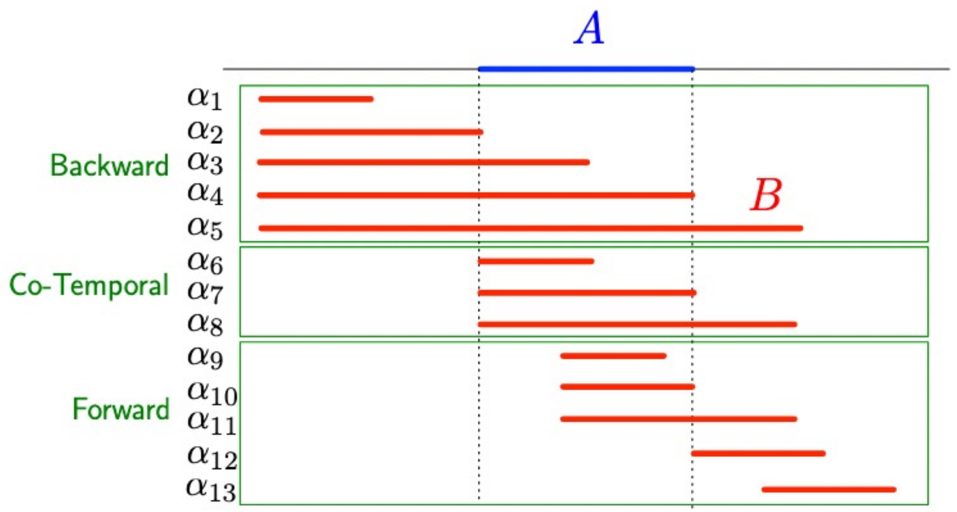
- Answer:
  - *An $\alpha_{2,9,10}$-path whose α relations are: $\alpha_9(A_2, B_2)$, $\alpha_2(B_2, C_1)$, $\alpha_{11}(C1, D1)$*
  - *An $\alpha_{4,9}$-path path whose α relations are $\alpha_9(A_2, B_2)$, $\alpha_9(B_2, C_2)$, $\alpha_4(C_2, D_1)$*





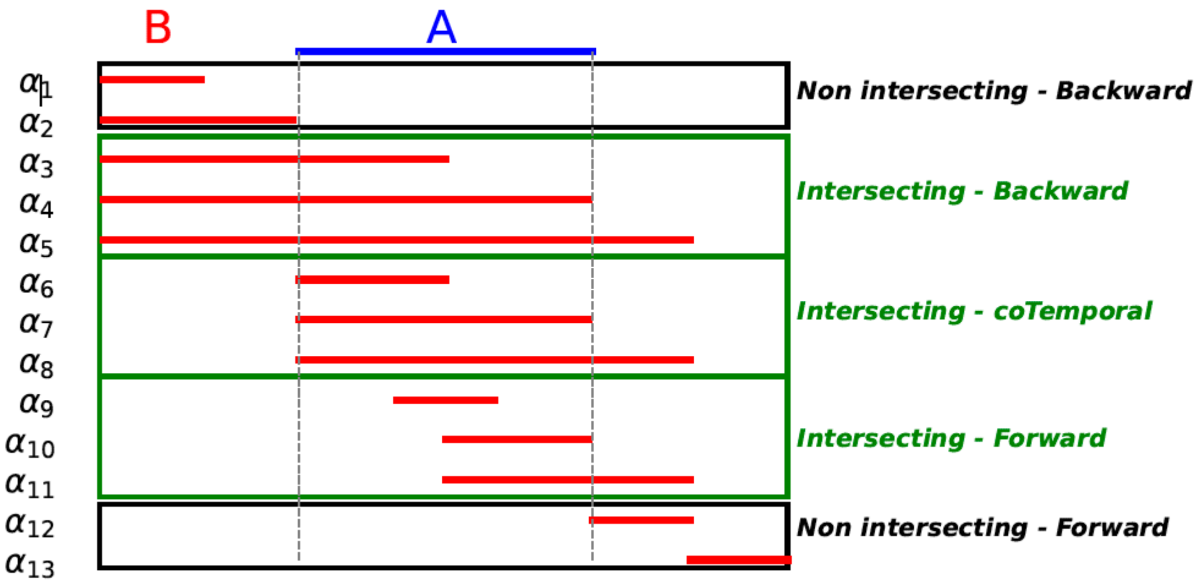
Classification of paths

- The base Allen relations can be divided into three groups, based on when the second interval in the relation starts, with respect to the first interval
  - Backward, containing the relations α_1, α_2 , α_3, α_4 , α_5
  - Co-Temporal, containing the relations α_6 α_7 , α_8
  - Forward, containing the relations α_9, α_{10} , α_{11}, α_{12} , α_{13}



Classification of paths

- Another classification depends on the intersections
  - Intersecting, contains the relations α_1, α_2 , α_3, α_4 , α_5 , α_6 , α_7, α_8 , α_9 , α_{10} , α_{11}
  - Non-Intersecting, contains the relations α_1, α_2 , α_{12}, α_{13}


B
A
α1
α2
α3
α4
α5
α6
α7
α8
α9
α10
α11
α12
α13
Non intersecting - Backward
Intersecting - Backward
Intersecting - coTemporal
Intersecting - Forward
Non intersecting - Forward

Problem

- Define our (intuitive) paths in terms of Allen’s algebra
- Define other useful paths based on this algebra

Continuous path (Allen's intervals)

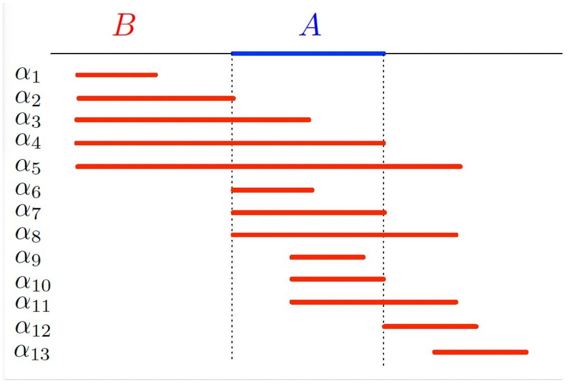
- Given a Temporal Network (N, r), a continuous path (cp) with interval T from node n_1 to node n_k , traversing a relationship r , is a structure ((γ, *I*), T) where:
  - (γ ,*I*) is an $\alpha_{3 \rightarrow 11}$-path.
  - $T = \cap\ T_{i=1,k}\ I_i$
  - T != Ø.

As we can see, in this kind of paths, all relations belong to the *Intersecting* group either Forward, Backward or Co-Temporal.

Continuous path queries

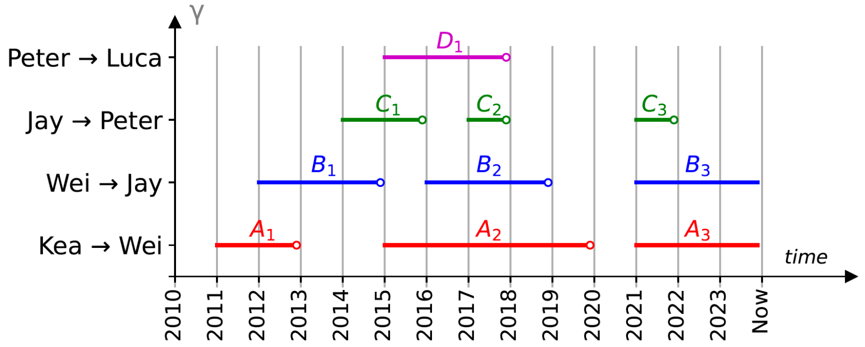
Query 7. *Find the continuous paths between Kea Sanders-Upcott and Luca Mori with a minimum length of 3 and a maximum length of 4*

- γ = (Kea, Wei, Jay, Peter, Luca)
- *I* = (A_2, $B_{2,}$ C_2, D_1)
- *$a_9(A_2, B_2)$, $\alpha_9(B_2, C_2)$, $\alpha_{10}(C_2, D_1)$ => $a_{9,10}$-path*
- T = [2017,2018)



Query 8. *Find the continuous paths between Kea Sanders-Upcott and Peter Norton between* [2021-Now].

- γ = (Kea, Wei, Jay, Peter)
- *I* = (A_3, $B_{3,}$ C_3)
- *I* = $\alpha_7(A_3, B_3)$, $\alpha_6(B_3, C_3)$ => *$a_{6,7}$-path*
- T = [2021, 2022)

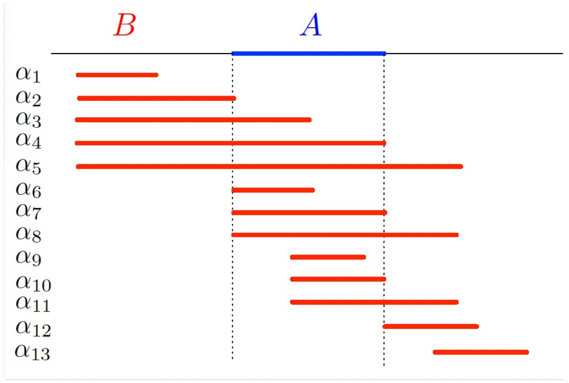


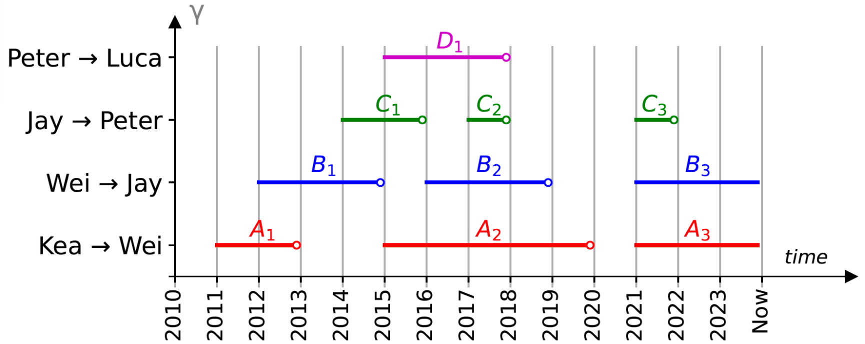
Pairwise continuous path (Allen's intervals)

Given a Temporal Network (N, r), a pairwise continuous path (pcp) traversing a relationship r is an $\alpha_{3 \to 11}$-path

Query 9 *Find the pairwise continuous paths between Kea Sanders-Upcott and Peter Norton, with a minimum length of two and a maximum length of three.*

- γ = (Kea, Wei, Jay, Peter)
- $I = \alpha_{11}(A_1, B_1), \alpha_{11}(B_1, C_1)$ [2011,2013), [2012,2015),[2014, 2016)
- $I = \alpha_9(A_2, B_2), \alpha_9(B_2, C_2)$ [2015,2020), [2016,2019),[2017, 2018)
- $I = \alpha_7(A_3, B_3), \alpha_6(B_3, C_3)$ [2021,Now], [2022,Now],[2021, 2022)

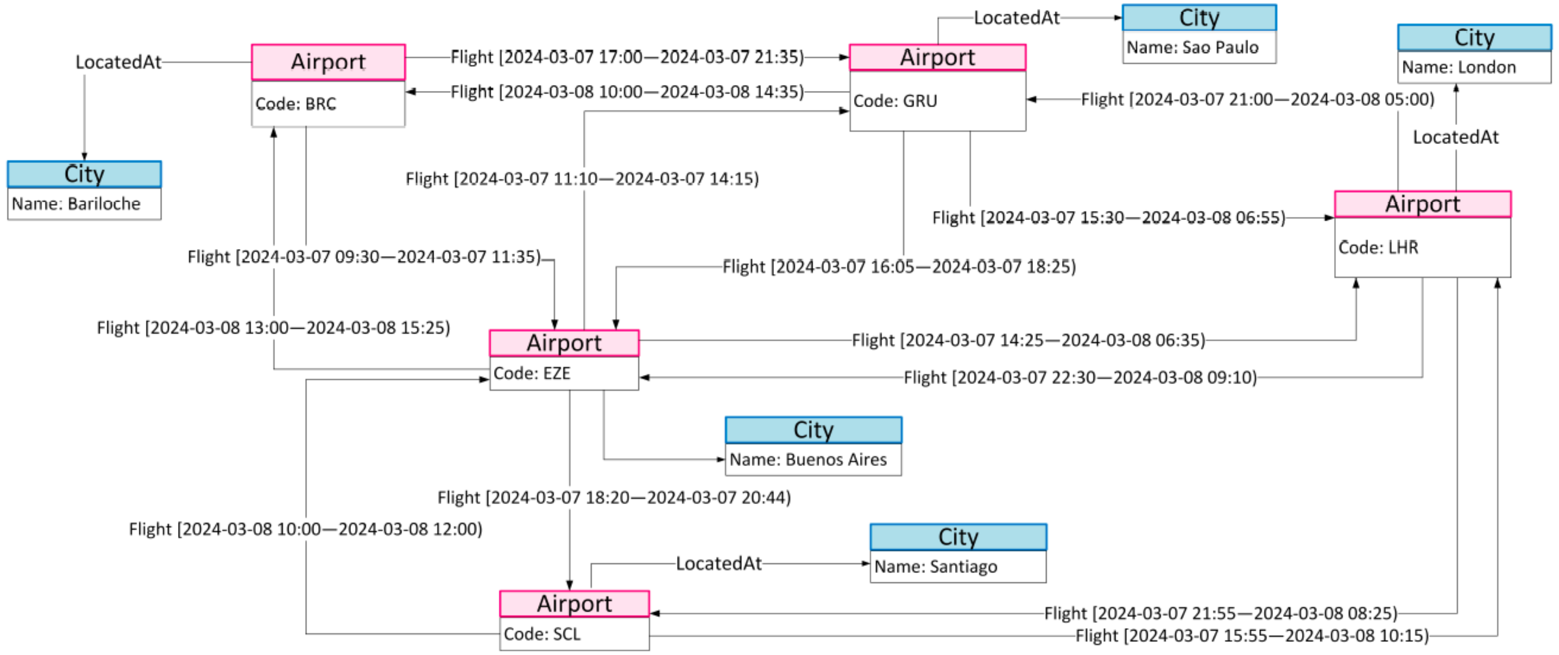




Consecutive path (Allen's intervals)

Given a Temporal Network (N, r), a consecutive path (pcp) traversing a relationship r is an α_{13}-path. Can be forward, backward or co-temporal.

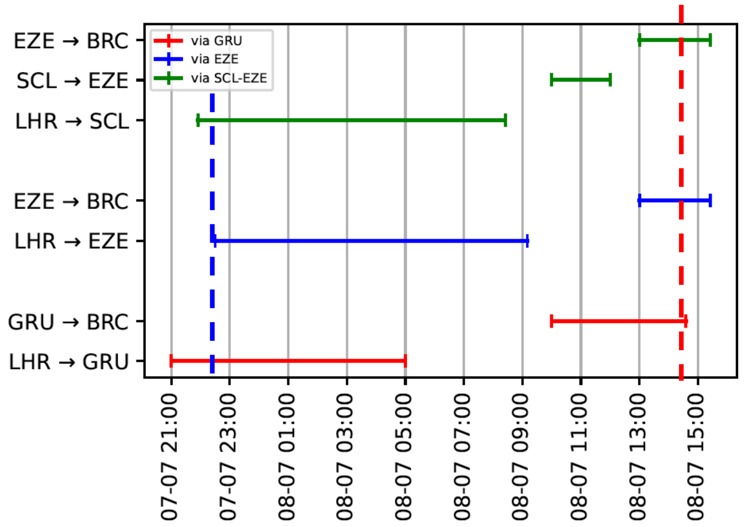
- Used for scheduling
- Example


Airport
Code: BRC
City
Name: Bariloche
Airport
Code: GRU
City
Name: Sao Paulo
City
Name: London
Airport
Code: LHR
Airport
Code: EZE
City
Name: Buenos Aires
City
Name: Santiago
Airport
Code: SCL
LocatedAt
Flight [2024-03-07 17:00—2024-03-07 21:35)
Flight [2024-03-08 10:00—2024-03-08 14:35)
Flight [2024-03-07 21:00—2024-03-08 05:00)
Flight [2024-03-07 11:10—2024-03-07 14:15)
Flight [2024-03-07 15:30—2024-03-08 06:55)
Flight [2024-03-07 09:30—2024-03-07 11:35)
Flight [2024-03-07 16:05—2024-03-07 18:25)
Flight [2024-03-08 13:00—2024-03-08 15:25)
Flight [2024-03-07 14:25—2024-03-08 06:35)
Flight [2024-03-07 22:30—2024-03-08 09:10)
Flight [2024-03-07 18:20—2024-03-07 20:44)
Flight [2024-03-08 10:00—2024-03-08 12:00)
Flight [2024-03-07 21:55—2024-03-08 08:25)
Flight [2024-03-07 15:55—2024-03-08 10:15)

Consecutive path (Allen's intervals)

Query 10 *How can I go from London (LHR) to Bariloche (BRC) arriving as early as possible? => earliest arrival path query , => (LHR, GRU, BRC)*
Query 11 *How can I arrive in BRC departing from LHR as late as possible and arriving before July 8th at 8 pm? => latest departure path query (LHR, EZE, BRC)*

The two options are consecutive paths (α_{13}-paths)



Agenda

- **Temporal Graph Databases**
    - Temporal graph data models
    - An abstract data model
    - The T-GQL query language

T-GQL: A query language for temporal graphs

- Defined over the abstract model
- Based on Cypher, Neo4j’s high-level query language
- Cypher's formal semantics can be found in *
- We assume functions compute paths (cPath, etc).
- Typical SELECT-MATCH-WHERE form
- SELECT projects variables defined in the MATCH clause (aliases allowed)
- MATCH clause may contain one or more patterns and function calls
- A T-GQL query returns a temporal graph. Can be modified by the SNAPSHOT operator, which returns a non-temporal graph

* Francis et al. Cypher: An Evolving Query Language for Property Graphs. SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018.

T-GQL by Example

- *“Name of the friends of the friends of Wei Xu”.*

```
SELECT p2.Name AS friend_name
MATCH (p1:Person)-[:Friend*2]->(p2:Person)
WHERE p1.Name = ‘Wei Xu’
```

To list the three paths in the answer we write:

```
SELECT *
MATCH (p1:Person)-[:Friend*2]->(p2:Person)
WHERE p1.Name = ‘Wei Xu’
```

T-GQL by Example – Temporal operators

- SNAPSHOT: Returns the state of the graph at a certain point in time. Yields a non-temporal graph

- *“Who were friends of Wei Xu in 2018?”*

```
SELECT p2.Name AS friend_name
MATCH (p1:Person) - [:Friend] -> (p2:Person)
WHERE p1.Name = 'Wei Xu'
SNAPSHOT '2018'
```

T-GQL by Example – Temporal operators

- BETWEEN: Computes the intersection of the graph intervals with a given interval Exactly one interval is allowed. The granularity of both intervals must be the same

- *“Where did people who where friends of Kea between 2011 and 2013 live at that time?”*

```
SELECT c.Name
MATCH (p1:Person) - [:Friend] -> (p2:Person),
      (p2) - [:LivedIn] -> (c:City)
WHERE p1.Name = ‘Kea Sanders-Upcott’
BETWEEN ’2011’ and ’2013’
```

The only *Friend* relationship valid during the interval was Wei Xu, so the query returns *Mumbai*, the city where Wei lived since 2001.

T-GQL by Example – Temporal operators

- WHEN: Expresses parallel-period queries. Function calls not allowed in inner queries.

- *“Who were friends of Kea while she was living in Buenos Aires?”*

```
SELECT p2.Name AS friend_name
MATCH (p1:Person) - [:Friend] -> (p2:Person)
WHERE p1.Name = ‘Kea Sanders-Upcott’
WHEN
  MATCH (p1) - [e:LivedIn] -> (c:City)
  WHERE c.Name = ’Buenos Aires’
```

Kea lived in Buenos Aires between [2016-Now], thus, any person who was a friend of Kea at any instant of that interval would be in the result (in this case, Wei and Peter).

T-GQL by Example – Temporal operators

- WHEN: Expresses parallel-period queries. Function calls not allowed in inner queries.

- *"Where did Jay live when he and Sandra followed the same brands?"*

```
SELECT c.Name as city, b1.Name as brand
MATCH (p1:Person) - [:LivedIn] -> (c:City),
      (p1) - [:Fan] -> (b1:Brand)
WHERE p1.Name = 'Jay Bedi'
WHEN
   MATCH (p2:Person) - [f:Fan] -> (b2:Brand)
   WHERE p2.Name = 'Sandra Perez' AND b1.Name = b2.Name
```

Jay and Sandra followed LG together between 2020 and 2023. In this interval, Jay lived in Mumbai and Sandra in Paris. So, (Mumbai, LG) is the answer to the query.

T-GQL: Continuous Path queries

- *"Compute the friends of the friends of each person, and the period such that the relationship holds continuously through all the path."*

```
SELECT path
MATCH (n:Person),
      path = cPath((n)-[:Friend*2] -> (:Person))
```

The modifiers SKIP and LIMIT can be used, as in Cypher, to get a specific path or a range.

```
SELECT path
MATCH (n:Person),
      path = cPath((n)-[:Friend*2] -> (:Person))
SKIP 2
LIMIT 1
```

T-GQL: Continuous path queries

- *Find the continuous paths between Kea Sanders-Upcott and Luca Mori with a minimum length of three and a maximum length of four.*

```
SELECT paths
MATCH (p1:Person), (p2:Person),
       paths = cPath((p1) - [:Friend*3..4] -> (p2))
WHERE p1.Name = ‘Kea Sanders-Upcott’ AND p2.Name = ‘Luca Mori’
```

- The cPath function computes the continuous path

T-GQL: Continuous path queries

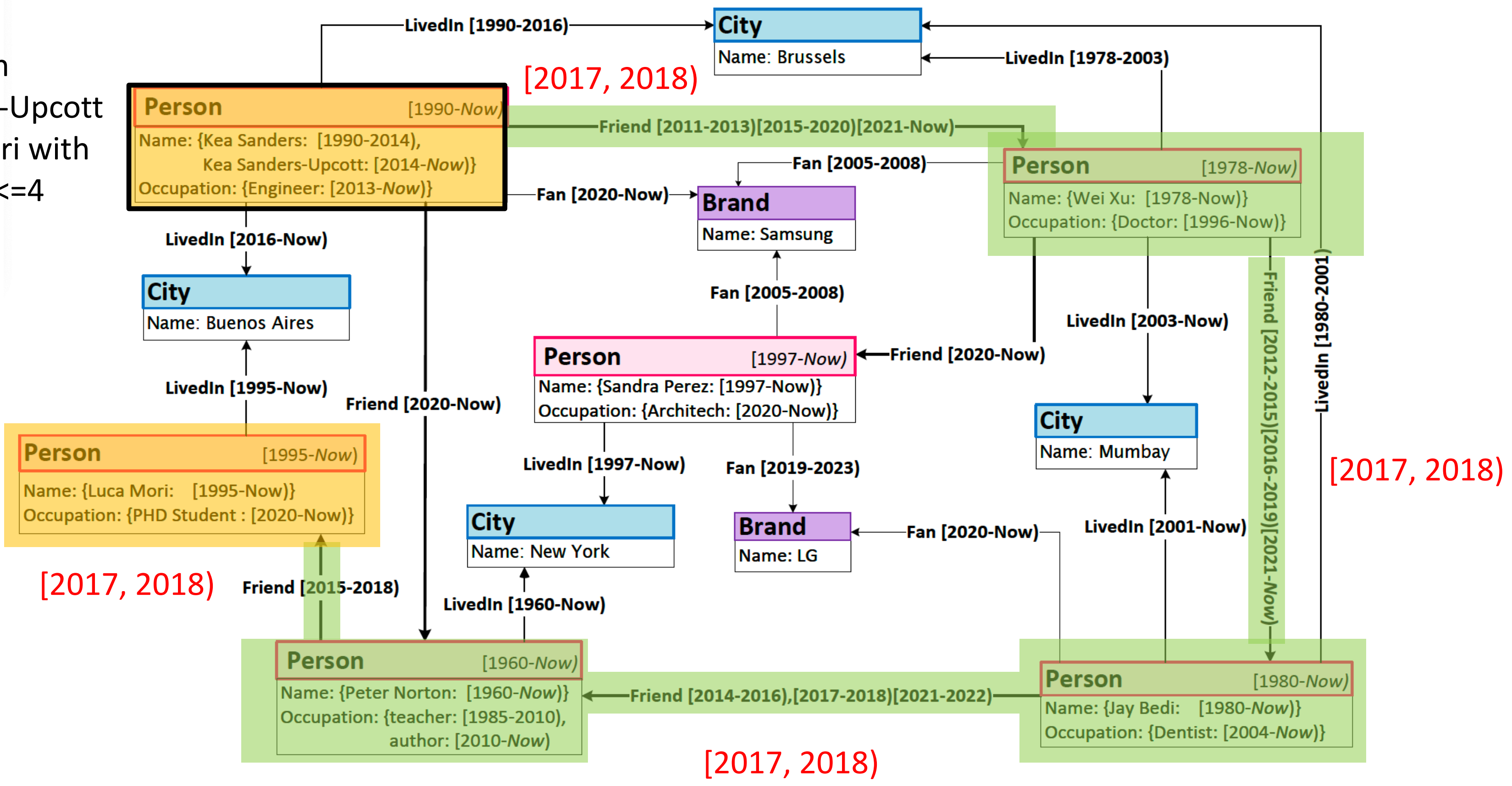
- *Find the continuous paths between Kea Sanders-Upcott and Luca Mori with a minimum length of three and a maximum length of four in the interval [2017,2018).*

```
SELECT paths
MATCH (p1:Person), (p2:Person),
  paths = cPath((p1) - [:Friend*3..4] -> (p2), ‘2017', ‘2018')
WHERE p1.Name = ‘Kea Sanders-Upcott’ AND p2.Name = ‘Luca Mori’
```

- Intermediate results of a query can be filtered by a user-provided interval W that filters out the paths whose interval does not intersect with W
- The result is a single path of length four (the other possible path, with length one, is discarded), with interval [2017, 2018).

T-GQL: Continuous path queries

CPs between
Kea Sanders-Upcott
and Luca Mori with
3 <= length <=4


LivedIn [1990-2016)
City
Name: Brussels
LivedIn [1978-2003)
[2017, 2018)
Person [1990-Now)
Name: {Kea Sanders: [1990-2014), Kea Sanders-Upcott: [2014-Now)}
Occupation: {Engineer: [2013-Now)}
Friend [2011-2013)[2015-2020)[2021-Now)
Fan [2005-2008)
Person [1978-Now)
Name: {Wei Xu: [1978-Now)}
Occupation: {Doctor: [1996-Now)}
Fan [2020-Now)
Brand
Name: Samsung
LivedIn [2016-Now)
City
Name: Buenos Aires
Fan [2005-2008)
LivedIn [2003-Now)
Person [1997-Now)
Name: {Sandra Perez: [1997-Now)}
Occupation: {Architech: [2020-Now)}
Friend [2020-Now)
LivedIn [1995-Now)
Friend [2020-Now)
City
Name: Mumbay
Friend [2012-2015)[2016-2019)[2021-Now)
LivedIn [1980-2001)
Person [1995-Now)
Name: {Luca Mori: [1995-Now)}
Occupation: {PHD Student : [2020-Now)}
LivedIn [1997-Now)
Fan [2019-2023)
[2017, 2018)
City
Name: New York
Brand
Name: LG
Fan [2020-Now)
LivedIn [2001-Now)
[2017, 2018)
Friend [2015-2018)
LivedIn [1960-Now)
Person [1960-Now)
Name: {Peter Norton: [1960-Now)}
Occupation: {teacher: [1985-2010), author: [2010-Now)}
Friend [2014-2016),[2017-2018)[2021-2022)
Person [1980-Now)
Name: {Jay Bedi: [1980-Now)}
Occupation: {Dentist: [2004-Now)}
[2017, 2018)

T-GQL: Continuous path queries

- Find the names of the persons such that there is a continuous path of length 2 or 3 from them to Peter Norton.

```
SELECT p1.Name
MATCH (p1:Person), (p2:Person)
WHERE p2.Name = ’Peter Norton’ AND
      cPath((p1) - [:Friend*2..3] -> (p2))
```

- The cPath function is **overloaded to return a Boolean value**
- In this case the function call is in the WHERE clause (NOT in the SELECT)

T-GQL: Pairwise Continuous path queries

- *Find the pairwise continuous paths between Kea Sanders-Upcott and Peter Norton with a minimum length of two and a maximum length of three.*

```
SELECT paths
MATCH (p1:Person), (p2:Person),
       paths = pairCPath((p1) - [:Friend*2..3] -> (p2))
WHERE p1.Name = ‘Kea Sanders-Upcott’
      AND p2.Name = ‘Peter Norton’
```

T-GQL: α-path queries

- *Compute all the α-paths between Kea and Luca during the interval [2015-2021)*

```
SELECT paths
MATCH (p1:Person), (p2:Person),
  paths = alphaPath((s1)-[:Friend*4]-> (s2),‘2015’,’2021’)
WHERE p1.Name = ’Kea Sanders-Upcott’ AND
      p2.Name = ’Luca Mori’
```

There are two α-paths in the result in this case.

T-GQL: Consecutive path queries

- *How can I go from London to Bariloche arriving as early as possible?*

```
SELECT path
MATCH (c1:City)-[:LocatedAt]->(a1:Airport),
      (c2:City)-[:LocatedAt]->(a2:Airport),
  path = fastestPath((a1)-[:Flight*]->(a2))
WHERE c1.Name = ‘London' AND c2.Name = ‘Bariloche’
```

- *Show the way to arrive in Bariloche departing from London as late as possible and arriving before July 8th at 8 pm.*

```
SELECT path
MATCH (c1:City)-[:LocatedAt]->(a1:Airport),
      (c2:City)-[:LocatedAt]->(a2:Airport),
  path = latestDeparturePath((a1)-[:Flight*]-> (a2),'2019-07-15
       20:20')
WHERE c1.Name= ‘London' AND c2.Name = ‘Bariloche’
```

Agenda

- Introduction and motivation
- Temporal Graph Databases
- **Implementation**
- Temporal Graphs in Sensor Networks
- Conclusion

Agenda

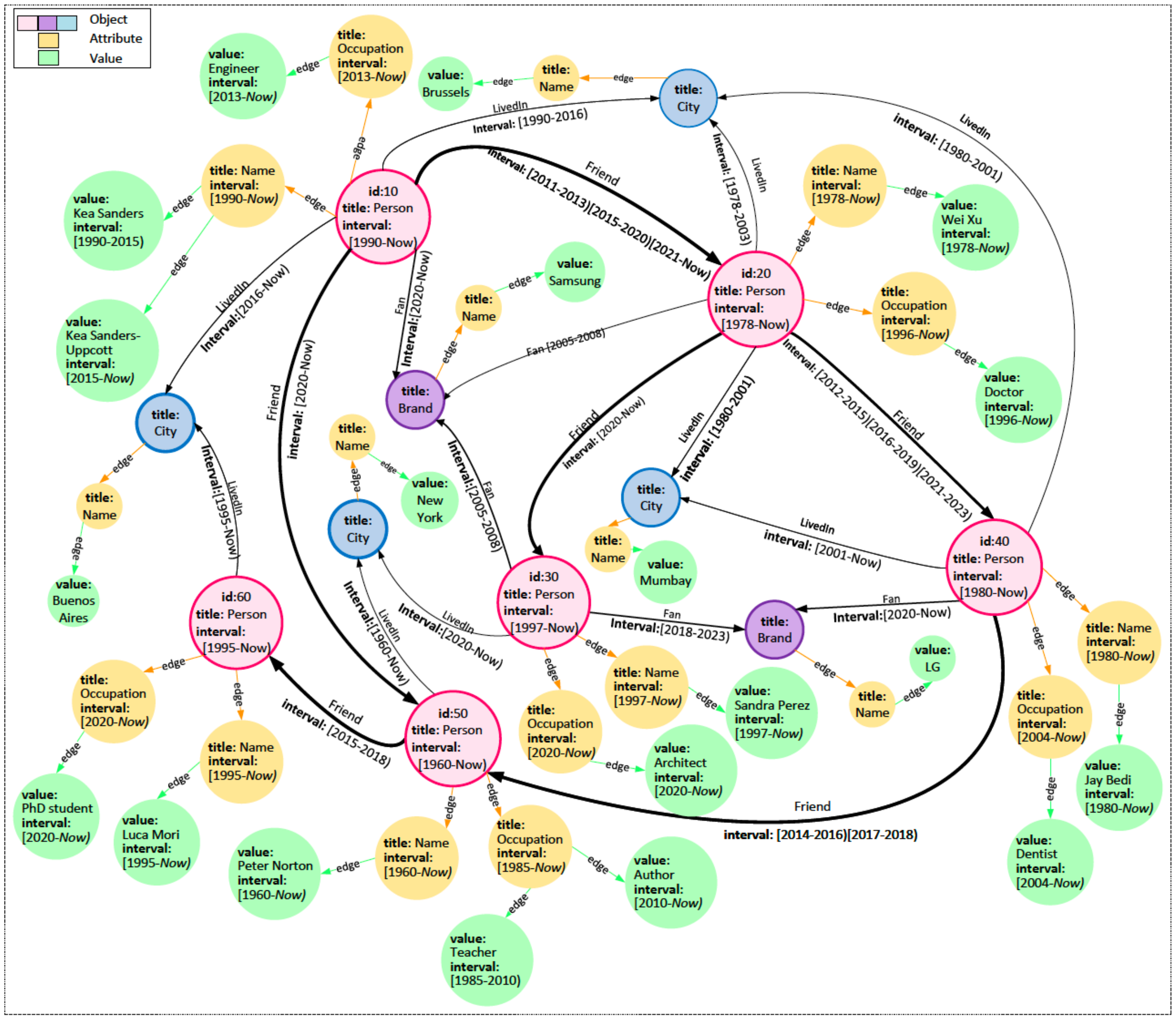
- **Implementation**
  - TGraph: a logical model
  - Implementing TGraph

Agenda

- **Implementation**
  - TGraph: a logical model
  - Implementing TGraph

A logical model for temporal graphs

- We now implement the abstract model
- The logical model is called TGraph, derived from the abstract model
- Implementation based over the property graph data model, in particular, Neo4j
- Abstract model: easy to understand, helps the user to think about the queries, without caring about implementation details
  - Must be translated into the TGraph logical model


Object
Attribute
Value
id:10 title: Person interval: [1990-Now)
id:20 title: Person interval: [1978-Now)
id:30 title: Person interval: [1997-Now)
id:40 title: Person interval: [1980-Now)
id:50 title: Person interval: [1960-Now)
id:60 title: Person interval: [1995-Now)
title: City
title: Brand
title: Name
value: Brussels
value: Samsung
value: New York
value: Mumbay
value: LG
value: Buenos Aires
title: Occupation interval: [2013-Now)
value: Engineer interval: [2013-Now)
title: Name interval: [1990-Now)
value: Kea Sanders interval: [1990-2015)
value: Kea Sanders-Uppcott interval: [2015-Now)
title: Name interval: [1978-Now)
value: Wei Xu interval: [1978-Now)
title: Occupation interval: [1996-Now)
value: Doctor interval: [1996-Now)
title: Name interval: [1997-Now)
value: Sandra Perez interval: [1997-Now)
title: Occupation interval: [2020-Now)
value: Architect interval: [2020-Now)
title: Name interval: [1980-Now)
value: Jay Bedi interval: [1980-Now)
title: Occupation interval: [2004-Now)
value: Dentist interval: [2004-Now)
title: Name interval: [1960-Now)
value: Peter Norton interval: [1960-Now)
title: Occupation interval: [1985-Now)
value: Teacher interval: [1985-2010)
value: Author interval: [2010-Now)
title: Occupation interval: [2020-Now)
value: PhD student interval: [2020-Now)
title: Name interval: [1995-Now)
value: Luca Mori interval: [1995-Now)
edge
LivedIn Interval: [1990-2016)
LivedIn interval: [1980-2001)
LivedIn Interval: [1978-2003)
Friend interval: [2011-2013)[2015-2020)[2021-Now)
LivedIn Interval:[2016-Now)
Fan Interval:[2020-Now)
Fan [2005-2008)
Friend interval: [2020-Now)
Friend interval: [2020-Now)
LivedIn interval: [1980-2001)
Friend interval: [2012-2015)[2016-2019)[2021-2023)
LivedIn Interval:[1995-Now)
Fan Interval:[2005-2008)
LivedIn interval: [2001-Now)
LivedIn Interval:[1960-Now)
LivedIn Interval:[2020-Now)
Fan Interval:[2018-2023)
Fan Interval:[2020-Now)
Friend interval: [2015-2018)
Friend interval: [2014-2016)[2017-2018)

TGraph: a temporal property graph model *

- A structure G(No, Na, Nv, E) where

G: name of the graph, E a set of edges
No: set of object nodes
Na: set of attribute nodes
Nv: set of Value nodes

- Object nodes and attribute nodes are associated with a tuple (*title, interval*)
  - *title* represents the content of the node
  - *interval* are the time periods when the node is (or was) valid
- Similarly, value nodes are associated with a tuple (*value, interval*)
- A special value **Now** tells that the node is valid at the current time
- All nodes also have a (non-temporal) identifier denoted id
- + temporal constraints

* (Debrouvier et al, VLDB J. 30(5): 825-858 (2021))

TGraph: a temporal property graph model

- Constraints

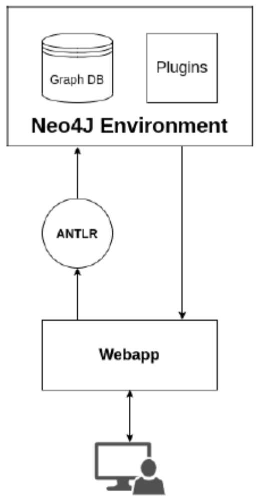
- All nodes in the graph have a different id
- All (value) nodes with the same value associated with the same attribute node are coalesced (*interval* is a set of intervals)
- All edges with the same name between the same pair of nodes, are coalesced
- An Object node can only be connected to an attribute node or to another object node
- An Attribute node can only be connected to a non-attribute node
- A Value node can only be connected to attribute nodes
- Attribute nodes must be *connected by only one edge* to an object node
- Value nodes must only be *connected to one attribute node with one* edge.

Agenda

- **Implementation**
  - TGraph: a logical model
  - Implementing TGraph

T-GQL over the logical model

- T-GQL: a high-level query language for graph databases
- T-GQL implementation extends Cypher with a collection of functions, implemented as in the APOC library, added as a .jar file
- The T-GQL language grammar was implemented using ANTLR
- T-GQL queries translated into Cypher and executed on a Neo4j server that contains the plugins to run the temporal operators and path algorithms



T-GQL translation

```
SELECT p
MATCH (p:Person)
WHERE p.Name = 'John' and p.Age = 18
```

- Object nodes in the MATCH clause translated as {alias:Object {title: ‘Name’}}
- E.g., “(p:Person)” would be translated to {p:Object {title: ‘Person’}}.
- Edges not translated, match the Cypher’s grammar
- For each attribute in the SELECT clause, a three-node path (Object - Attribute - Value) is produced from the object node
- Variables starting with ‘internal’ (generated by the parser) are reserved.
- The condition `p.Name = 'John' and p.Age = 18` translated as:

```
MATCH (p)-->(internal_n:Attribute{title:'Name'})-->(internal_v:value)
MATCH (p)-->(internal_a:Attribute{title:'Age'})-->(internal_v1:value)
WHERE internal_v.value = 'John' and internal_v1.value = 18
```

T-GQL: Temporal granularity

- Problem extensively studied in temporal database theory
- Queries may mention a granularity *qg* different than the graph's objects *og*
- When this is the situation, two cases may occur:
    - *qg* is finer than *og* => the coarser granularity transformed into the finer one. E.g., if *og.interval* = [2010, 2012), and the condition is *t* IN *og.interval* where *t* = 2/10/2012, then, the interval is transformed into *og.interval* = [1/1/2010, 1/1/2013) => answer = true
    - *qg* is coarser than *og* => one time instant in the granularity of *og* is chosen (the finer in transformed into the coarser). E.g., if *og.interval* = [15/10/2010, 23/12/2010), and the condition is 2010 IN *og*.interval, the the object's interval becomes og= [2010, 2011) => answer = true

T-GQL: Temporal granularity

- In our example, if a query asks for Jay's friends on October 10th, 2018, we cannot give a precise answer, and the query must use the semantics explained

- T-GQL supports the following granularities

  Year: yyyy
  YearMonth: yyyy-MM
  Date: yyyy-MM-dd
  Datetime: yyyy-MM-dd HH:mm

T-GQL: Path queries

- The T-GQL language supports the path semantics studied above (we extend this later)
  - Continuous path semantics
  - Pairwise Continuous path semantics
  - Consecutive path semantics
  - α –path semantics
- Semantics implemented by means of functions, which are included in a library of Neo4j plugins
- To compute temporal paths, two types of functions are defined
  - Coexisting
  - Consecutive
- Both receive two nodes as arguments

T-GQL translation

```
SELECT p.path as path, p.interval AS interval
MATCH (p1:Person), (p2:Person), p = cPath((p1)-
  [:Friend*2..3]->(p2),'2018','2021')
WHERE p1.Name = 'Kea Sanders-Upcott'
```

- Translation:

```
MATCH(p1:Object{title:'Person'}),(p2:Object{title:'Person'})
MATCH(p1)-->(internal_n0:Attribute{title:'Name'})
                -->(internal_v0:value)
WHERE internal_v0.value = 'Kea Sanders-Upcott'
CALL coexisting.coTemporalPaths(p1,p2,2,3 {edgesLabel:'Friend',
     nodesLabel:'Person', between:'2018-2021',direction:'outgoing'})
     YIELD path as internal_p1, interval as internal_i1
WITH {path:internal_p1,interval:internal_i1} AS p
RETURN p.path AS 'path', p.interval AS 'interval'
```

TGraph implementation

- *Find the continuous paths between Kea Sanders-Upcott and Peter Norton with a minimum length of two and a maximum length of three,* in the interval [2016,2020].

```
SELECT paths
MATCH (p1:Person), (p2:Person),
      paths = cPath((p1) - [:Friend*2..3] -> (p2), '2018', '2020')
WHERE p1.Name = ‘Kea Sanders-Upcott’ AND p2.Name = ‘Peter Norton’
```

T-GQL: Continuous path queries

- *Find the continuous paths between Kea Sanders-Upcott and Peter Norton with a minimum length of two and a maximum length of three.*

```
"path": [
    { "interval": [”1990-Now" ], // interval of the attribute node
      "attributes": {
        "Name": [
          {"value": "Kea Sanders-Upcott",
            "interval": "[2017 - 2018]" }] // value node interval
           },                                returned is the
      "id": 10, // id of Object node.        interval of the CP
      "title": "Person” // Object node       note: value node
    },                                       inlined in the
    {                                        attribute node
      ...
    }
  ],
  "interval": "2017-2018” // interval of the CP
}
```

T-GQL: Continuous path queries

- *Find the continuous paths between Kea Sanders-Upcott and Peter Norton with a minimum length of two and a maximum length of three.*

- The figure shows the format of the result.
- Attribute and value nodes are inlined to facilitate their search
- The value “Kea Sanders” is ignored since its interval [1990-2014] does not intersect with the CP’s interval [2017-2018]
- The value node returned has the interval [2017–2018], i.e., the intersection of the intervals [1990-Now] (the interval of the value node) and [2017-2018]
- Finally, the interval of the CP is [2017-2018], which is the result of the intersection between the traversed edges

Agenda

- Introduction and motivation
- Temporal Graph Databases
- Implementation
- **Temporal Graphs in Sensor Networks**
- Conclusion

Agenda

- **Temporal Graphs in Sensor Networks**
    - Abstract graph model for sensor networks
    - Paths in sensor networks
    - Use case

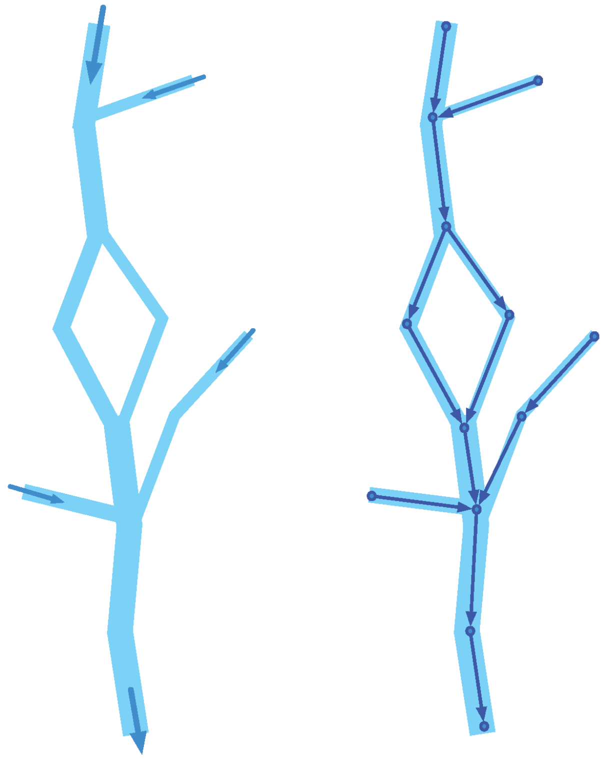
Agenda

- **Temporal Graphs in Sensor Networks**
  - Abstract graph model for sensor networks
  - Paths in sensor networks
  - Use case

Transportation networks

- Transportation networks: physical networks through which objects or substances can move. Examples include:

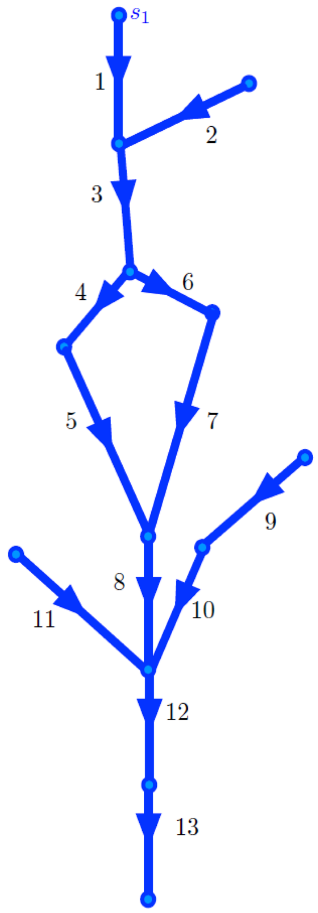
- River networks (water and other substances move)
- Road networks (cars, bicycles and pedestrians move)
- Computer networks (information moves)
- Electricity grids (electricity moves)

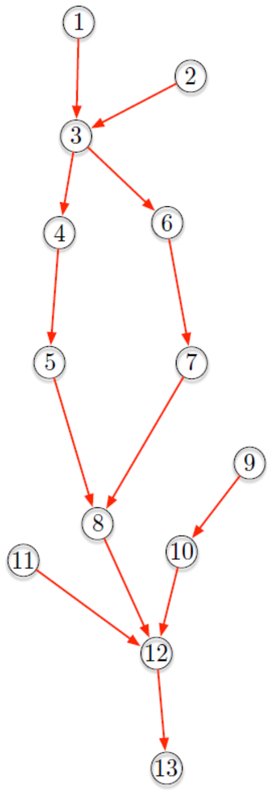


- These physical networks typically embedded in a geographic space
- Left: physical network; Right: graph model

Transportation networks

- Topological data model
- Left: segments as edges
- Right: segments as nodes (flow model*)





* Bollen, E., Hendrix, R., Kuijpers, B., & Vaisman, A. Towards the Internet of Water: Using Graph Databases for Hydrological Analysis on the Flemish River System. Trans. in GIS, 25 (6), 2907-2938, 2021. https://doi.org/10.1111/tgis.12801

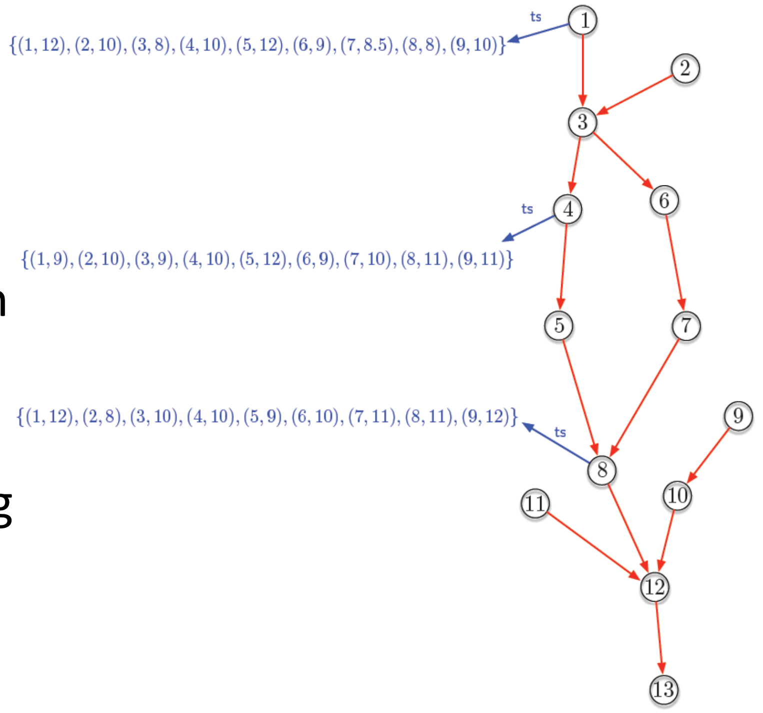
Transportation networks

- A transportation network TN is a directed graph (N, Flow), where N is a finite set of nodes and Flow $\subseteq$ N x N is a set of directed edges, representing the flow of a subject (e.g., water) from one node to another
- Sensor-equipped transportation network when, at certain locations in the TN, there are sensors that measure some parameter
- These measurements come together with a timestamp
- We use a river system example*, sensors measure water height, temperature and salinity of the water (electric conductivity)
- Sensor measurements often taken at regular moments in time and the frequency may vary from once per minute to once per hour, etc.

* Bollen, E., Hendrix, R., Kuijpers, B., & Vaisman, A. A. Time-series-based queries on stable transportation networks equipped with sensors. Int. J. Geo Inf., 10 (8), 531, 2021. https://doi.org/10.3390/ijgi10080531

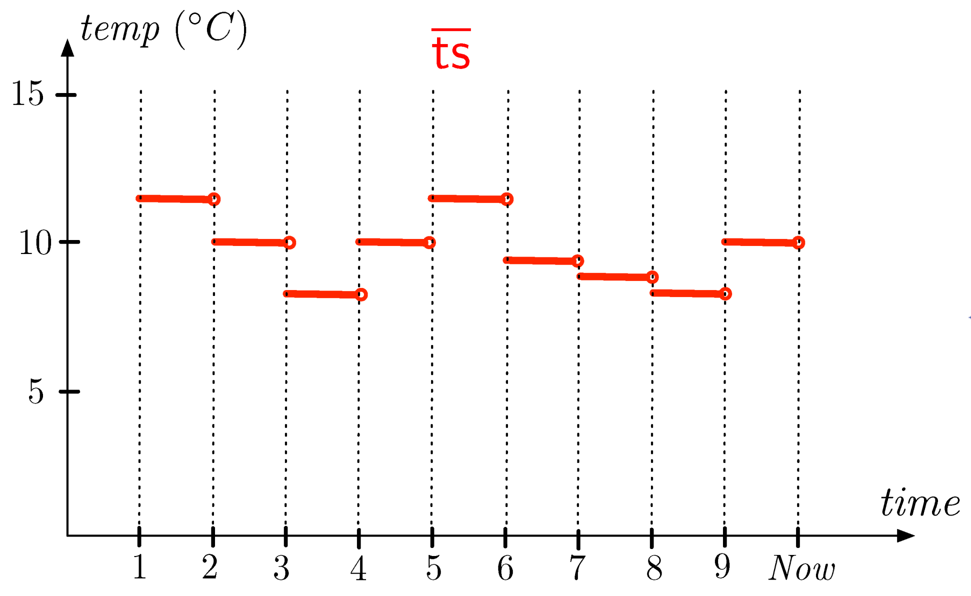
Sensor networks

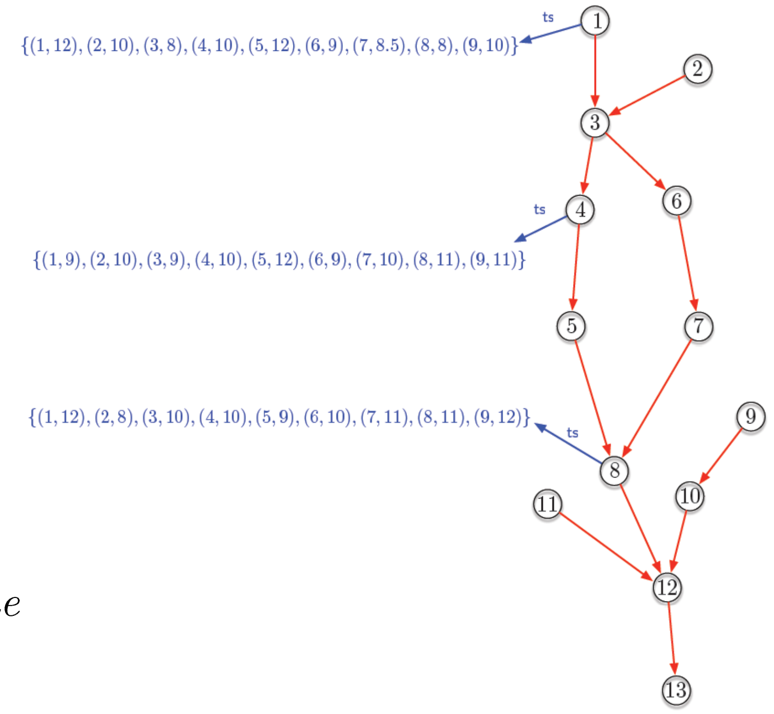
- A sensor-equipped transportation network (or sensor network, for short) SN is a four-tuple (N, Flow, S, ts) s.t:
  - (N, Flow) is a transportation network;
  - S ⊆ N is a set of sensor-equipped nodes (sensors, for short); and
  - ts : S → $2^{\mathbb{T} \times \mathbb{V}}$ is a (time-series) function that maps sensors to finite functions from $\mathbb{T}$ to $\mathbb{V}$
- In general, we consider $\mathbb{T}$ and $\mathbb{V}$ as being the set of the real or rational numbers, but $\mathbb{V}$ can also be natural numbers, e.g., categories



Transportation networks

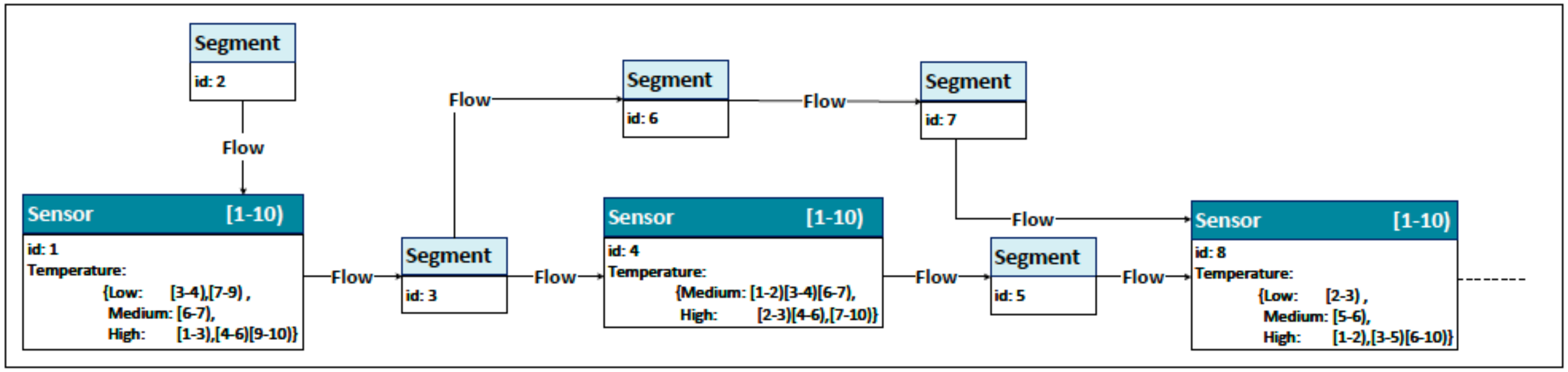
- Between measurements we can use any interpolation function
- Here we assume a step function
- For Sensor 1 we have:





Abstract data model for sensor networks

- An abstract representation of a sensor network
- Sensor nodes: nodes that hold a sensor, different from Segment nodes
- Intervals in sensor nodes: periods when a segment holds a working sensor
- Intervals may indicate the presence of a sensor that no longer works or the removal of a sensor
- Sensor nodes contain time series of the values of categorical variables
- Edges between nodes are labeled Flow
- We denote this as the abstract sensor network model or SN Graph



Agenda

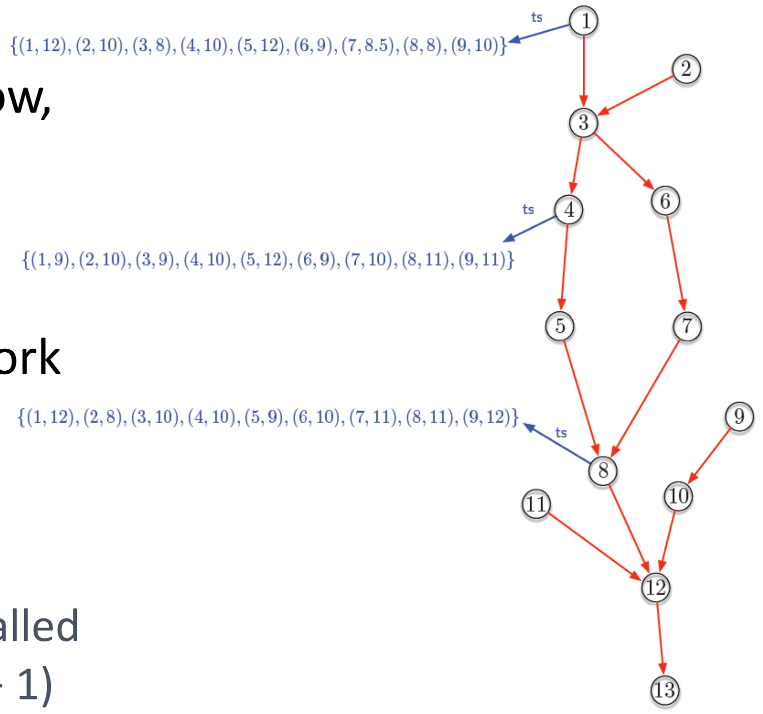
- **Temporal Graphs in Sensor Networks**
  - Abstract graph model for sensor networks
  - Paths in sensor networks
  - Use case

Temporal paths in sensor networks

- Based on the abstract model, we redefine the paths
- We want to address queries like "*Starting from a segment, obtain all the paths and their corresponding time intervals I_i such that the temperature along the path has been simultaneously High for all nodes in the path during I_i*".
- This is a "special" kind of CP
- Now, the notion of CP depends not only on the network topology but also on the values of the variables
- We denote them *SN Continuous paths*
- We study SN paths next and generalize them based on Allen's Algebra

Temporal paths in sensor networks

- A path $\gamma = (n_1, n_2, \ldots n_k)$ in a sensor network SN, is a directed path in (N, Flow, S, ts).
- E.g., $\gamma = (1, 3, 4, 5, 8, 12)$ is a path of length 5.
- A sub-path of a path γ in a sensor network SN consisting of all its sensor nodes is a *full sensor sequence* of γ.
  - Denoted by fss(γ).
  - fss(γ) = $(s_1, s_2, \ldots, s_k)$, s_i and s_{i+1} are called consecutive sensors on γ $(i = 1, \ldots, k - 1)$

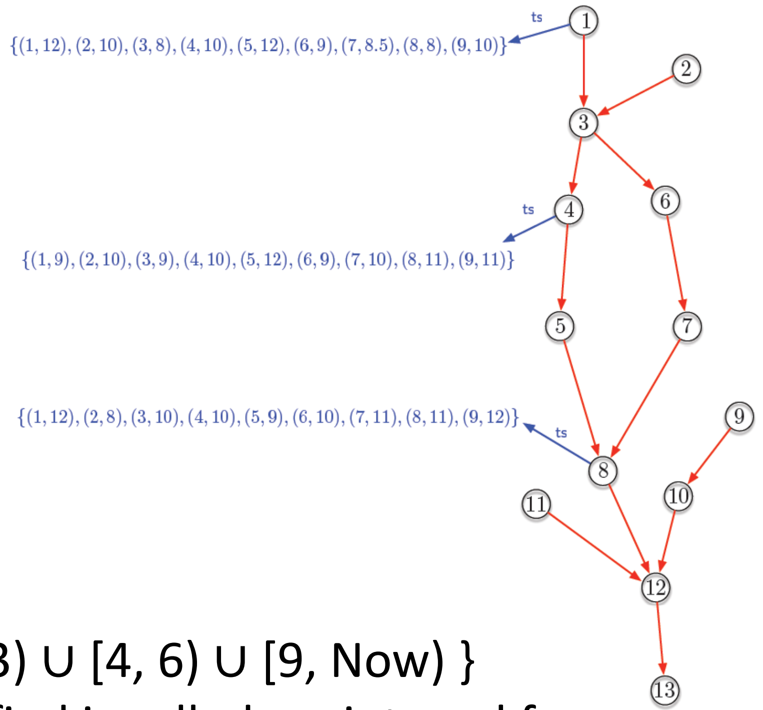


Bollen, E., Hendrix, R., Kuijpers, B., Soliani, V., & Vaisman, A. A. Temporal Paths in Real-World Sensor Networks Int. J. Geo Inf., 13(2) : 36, 2024 . https://doi.org/10.3390/ijgi13020036

Bollen, E., Hendrix, R., Kuijpers, B., Soliani, V., & Vaisman, A. A. Analysing River Systems with Time Series Data using Path Queries in Graph Databases. Int. J. Geo Inf., 12 (3), 94, 2023. https://doi.org/10.3390/ijgi12030094

Temporal paths in sensor networks

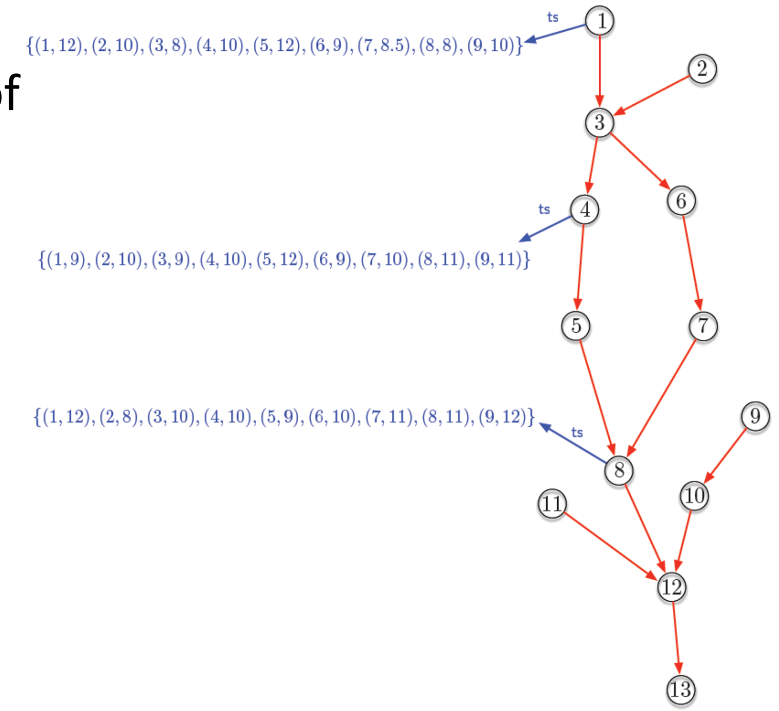
- Let s be a sensor node in N, with time series ts(s), and c a predicate on the set of measurements V
- The set $\mathrm{Val}_c(s)$ consists of all time instants t such that values in the time series of a sensor satisfies the predicate c. Called the *validity time set for condition c at sensor node s*

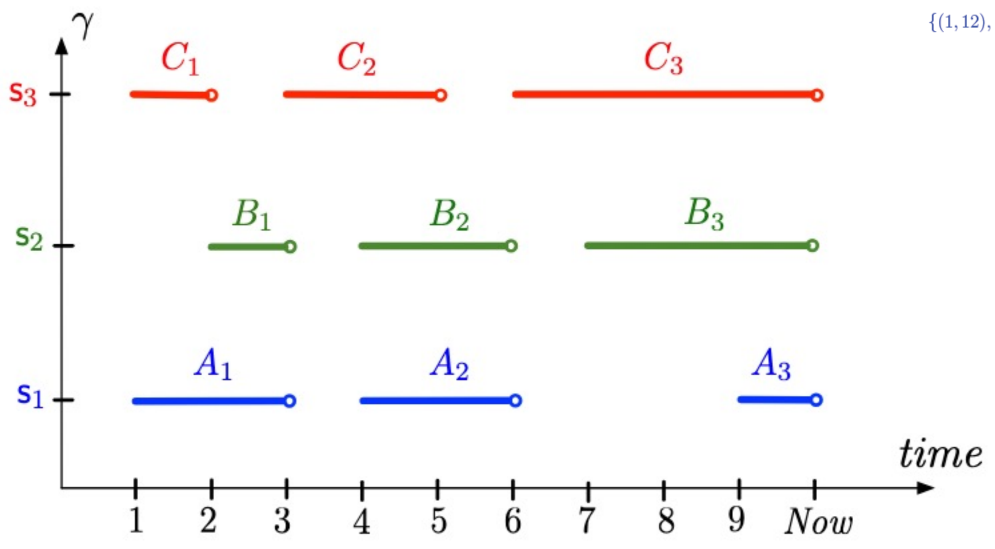


- Example: For *temp* ≥ 10, $\mathrm{Val}_c(s_1)$ = {[1, 3) ∪ [4, 6) ∪ [9, Now) }
- The time interval *I* when $\mathrm{Val}_c(s)$ is satisfied is called a c-interval for s
- If *I* is maximal, it is called a maximal c-interval

Temporal paths in sensor networks

- *c = temp* ≥ 10 and $s_1 = 1$, $s_2 = 4$, $s_3 = 8$
- Vertical axis represents the direction of the flow
- [1, 2): c-interval for s_1
- [4,6): maximal c-interval for s_1



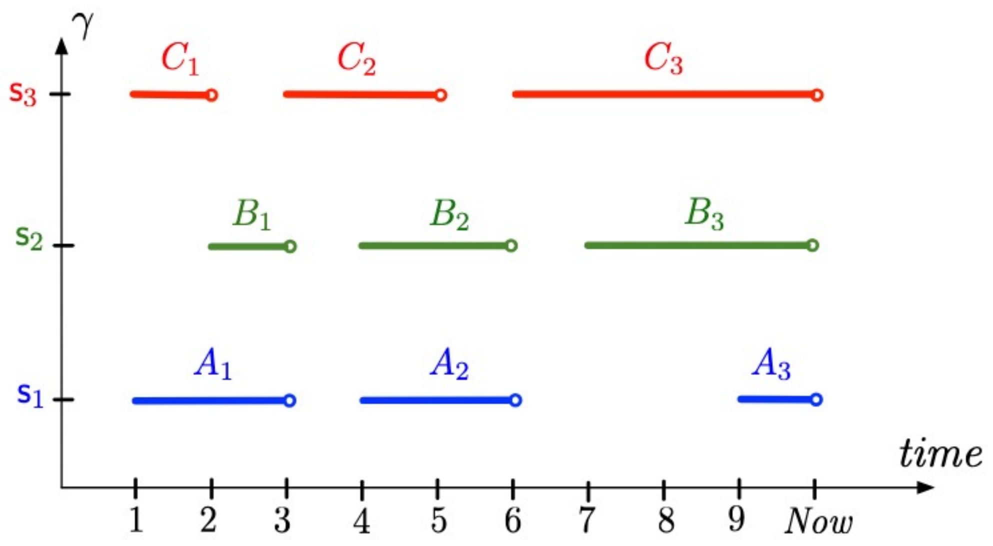


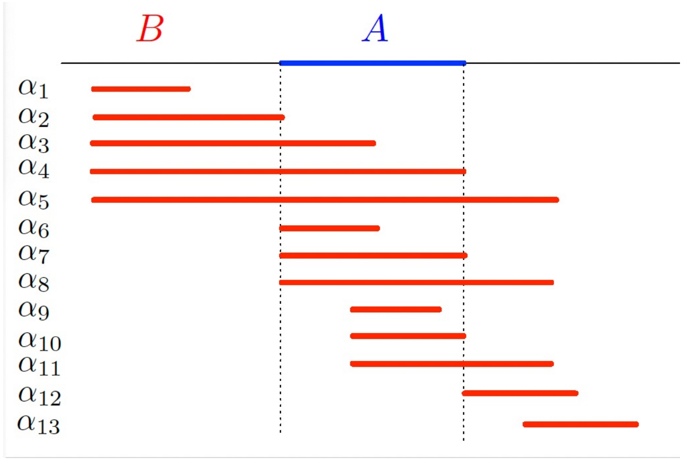
Temporal paths in sensor networks

- Let SN be a sensor network and let c be a condition on sensor values. Let α be a binary relation on temporal intervals. A *temporal α-path* in SN subject to condition c is a structure $(\gamma, ((s_1, I_1), (s_2, I_2), \ldots, (s_k, I_k)))$, where
  - γ is a path in SN;
  - $fss(\gamma) = (s_1, s_2, \ldots, s_k)$;
  - I_j is a maximal c-interval for s, for j = 1, . . . , k; and
  - $\alpha(I_j, I_{j+1})$ holds for j = 1, . . . , k −1
- When I_j is a c-interval (*not maximal*) for s, the path is called a *temporal sub-α-path in SN*

Example

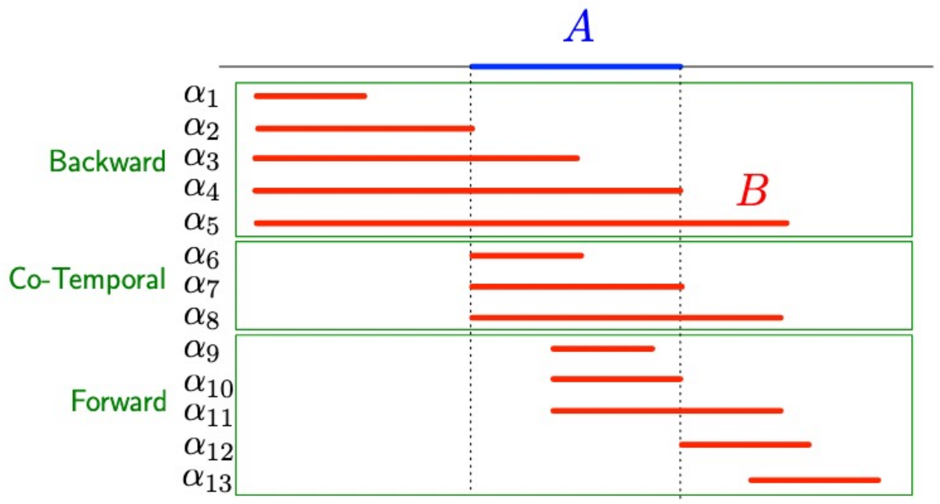
- $\gamma = (1, 3, 4, 5, 8, 12)$
- c = high water temperature
- $(\gamma, ((s_1, A_1), (s_2, B_2), (s_3, C_3)))$ is an $\alpha_{12,13}$-path, since $\alpha_{13}(A_1, B_2)$ and $\alpha_{12}(B_2, C_3)$ hold





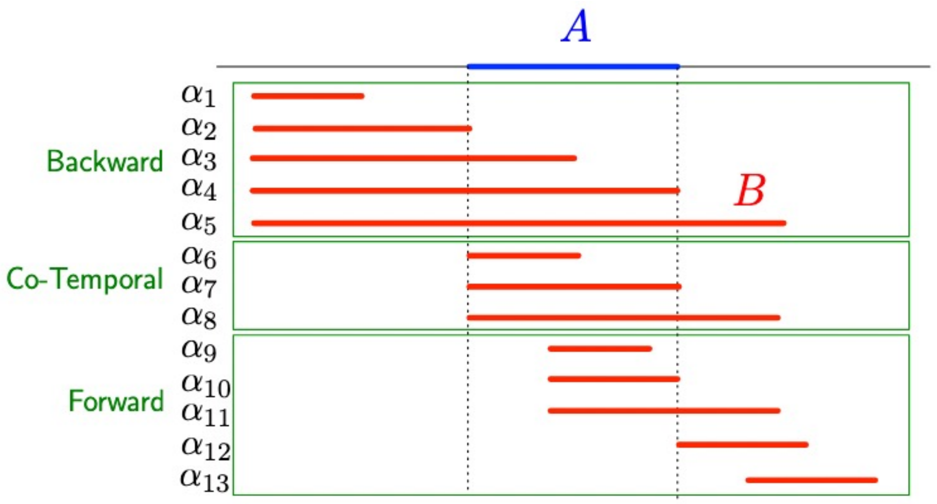
Path properties in sensor networks

- We study three properties of interest
  - Backward, co-temporal and forward relations (already defined)
  - Closure under sensor deletion
  - Temporal and spatial robustness



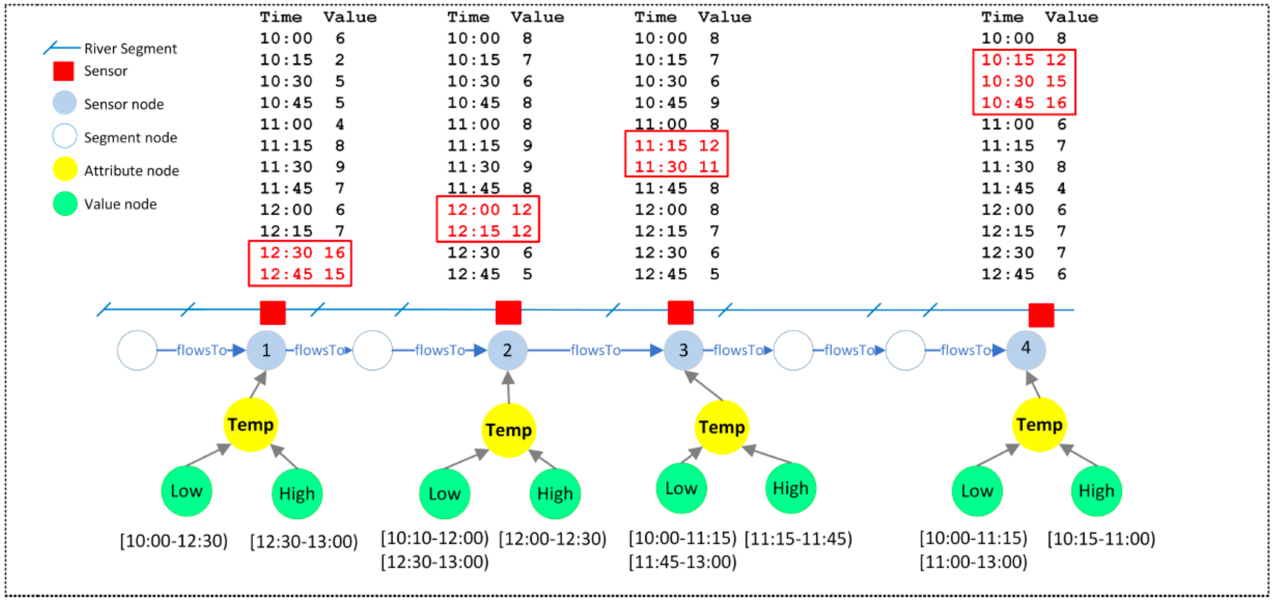
Backward relations

- $\alpha_{1 \rightarrow 5}(I_i, I_{i+1})$ => means that the phenomenon moves backward in the network
- Examples: pollution caused by a boat that travels upstream in a river network, a salmon swimming upstream or a traffic jam on a road network…



Backward relations

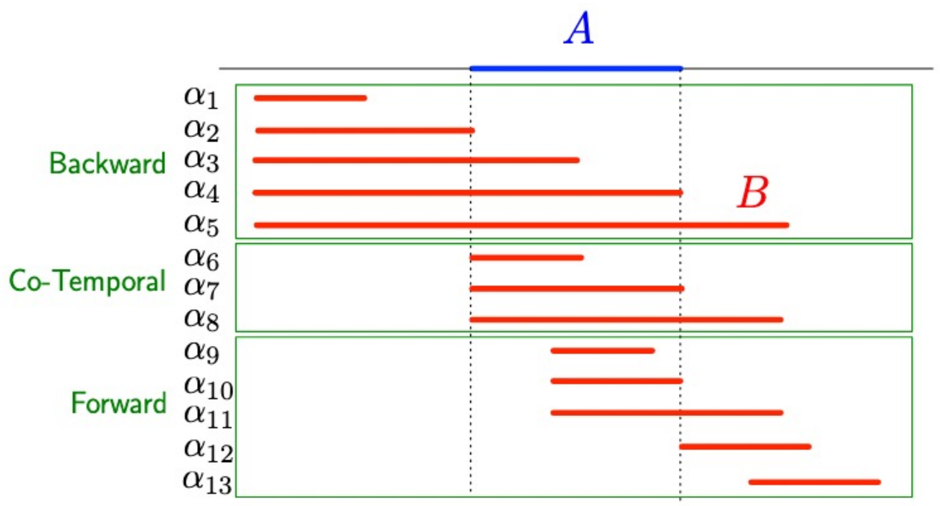
- High values of water temperature are produced earlier in the nodes closer to the sea, that is, the High temperatures arrive later to the sensors that are farther from the sea
- Example: temperature was High Node 4 at 10:15, but this effect only reached Node 1 at 12:30, temperature moves backwards



Time	Value	Time	Value	Time	Value	Time	Value
10:00	6	10:00	8	10:00	8	10:00	8
10:15	2	10:15	7	10:15	7	10:15	12
10:30	5	10:30	6	10:30	6	10:30	15
10:45	5	10:45	8	10:45	9	10:45	16
11:00	4	11:00	8	11:00	8	11:00	6
11:15	8	11:15	9	11:15	12	11:15	7
11:30	9	11:30	9	11:30	11	11:30	8
11:45	7	11:45	8	11:45	8	11:45	4
12:00	6	12:00	12	12:00	8	12:00	6
12:15	7	12:15	12	12:15	7	12:15	7
12:30	16	12:30	6	12:30	6	12:30	7
12:45	15	12:45	5	12:45	5	12:45	6

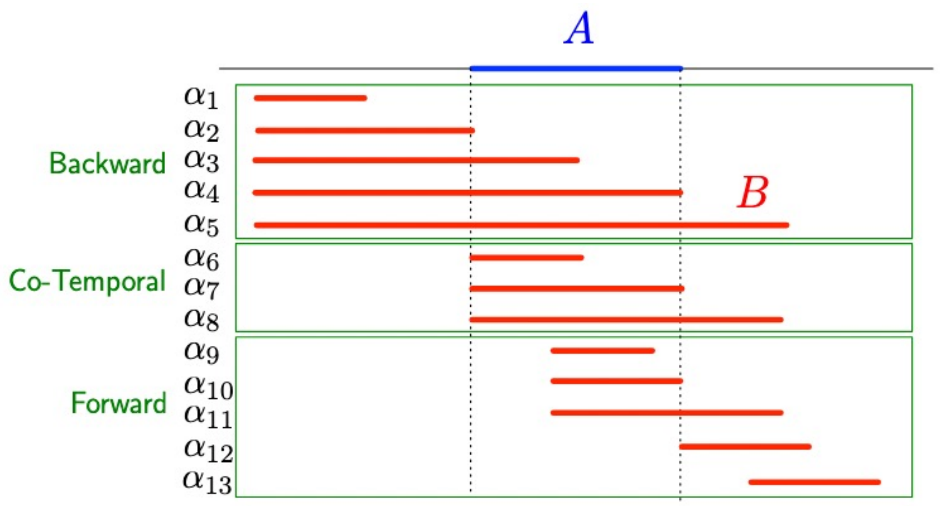
Co-temporal relations

- Both intervals start co-temporally, independent of the flow of the network and without any delay
- Example: causes that are external to the network, like rainfall, which starts at the same time at several locations in the network



Forward relations

- Typical for cases where some phenomenon is propagated through the network, following its natural flow
- Examples: external spills of pollutants in a river system and the density of traffic on a road network



Closure under sensor deletion (CUSD)

- Intuition: when a sensor does not work, or data are lost, the temporal path without this sensor still belongs to the same class of temporal paths
- Why would we want this?

  - When the number of sensors in a network is high, we may want to look for temporal paths belonging to a certain class on a sample of the sensors

  - If a class of temporal paths is CUSD, if we know that a temporal path belongs to that class, if we remove a sensor, the path will still belong to that class

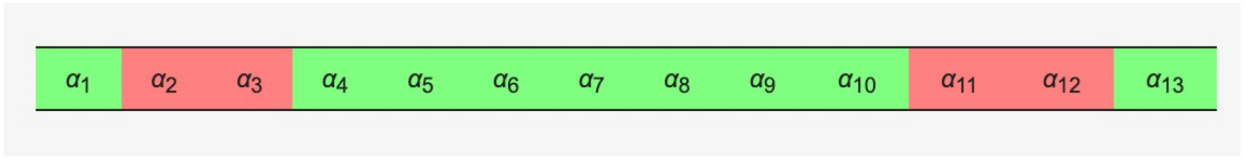
  - If a path of a certain type is CUSD and it is not found on a subset of sensors, we would not find it on the complete set of sensors

Closure under sensor deletion

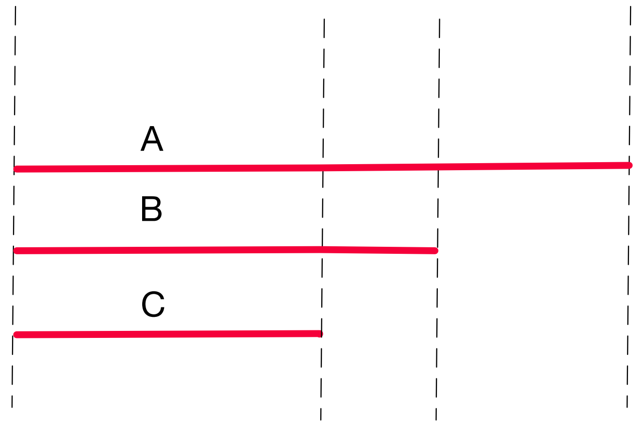
- We have a temporal α-path $(\gamma, ((s_1, I_1), (s_2, I_2), \ldots, (s_k, I_k)))$, where α is some union of basic Allen relations
- Question is: when we remove one of the sensor nodes s_j from the path, is $(\gamma, ((s_1, I_1), (s_2, I_2), \ldots, (s_{j-1}, I_{j-1}), (s_{j+1}, I_{j+1}), \ldots, (s_k, I_k)))$ still an α-path?
- All classes are defined in terms of relationships between intervals of consecutive sensors => it suffices to look at any three successive sensors and their intervals I_{j-1} and I_{j+1}
- In other words: If (I_{j-1}, I_j) and (I_j, I_{j+1}) satisfy relation α, does (I_{j-1}, I_{j+1}) also satisfy α?
- The relation α is ***closed under sensor deletion*** if and only if α is a transitive relation

Closure under sensor deletion

- In green, classes CUSD

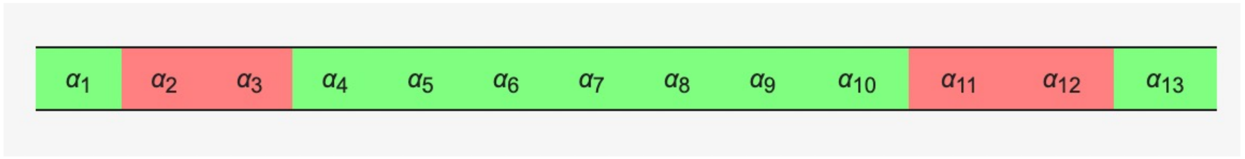


- α_6 CUSD because, if $\alpha_6(A, B)$ and $\alpha_6(B, C)$

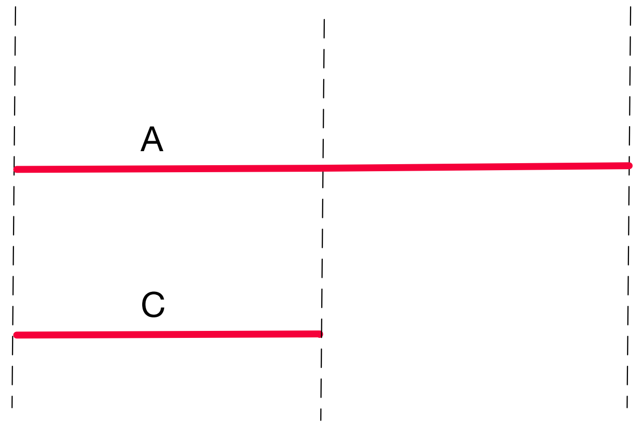


Closure under sensor deletion

- In green, classes CUSD

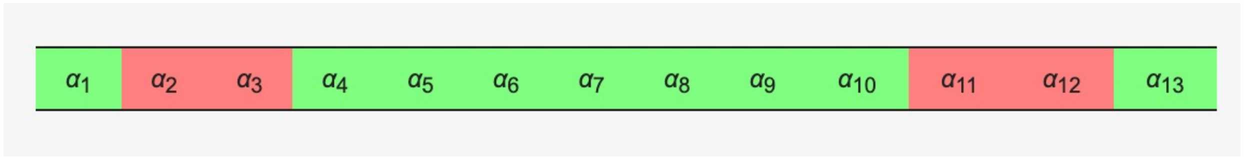


- α_6 CUSD because, if α_6(A, B) and α_6(B, C), if we delete B, we still are in the α_6(A, C) class

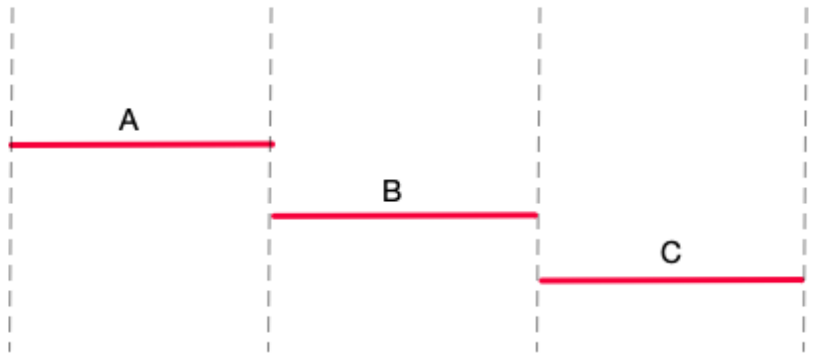


Closure under sensor deletion

- In green, classes CUSD

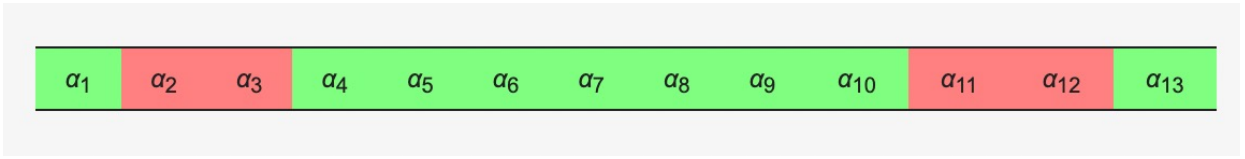


- α_{12} not CUSD because, if $\alpha_{12}(A, B)$ and $\alpha_{12}(B, C)$

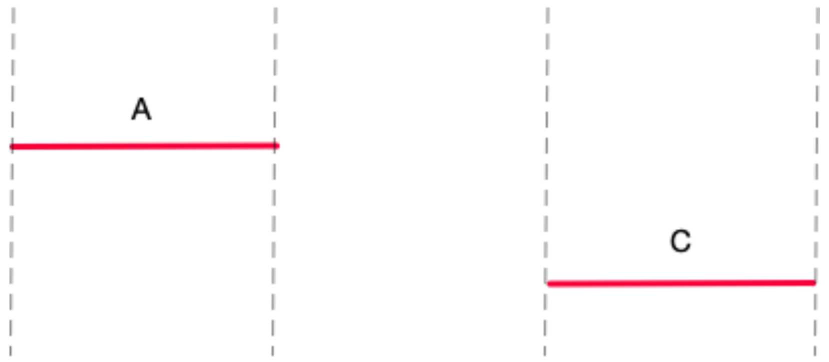


Closure under sensor deletion

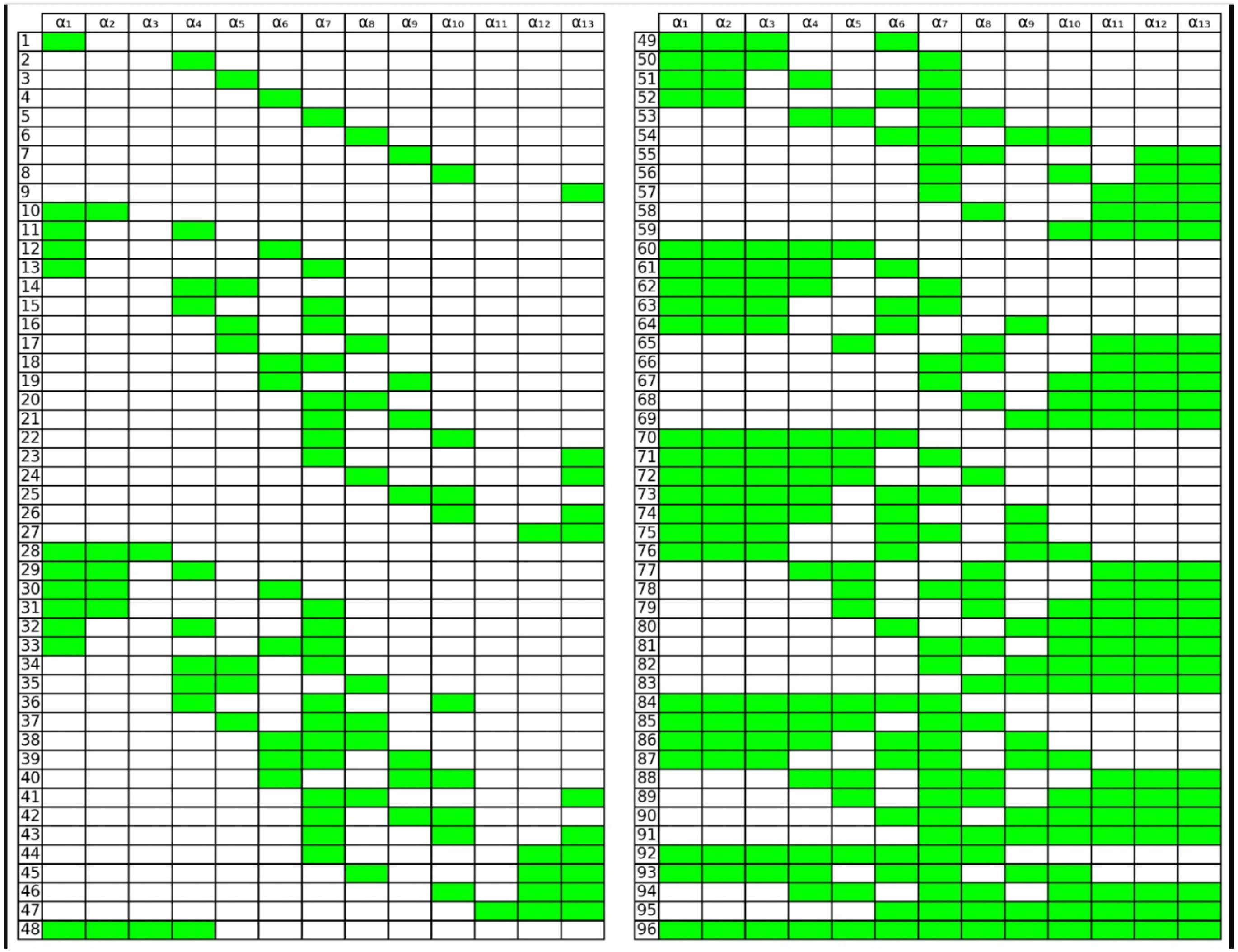
- In green, classes CUSD



- α_{12} not CUSD because, if $\alpha_{12}(A, B)$ and $\alpha_{12}(B, C)$, if we delete B, a gap between A and C is created => we are in the $\alpha_{13}(A, C)$ class
- In fact, α_{13} is CUSD, as well as $\alpha_{12} \cup \alpha_{13}$



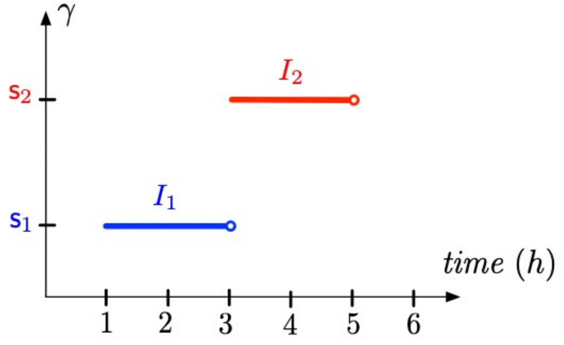
Combinations closed under sensor deletion

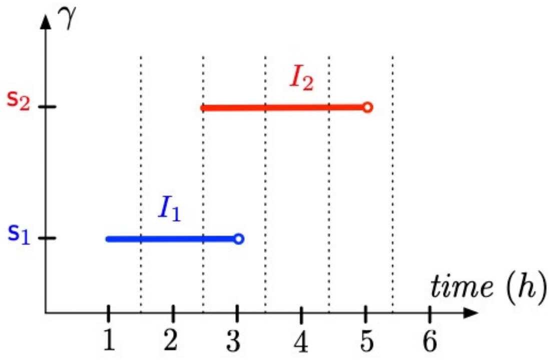


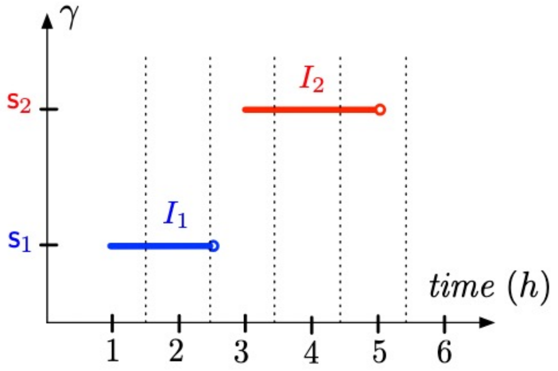
	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9	α_{10}	α_{11}	α_{12}	α_{13}
1	🟩												
2				🟩									
3					🟩								
4						🟩							
5							🟩						
6								🟩					
7									🟩				
8										🟩			
9													🟩
10	🟩	🟩											
11	🟩			🟩									
12	🟩					🟩							
13	🟩						🟩						
14				🟩	🟩								
15				🟩			🟩						
16					🟩		🟩						
17					🟩			🟩					
18						🟩	🟩						
19						🟩			🟩				
20							🟩	🟩					
21							🟩		🟩				
22							🟩			🟩			
23							🟩						🟩
24								🟩					🟩
25									🟩	🟩			
26										🟩			🟩
27												🟩	🟩
28	🟩	🟩	🟩										
29	🟩	🟩		🟩									
30	🟩	🟩				🟩							
31	🟩	🟩					🟩						
32	🟩			🟩			🟩						
33	🟩					🟩	🟩						
34				🟩	🟩		🟩						
35				🟩	🟩			🟩					
36				🟩			🟩			🟩			
37					🟩		🟩	🟩					
38						🟩	🟩	🟩					
39						🟩	🟩		🟩				
40						🟩			🟩	🟩			
41							🟩	🟩					🟩
42							🟩		🟩	🟩			
43							🟩			🟩			🟩
44							🟩					🟩	🟩
45								🟩				🟩	🟩
46										🟩		🟩	🟩
47											🟩	🟩	🟩
48	🟩	🟩	🟩	🟩									
49	🟩	🟩	🟩			🟩							
50	🟩	🟩	🟩				🟩						
51	🟩	🟩		🟩			🟩						
52	🟩	🟩				🟩	🟩						
53				🟩	🟩		🟩	🟩					
54						🟩	🟩		🟩	🟩			
55							🟩	🟩				🟩	🟩
56							🟩			🟩		🟩	🟩
57							🟩				🟩	🟩	🟩
58								🟩			🟩	🟩	🟩
59										🟩	🟩	🟩	🟩
60	🟩	🟩	🟩	🟩	🟩								
61	🟩	🟩	🟩	🟩		🟩							
62	🟩	🟩	🟩	🟩			🟩						
63	🟩	🟩	🟩			🟩	🟩						
64	🟩	🟩	🟩			🟩			🟩				
65					🟩			🟩			🟩	🟩	🟩
66							🟩	🟩			🟩	🟩	🟩
67							🟩			🟩	🟩	🟩	🟩
68								🟩		🟩	🟩	🟩	🟩
69									🟩	🟩	🟩	🟩	🟩
70	🟩	🟩	🟩	🟩	🟩	🟩							
71	🟩	🟩	🟩	🟩	🟩		🟩						
72	🟩	🟩	🟩	🟩	🟩			🟩					
73	🟩	🟩	🟩	🟩		🟩	🟩						
74	🟩	🟩	🟩	🟩		🟩			🟩				
75	🟩	🟩	🟩			🟩	🟩		🟩				
76	🟩	🟩	🟩			🟩			🟩	🟩			
77				🟩	🟩			🟩			🟩	🟩	🟩
78					🟩		🟩	🟩			🟩	🟩	🟩
79					🟩			🟩		🟩	🟩	🟩	🟩
80						🟩			🟩	🟩	🟩	🟩	🟩
81							🟩	🟩		🟩	🟩	🟩	🟩
82							🟩		🟩	🟩	🟩	🟩	🟩
83								🟩	🟩	🟩	🟩	🟩	🟩
84	🟩	🟩	🟩	🟩	🟩	🟩	🟩						
85	🟩	🟩	🟩	🟩	🟩		🟩	🟩					
86	🟩	🟩	🟩	🟩		🟩	🟩		🟩				
87	🟩	🟩	🟩			🟩	🟩		🟩	🟩			
88				🟩	🟩		🟩	🟩			🟩	🟩	🟩
89					🟩		🟩	🟩		🟩	🟩	🟩	🟩
90						🟩	🟩		🟩	🟩	🟩	🟩	🟩
91							🟩	🟩	🟩	🟩	🟩	🟩	🟩
92	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩					
93	🟩	🟩	🟩	🟩		🟩	🟩		🟩	🟩			
94				🟩	🟩		🟩	🟩		🟩	🟩	🟩	🟩
95						🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
96	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩

Temporal Robustness

- We have an α_{12}-path, $\alpha_{12}(I_1, I_2)$ (left) measured by the hour
- We **double the frequency**, and c is satisfied at 2:30 at s_2, and at 3:00 at s_1 (center) OR
- We **double the frequency**, and c is satisfied at 2:30 at s_1 and not at 2:30 at s_2(right)
- Then, $\alpha_{12}(I_1, I_2)$ is transformed into $\alpha_{11}(I_1, I_2)$ (center) or $\alpha_{13}(I_1, I_2)$ (right)
- A **robust** version would be $\alpha_{11} \cup \alpha_{12} \cup \alpha_{13}$

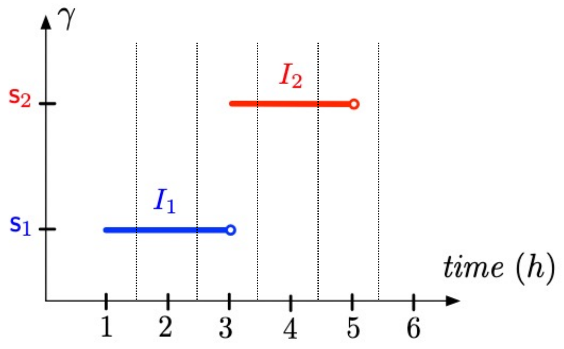


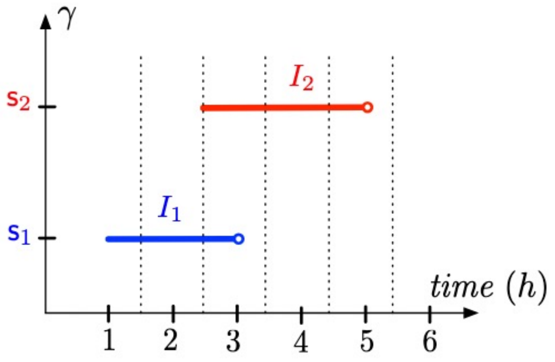


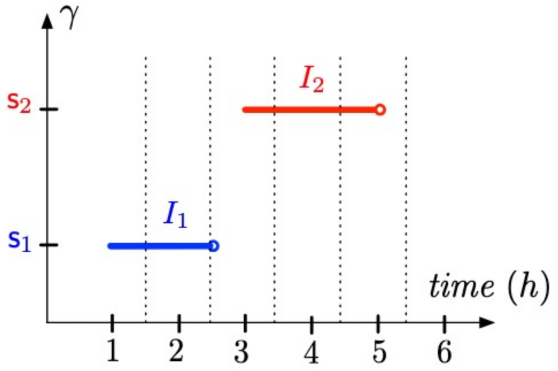


Spatial Robustness

- We have an α_{12}-path, $\alpha_{12}(I_1, I_2)$ (left) reflecting a forward movement of a phenomenon
  - – I_2 can be seen as a "delayed" version of I_1 (with delay d)
  - – *Depending on the distance between sensors*
- If sensors were **placed closer to each other** there may be an overlap => $\alpha_{11}(I_1, I_2)$
- If sensors were placed **farther from each other**, there may be no overlap => $\alpha_{13}(I_1, I_2)$
  - – This is because the delay would prevent the phenomenon to reach s2 on time
- A **robust** version would include $\alpha_{11} \cup \alpha_{12} \cup \alpha_{13}$



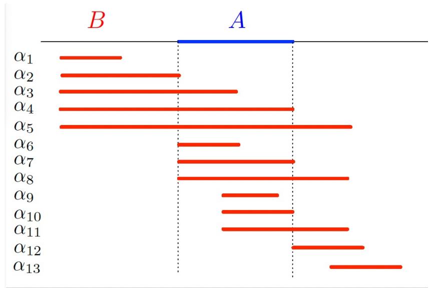




Robustification Algorithm

- Rationale: when a relation involves matching start or end points of intervals, we also include the α_i that corresponds to starting (or ending) a bit before or after that matching point
- Two options: go from coarser to finer granularity or vice versa
- Repeat the transformations until a fixed point is reached
- From coarser to finer:

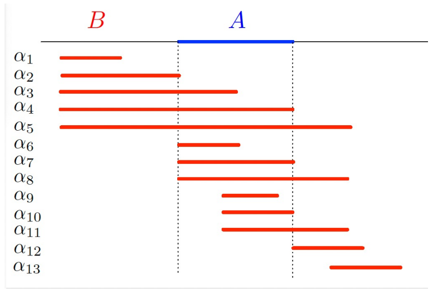
If Contains	Then Add
α_2	$\alpha_1 \cup \alpha_3$
α_4	$\alpha_3 \cup \alpha_5$
α_6	$\alpha_3 \cup \alpha_9$
α_7	$\alpha_6 \cup \alpha_8$
α_7	$\alpha_4 \cup \alpha_{10}$
α_8	$\alpha_5 \cup \alpha_{11}$
α_{10}	$\alpha_9 \cup \alpha_{11}$
α_{12}	$\alpha_{11} \cup \alpha_{13}$



Robustification Algorithm

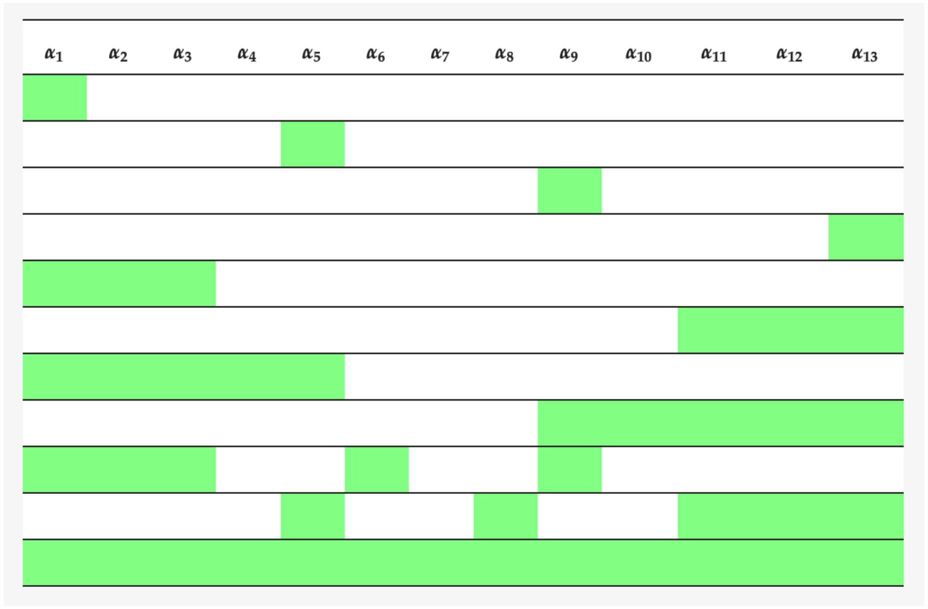
- Rationale: when a relation involves matching start or end points of intervals, we also include the α_i that corresponds to starting (or ending) a bit before or after that matching point
- Two options: go from coarser to finer granularity or vice versa
- Repeat the transformations until a fix point is reached
- From finer to coarser:

If Contains	Then Add
$\alpha_1 \cup \alpha_3$	α_2
$\alpha_3 \cup \alpha_5$	α_4
$\alpha_3 \cup \alpha_9$	α_6
$\alpha_6 \cup \alpha_8$	α_7
$\alpha_4 \cup \alpha_{10}$	α_7
$\alpha_5 \cup \alpha_{11}$	α_8
$\alpha_9 \cup \alpha_{11}$	α_{10}
$\alpha_{11} \cup \alpha_{13}$	α_{12}



Combination of properties

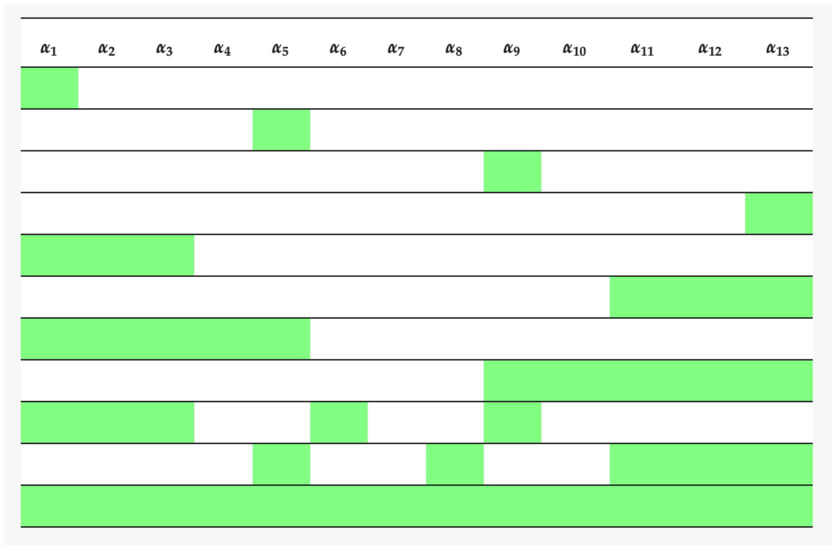
- Goal: Reduce the initial 8192 elements of the Allen interval algebra to obtain a manageable number of cases that could be recognized as real-world situations
- There are eleven combinations of the Allen interval algebra that are both robust and closed under sensor deletion



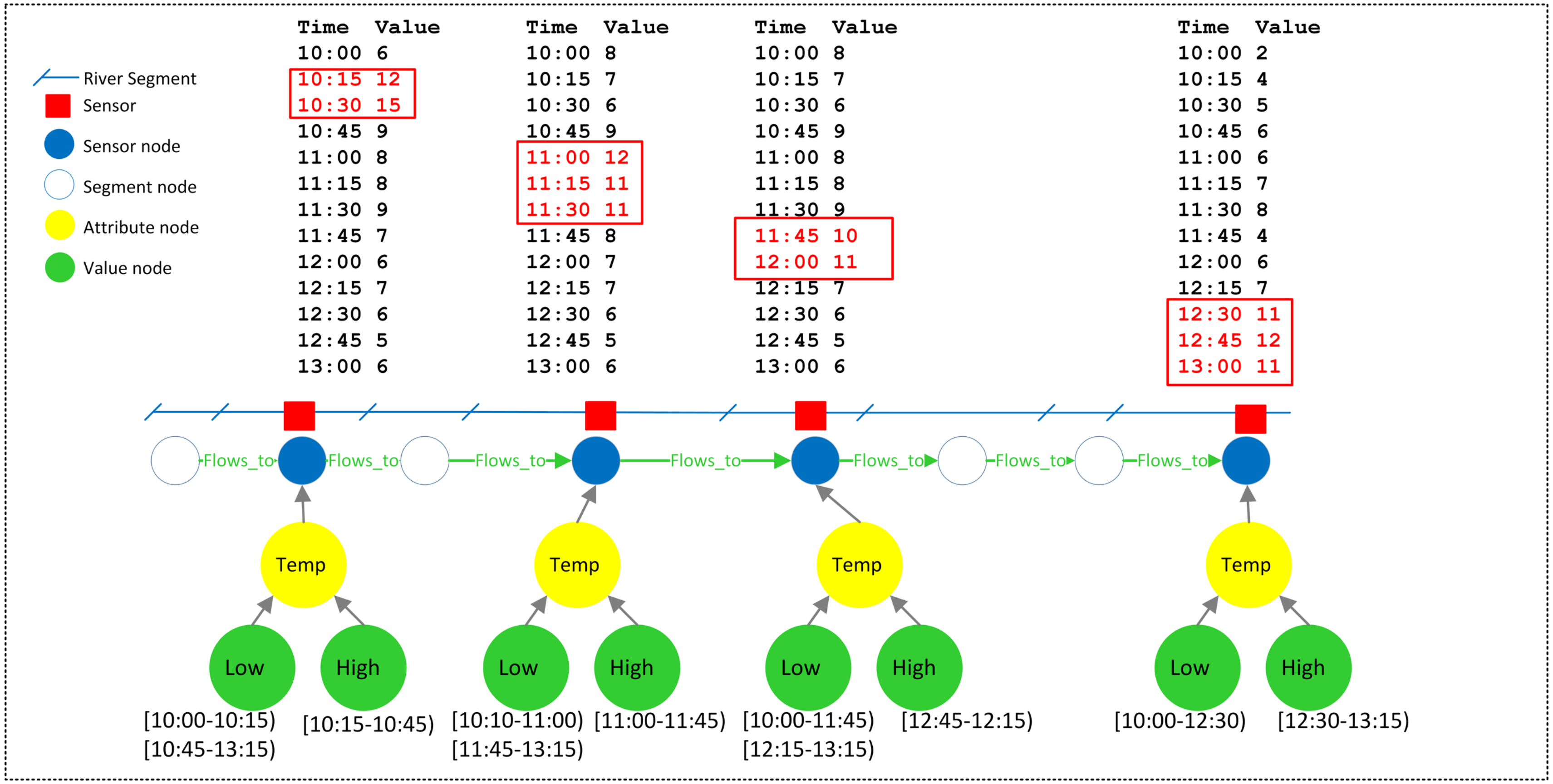
α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9	α_{10}	α_{11}	α_{12}	α_{13}
🟩												
				🟩								
								🟩				
												🟩
🟩	🟩	🟩										
										🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩								
								🟩	🟩	🟩	🟩	🟩
🟩	🟩	🟩			🟩			🟩				
				🟩			🟩			🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩

Paths that are CUSD and ROBUST

- α_{13}-paths: (γ, ((s_1, l_1), (s_2, l_2), . . . , (s_k, l_k)), we have $l_1 < l2 < \cdots < l_k$, where < means "strictly after" => Consecutive paths
- α_1-paths: the backward version of α_{13}-paths



Sensor network Consecutive path



Time	Value
10:00	6
10:15	12
10:30	15
10:45	9
11:00	8
11:15	8
11:30	9
11:45	7
12:00	6
12:15	7
12:30	6
12:45	5
13:00	6

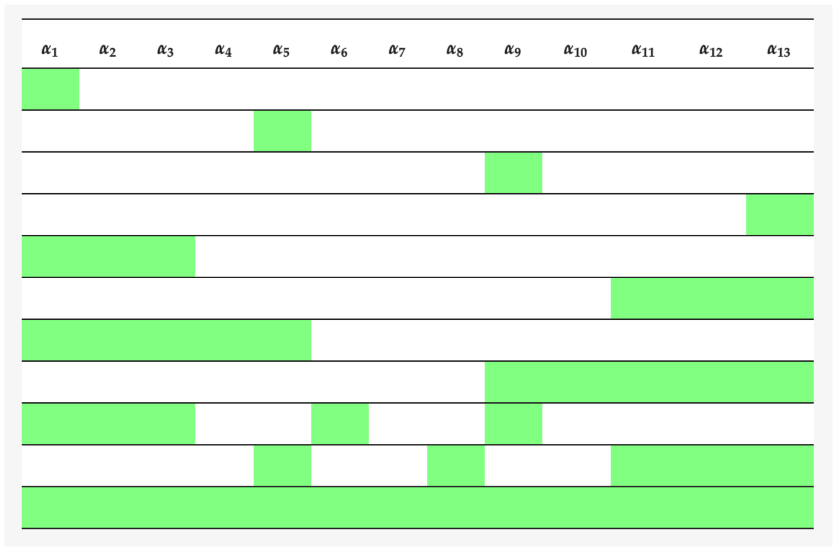
Time	Value
10:00	8
10:15	7
10:30	6
10:45	9
11:00	12
11:15	11
11:30	11
11:45	8
12:00	7
12:15	7
12:30	6
12:45	5
13:00	6

Time	Value
10:00	8
10:15	7
10:30	6
10:45	9
11:00	8
11:15	8
11:30	9
11:45	10
12:00	11
12:15	7
12:30	6
12:45	5
13:00	6

Time	Value
10:00	2
10:15	4
10:30	5
10:45	6
11:00	6
11:15	7
11:30	8
11:45	4
12:00	6
12:15	7
12:30	11
12:45	12
13:00	11

Paths that are CUSD and ROBUST

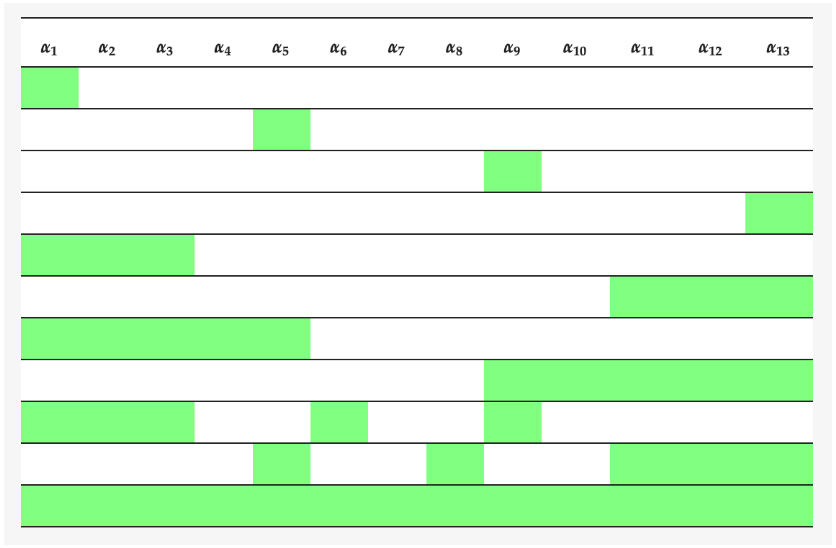
- α_9-paths: (γ, ((s_1, I_1), (s_2, I_2), . . . , (s_k, I_k)), we have $I_1 \supset I_2 \supset \cdots \supset I_k$, where the inclusions are strict. Forward paths that reflect a phenomenon that moves forward through the transportation network and diminishes in strength, e.g., salinity that gets dissolved as it moves in along the river. These are continuous paths
- α_5-paths: the backward version of α_9-paths

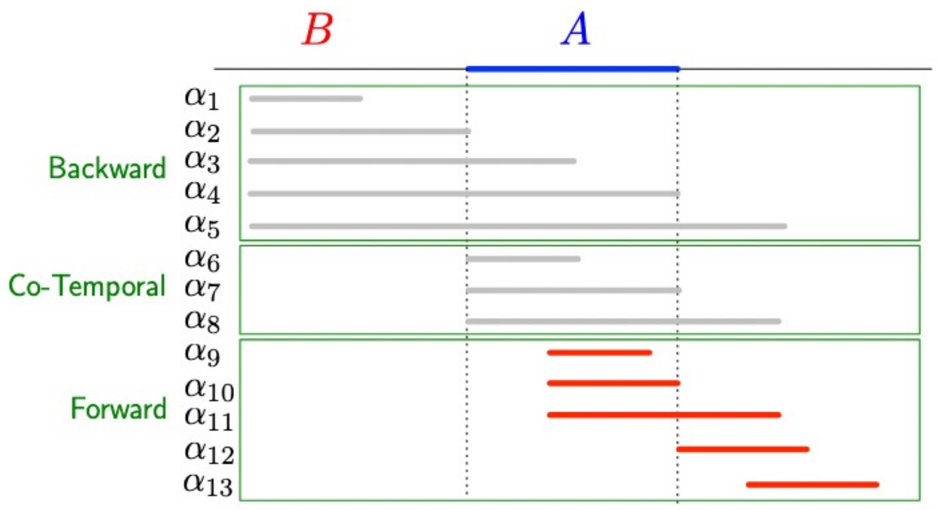


α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9	α_{10}	α_{11}	α_{12}	α_{13}
🟩												
				🟩								
								🟩				
												🟩
🟩	🟩	🟩										
										🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩								
								🟩	🟩	🟩	🟩	🟩
🟩	🟩	🟩			🟩			🟩				
				🟩			🟩			🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩

Paths that are CUSD and ROBUST

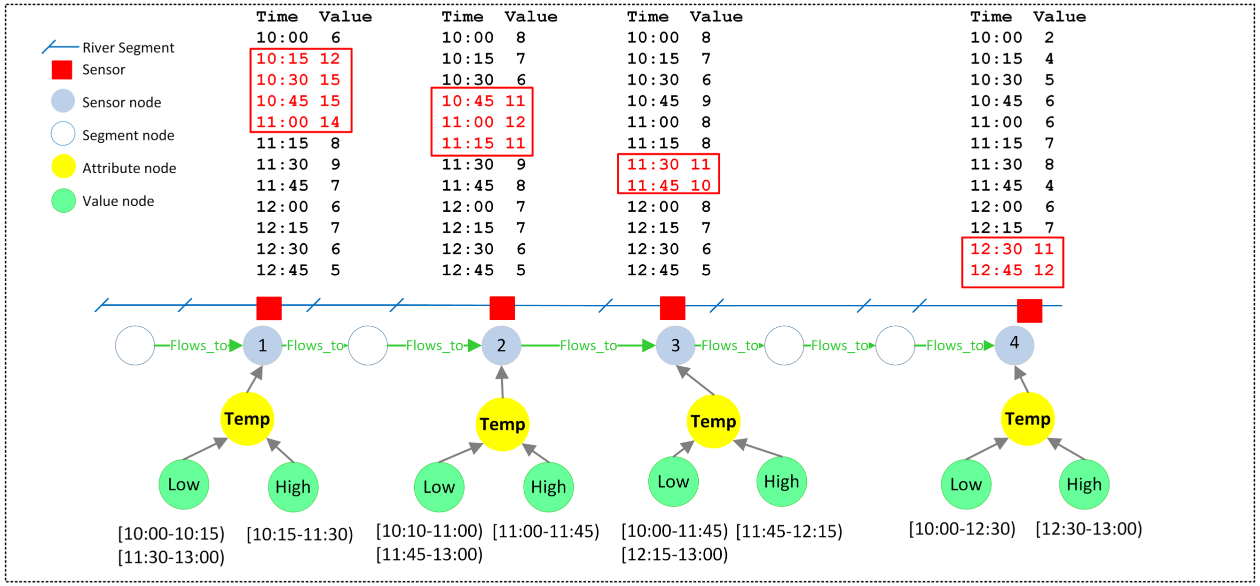
- $(\alpha_9 \cup \alpha_{10} \cup \alpha_{11} \cup \alpha_{12} \cup \alpha_{13})$ -paths: $(\gamma, ((s_1, l_1), (s_2, l_2), \ldots, (s_k, l_k))$
- Called Flow paths, reflect a phenomenon that moves forward through the transportation network, is detected at a given sensor and starts to be detected at the next consecutive one with a delay that usually corresponds to a network-related delay.





Sensor network Flow path

- A *High* value of Temperature detected in one sensor earlier than the first time it is detected in the next one
- The measurements overlap in the first pair of sensors but not in the other pairs
- Includes a consecutive path (with a gap due to a network delay)



Time	Value
10:00	6
10:15	12
10:30	15
10:45	15
11:00	14
11:15	8
11:30	9
11:45	7
12:00	6
12:15	7
12:30	6
12:45	5

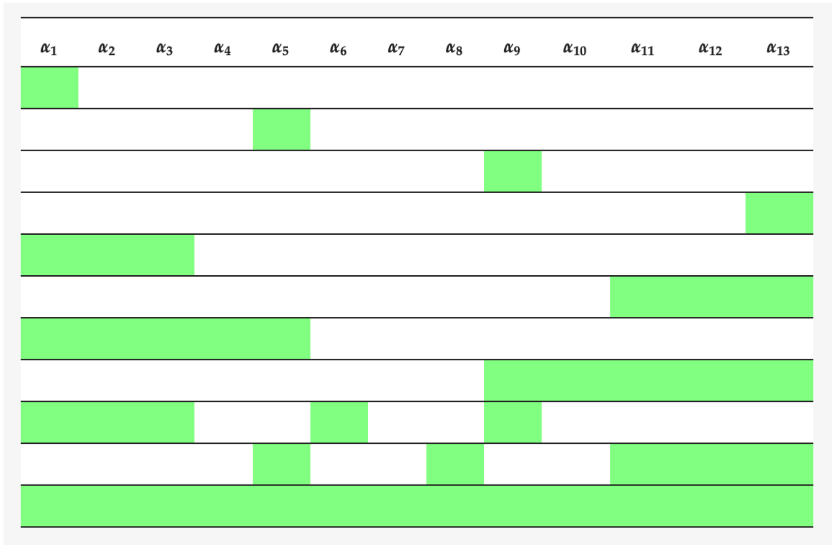
Time	Value
10:00	8
10:15	7
10:30	6
10:45	11
11:00	12
11:15	11
11:30	9
11:45	8
12:00	7
12:15	7
12:30	6
12:45	5

Time	Value
10:00	8
10:15	7
10:30	6
10:45	9
11:00	8
11:15	8
11:30	11
11:45	10
12:00	8
12:15	7
12:30	6
12:45	5

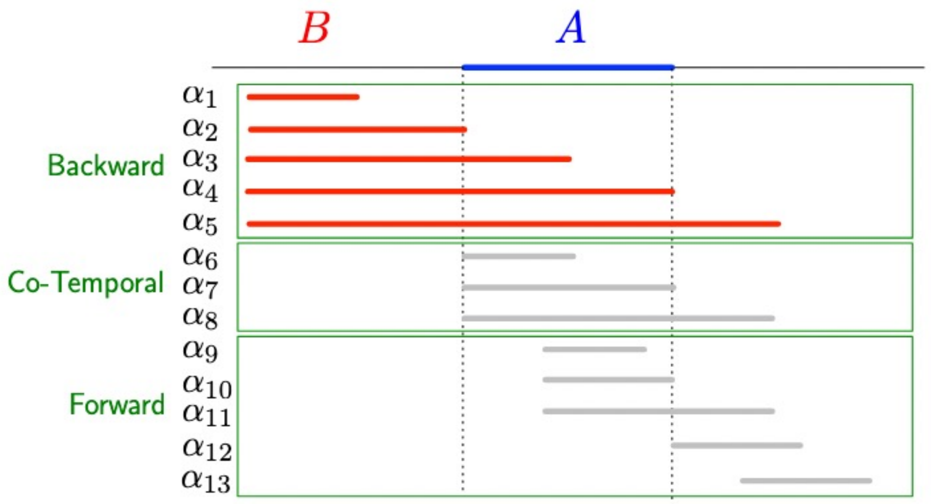
Time	Value
10:00	2
10:15	4
10:30	5
10:45	6
11:00	6
11:15	7
11:30	8
11:45	4
12:00	6
12:15	7
12:30	11
12:45	12

Paths that are CUSD and ROBUST

- ($\alpha_1 \cup \alpha_2 \cup \alpha_3 \cup \alpha_4 \cup \alpha_5$) - paths: the backward version of Flow paths
- Examples: A flock of salmon swimming upstream in a river system, or a traffic jam that propagates backward on a road network

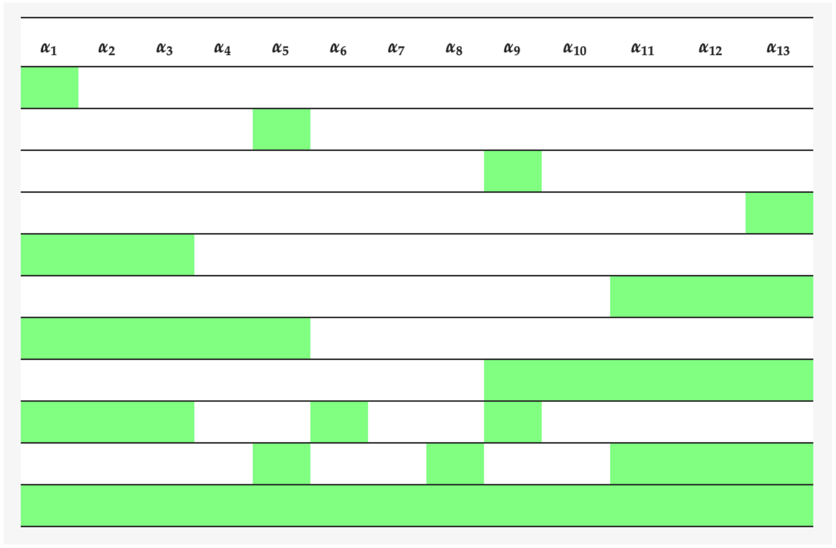


α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9	α_{10}	α_{11}	α_{12}	α_{13}
🟩												
				🟩								
								🟩				
												🟩
🟩	🟩	🟩										
										🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩								
								🟩	🟩	🟩	🟩	🟩
🟩	🟩	🟩			🟩			🟩				
				🟩			🟩			🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩



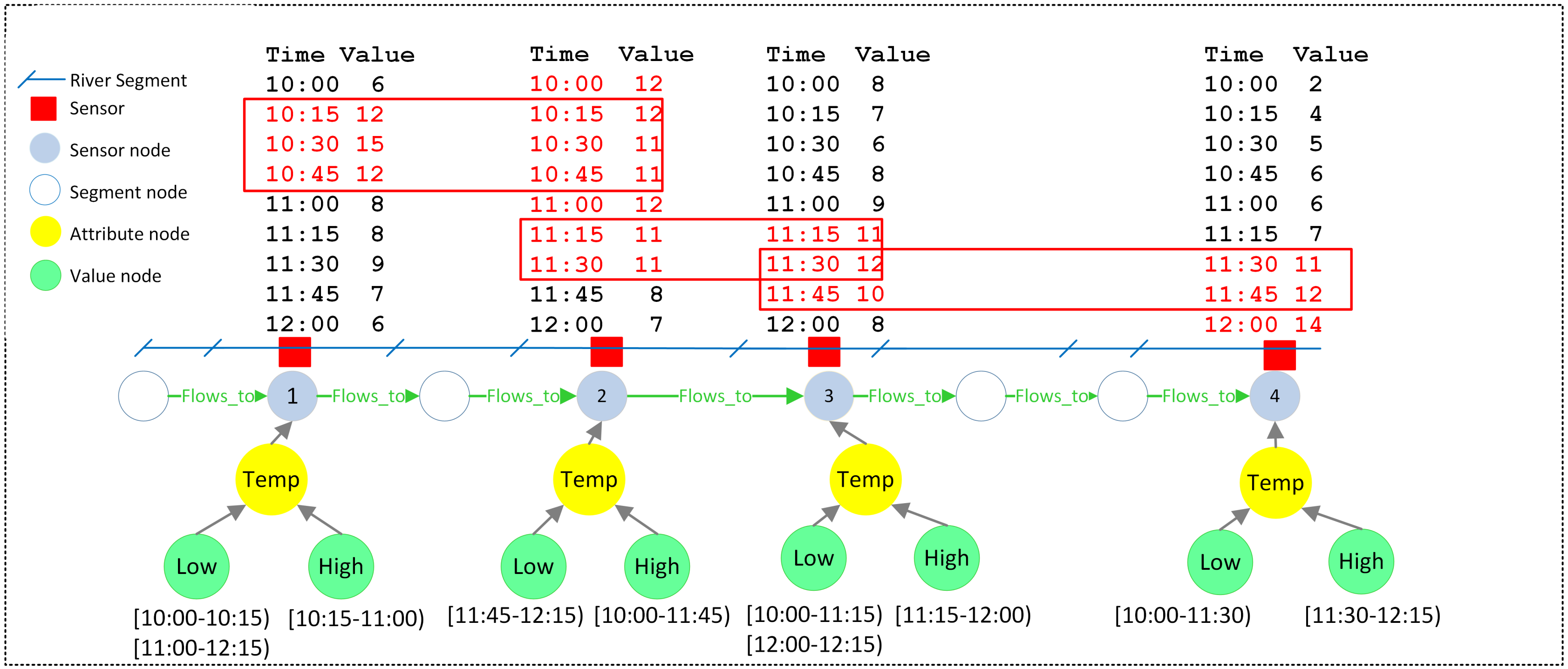
Paths that are not CUSD or robust (or both)

- $\alpha_{3\text{->}11}$- paths: pairwise continuous paths
- Neither robust nor transitive (e.g., in an α_{11}-path, we delete an intermediate sensor, and we get an α_{13}-path)
- However, they capture situations where every pair of consecutive intervals has non-empty intersection, which can arise in real-world situations



α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9	α_{10}	α_{11}	α_{12}	α_{13}
🟩												
				🟩								
								🟩				
												🟩
🟩	🟩	🟩										
										🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩								
								🟩	🟩	🟩	🟩	🟩
🟩	🟩	🟩			🟩			🟩				
				🟩			🟩			🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩

Sensor network Pairwise Continuous path



Time	Value
10:00	6
10:15	12
10:30	15
10:45	12
11:00	8
11:15	8
11:30	9
11:45	7
12:00	6

Time	Value
10:00	12
10:15	12
10:30	11
10:45	11
11:00	12
11:15	11
11:30	11
11:45	8
12:00	7

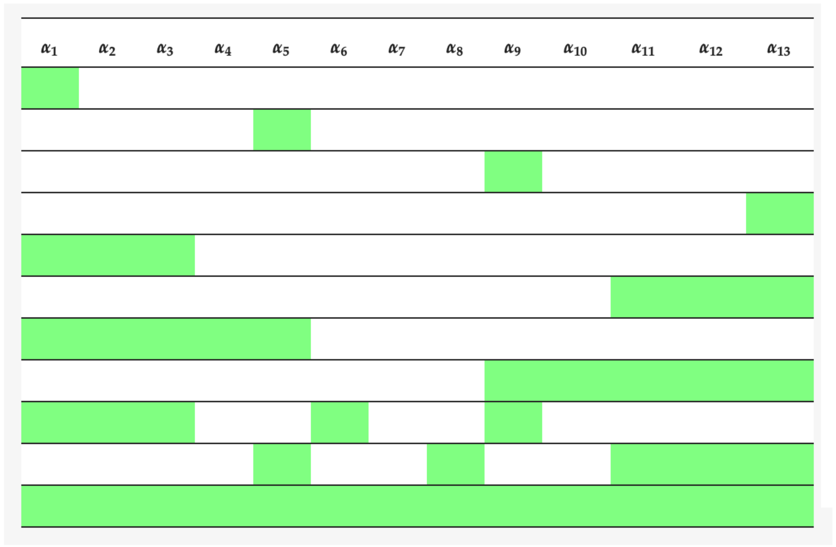
Time	Value
10:00	8
10:15	7
10:30	6
10:45	8
11:00	9
11:15	11
11:30	12
11:45	10
12:00	8

Time	Value
10:00	2
10:15	4
10:30	5
10:45	6
11:00	6
11:15	7
11:30	11
11:45	12
12:00	14

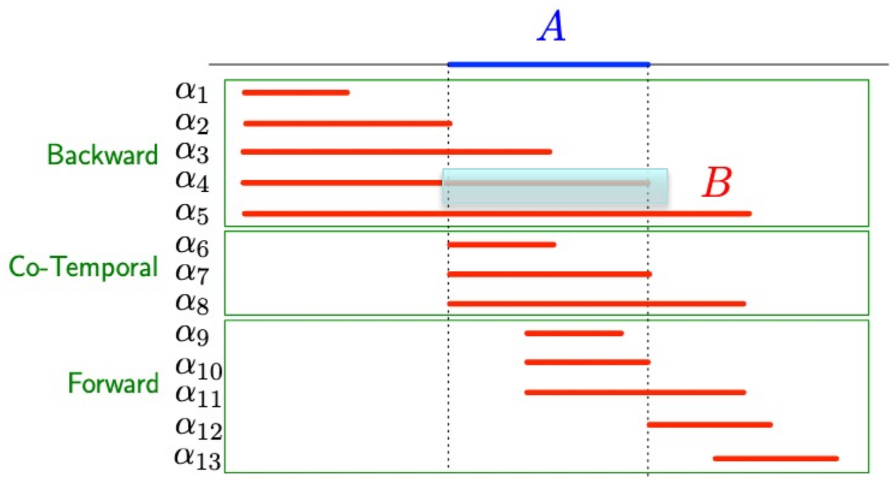
- If we delete Sensor 2, we get a consecutive path

Paths that are not CUSD or robust (or both)

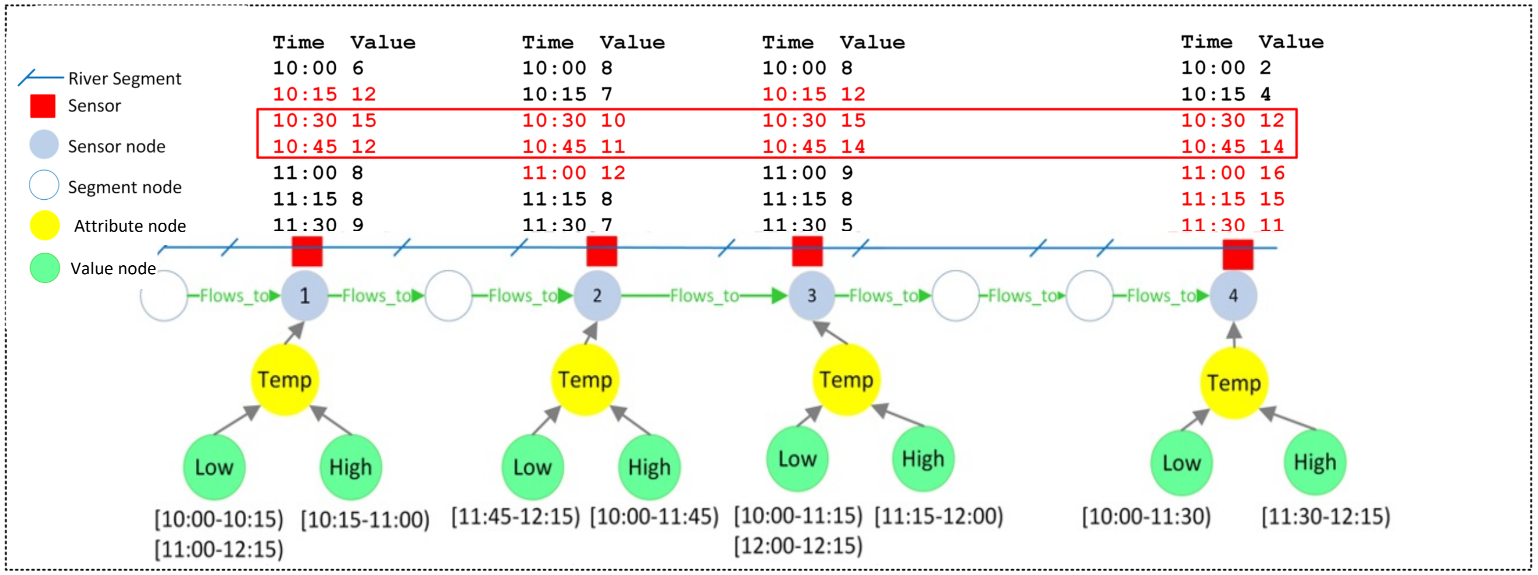
- Also, maximal sub-α_7-paths are Continuous paths
- Maximal sub-α_7-paths **are not robust**
- Nevertheless, they may capture interesting situations, e.g., an event that occurs simultaneously along a path of sensors



α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9	α_{10}	α_{11}	α_{12}	α_{13}
🟩												
				🟩								
								🟩				
												🟩
🟩	🟩	🟩										
										🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩								
								🟩	🟩	🟩	🟩	🟩
🟩	🟩	🟩			🟩			🟩				
				🟩			🟩			🟩	🟩	🟩
🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩



Sensor network Continuous path



Time	Value	Time	Value	Time	Value	Time	Value
10:00	6	10:00	8	10:00	8	10:00	2
10:15	12	10:15	7	10:15	12	10:15	4
10:30	15	10:30	10	10:30	15	10:30	12
10:45	12	10:45	11	10:45	14	10:45	14
11:00	8	11:00	12	11:00	9	11:00	16
11:15	8	11:15	8	11:15	8	11:15	15
11:30	9	11:30	7	11:30	5	11:30	11

- Red indicates a value of *High* for the variable Temperature

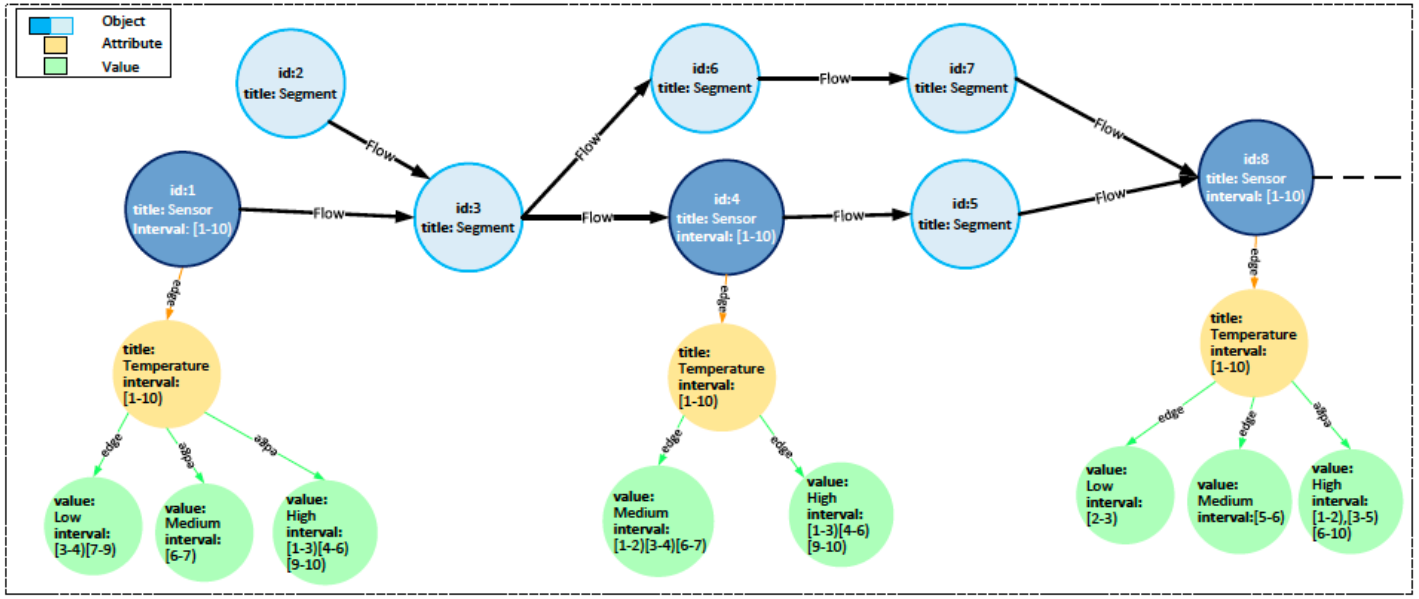
Querying sensor networks

- We extend de T-GQL language to address sensor networks
- Goal: Querying for paths is a sensor network, or finding the set of relations holding between every pair of consecutive sensors, i.e., the α–paths
- The variable being measured must be indicated
- Example: to find a path where the temperature value is high:

```
SELECT paths
MATCH (s1: Sensor),(s2: Sensor),
  paths = alphaPath(( s1 ) -[: Flows *3..5] - > (s2),‘1’, ‘10’,
          `Temperature’, ‘=’, ‘High’)
WHERE s1.id = 1;
```

- In this case, `‘1’` and `‘10’` correspond to the window query time interval. The parameters `` `Temperature’, ‘=’, ‘High’ `` represent the condition Temperature = High

Logical model: TGraph for sensor networks


Object
Attribute
Value
id:2 title: Segment
id:1 title: Sensor Interval: [1-10)
id:3 title: Segment
id:6 title: Segment
id:7 title: Segment
id:4 title: Sensor interval: [1-10)
id:5 title: Segment
id:8 title: Sensor interval: [1-10)
Flow
edge
title: Temperature interval: [1-10)
value: Low interval: [3-4)[7-9)
value: Medium interval: [6-7)
value: High interval: [1-3)[4-6) [9-10)
value: Medium interval: [1-2)[3-4)[6-7)
value: High interval: [1-3)[4-6) [9-10)
value: Low interval: [2-3)
value: Medium interval:[5-6)
value: High interval: [1-2),[3-5) [6-10)

Logical model: TGraph for sensor networks

- Based on TGraph, we define the SNGraph (Sensor Network Temporal Graph)
- A structure G(Ns, Na, Nv, E) , Ns, Na, and Nv sets of nodes, denoted Segment, Attribute, and Value nodes
  - **Sensor nodes:** Segment nodes that ever contained a sensor
    - In Sensor nodes, *title = 'Sensor'*; *interval:* the time when a sensor worked
    - Properties that do not change over time (static) may exist
  - **Segment (non – sensor) nodes** do not contain the attribute *interval*
  - An **Attribute node** represents a variable measured by the sensors

    title property: the name of the variable; *interval:* its lifespan
  - **Value node**

    value property: the values registered by the sensors

    interval: the period when the measure was valid
  - Edges between Segment nodes represent the flow between two segments; *interval* is the validity period of the edge

Agenda

- **Temporal Graphs in Transportation Networks**
  - Abstract graph model for sensor networks
  - Paths in sensor networks
  - Use case

Use case: the IoW project*

Overview of the Scheldt river, and the nine sensors considered

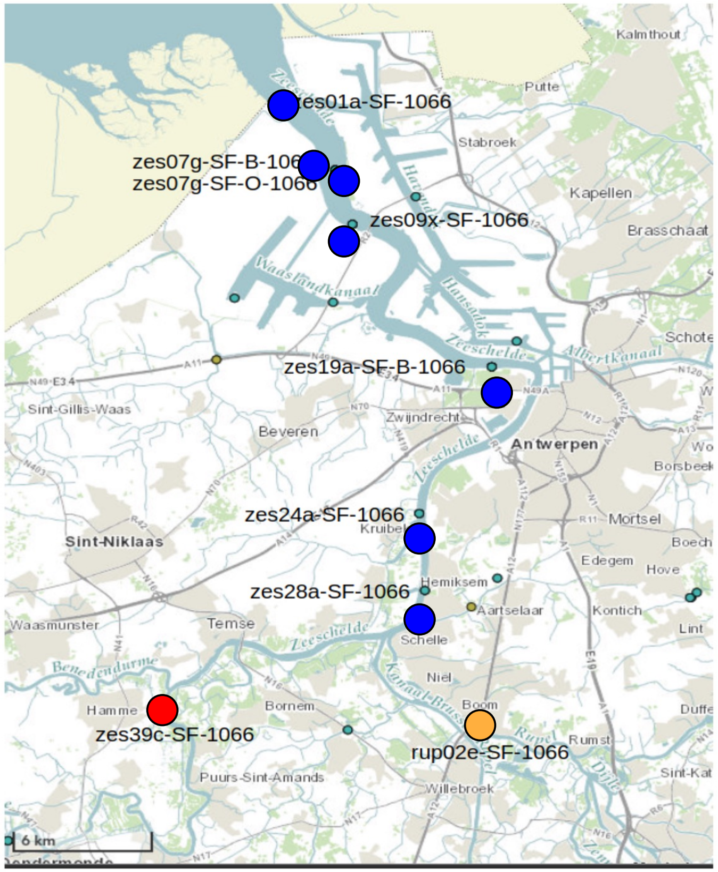


Figure obtained from http://waterinfo.be

* https://www.internetofwater.be/en/what-is-internet-or-water/

Use case: goals

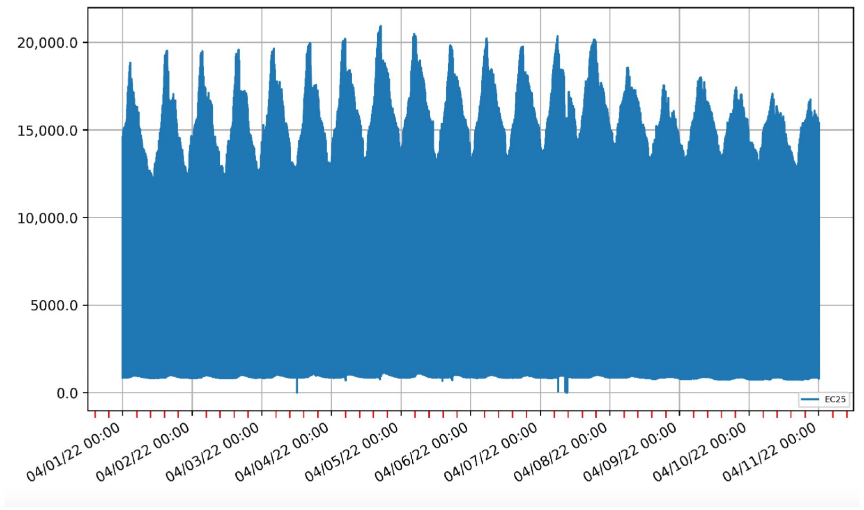
- Problem: The river is influenced by the tidal streams at the North sea
- The river height rises and falls twice a day following the tidal rhythm
- During high tides, close to the shore the water flows in the opposite direction with respect to the natural downstream flow of the river
- The salty sea water merges with the river's fresh water, influencing its salinity with an impact on the water quality, flora, and fauna of the region
- Sensors are used to monitor the river in real-time
- We   focus on the conductivity of the water, which indicates the presence of salt in the water: an increase of the content of salt => an increase of the electrical conductivity of the water
- Hydrologists want to understand how far the salty waters coming from the sea due to high tides, go into the river flow before dissolving into fresh water: this can be captured by temporal paths

Use case: goals

- We look for paths where the salty water starts to be detected when it arrives at the station closest to the sea
- As we move farther from the sea, salinity arrives at the next station, where it is first detected, and this repeats until it cannot be detected anymore, since it dissolves at a certain point
- However, it may still be detected at the first sensor at the same time when it vanishes completely at some point in the river
- It follows that every interval is smaller than the previous one (i.e., at the previous sensor)
- This pattern corresponds to an α_5-path
- If an α_5-path is not found, if at least we find a Backward path, this will show the spread of salinity and will let hydrologist know how far does salinity go and in what time span
- We use T-GQL to find these paths

Use case: data preparation

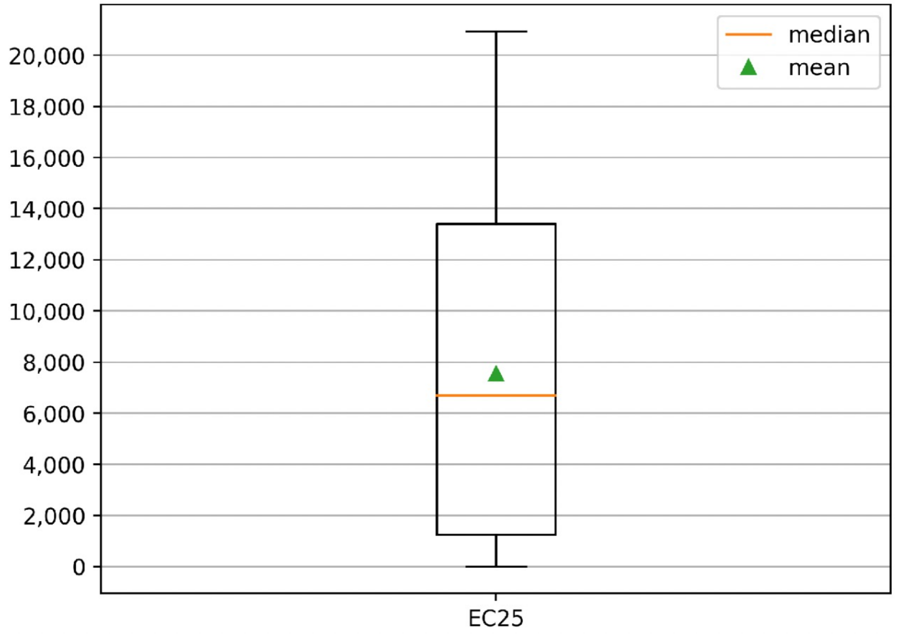
- We start with data exploration
- We can see two daily peaks, reflecting the tidal effect
  - *– the height of these peaks increases from days 1 through 7 and decreases from day 8 onward*



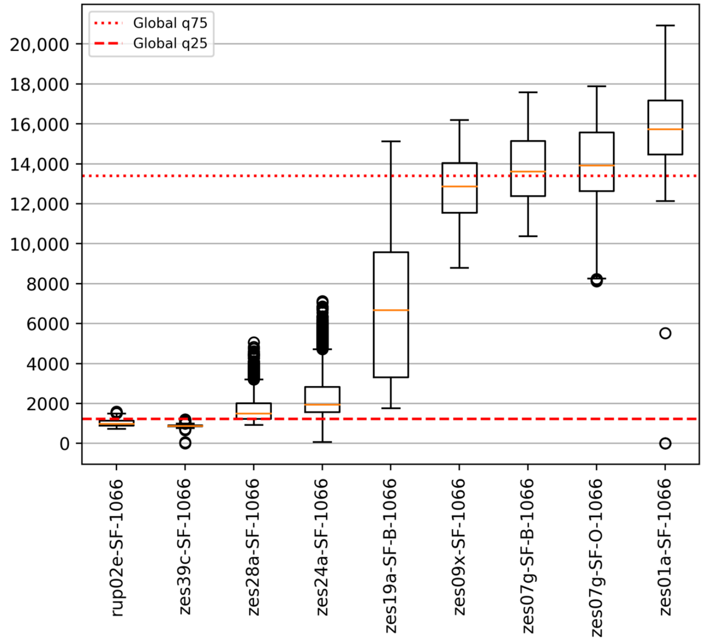
- We use categorical variables => we must set the category boundaries
- Problem: cannot use a unique set of boundaries, let us see why

Use case: data preparation

- Left: global statistics (all sensors considered)
- Right: statistics for each sensor
- We cannot use the same category boundaries for categorization
- Closer to the sea, conductivity is much higher

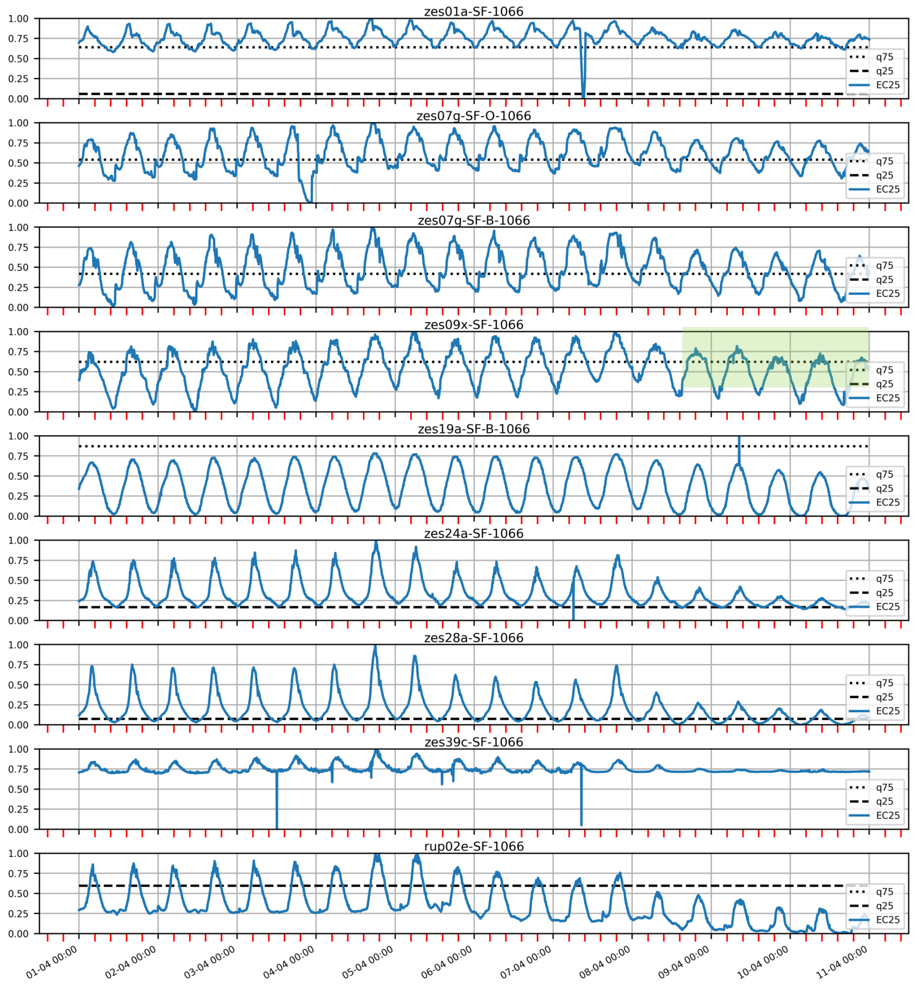


Sea →



Use case: data preparation

- Using global thresholds, all values will be High for stations close to the sea (at the top)
- We must use global AND local thresholds
- Fourth station from the top (in green): oscillate around the 0.75 quartile, using this value as threshold would produce many small intervals for the High category

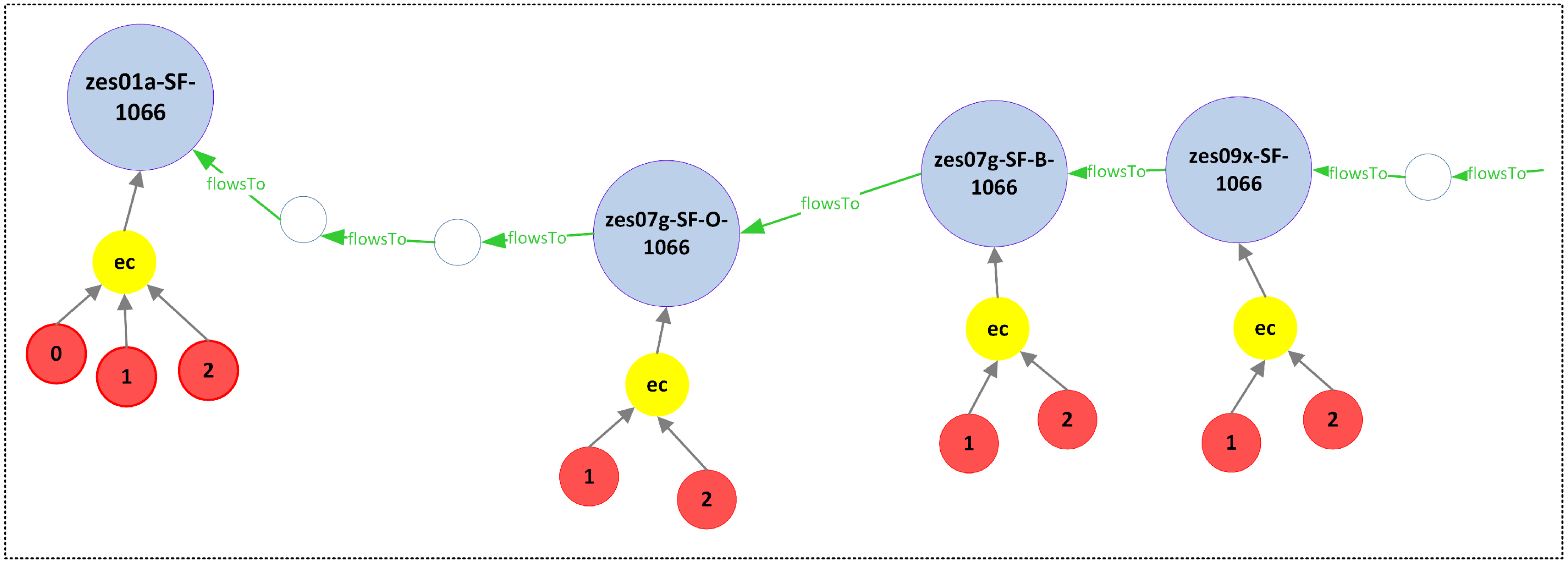


Use case: building the graph

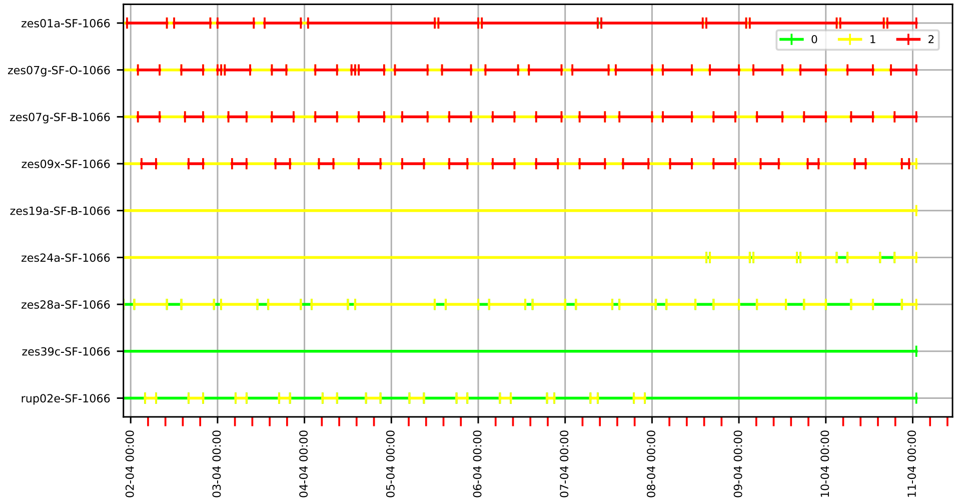
- Granularity is relevant
- We create four graphs:
  - **G10**: Global thresholds, granularity 10 min
  - **G60**: Global thresholds, granularity 60 min
  - **L10**: Local thresholds, Granularity 10 min
  - **L60**: Local thresholds, granularity 60 min
- For every station that measures the **ec** variable (electric conductivity), we create an *attribute* node and connect it to its corresponding sensor node
- For every category associated with that station (0,1,2 stand for low, medium and high), a *value* node connected to the attribute node
- Each value node labeled 0, 1 and 2 will contain a sequence of time intervals indicating when **ec** falls in the category
- EC25: parameter normalized to 25 ^{o}C

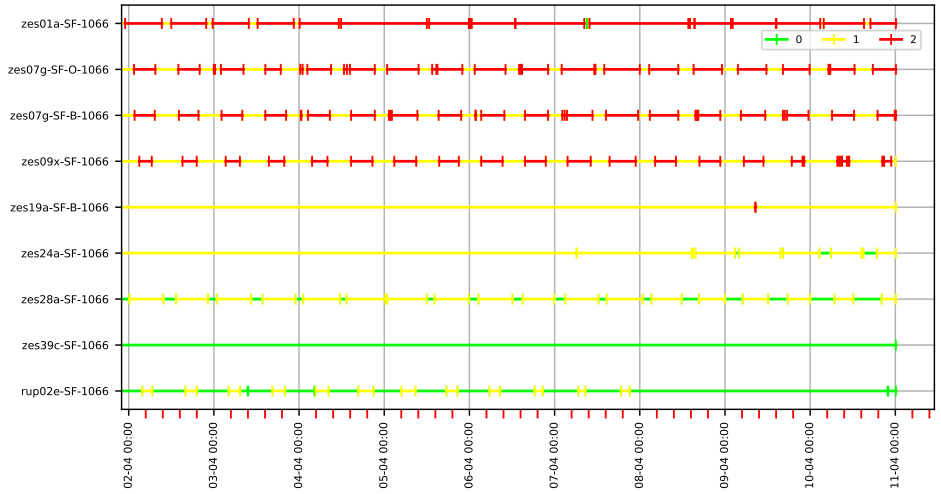
Use case: building the graph

- Neo4j graph for **G60**



Use case: flat graph representation





- Top: sensors closer to the sea
- Flow goes from bottom to top
- Left: intervals for **G60**; Right: intervals for **G10**
- Farther from the sea, less red intervals
- Due to the finer granularity, there are more intervals for each station in **G10** than in **G60**.

Analysis – Finding paths

- Finding Paths in **G60** , **global thresholds with 1-hour granularity**

- How far does salinity go before being dissolved in fresh water?
- We start with **G60** and look for α_5-paths, which would show the dissolution effect along the stations
- If we do not find such a path, we look for a backward path, which would show how salinity spreads along the river
- Some stations, like *zes07g-SF-O-1066* and *zes07g-SF-B-1066* are placed at the same location, likely to find co-temporal paths
- Water rises and falls twice a day; We capture this with a time window of about twelve hours, try to find a path within this window
- If a path is found in **G60,** verify that it is also present in a finer granularity graph, **G10**
- Finally check if we find the path in **L60** (same granularity with local thresholds)

Analysis – Finding paths

- Finding Paths in **G60**

- We want to find α_5-paths with High conductivity, that is, ec= 2
- We know that α_5-paths are closed under sensor deletion (CUSD)
- First pick two stations s_i and s_j, and discard the others
  - If the relation between s_i, s_j is not α_5, we know an α_5-path will not be found
  - Otherwise, we keep adding a station until we find a relation different from α_5 or until there are no more stations with ec = 2
- We first consider sensors s_2 and s_4, discard the others

```
SELECT paths
MATCH (s1:Sensor), (s2:Sensor),
paths = alphaPath((s1)<-[:flowsTo*2]-(s2),
        ‘2022-04-01 02:00’, ‘2022-04-02 11:00’,‘ec’,’=’,‘2’)
WHERE  s1.Name = ’zes07g-SF-O-1066’;
```

Analysis – Finding paths

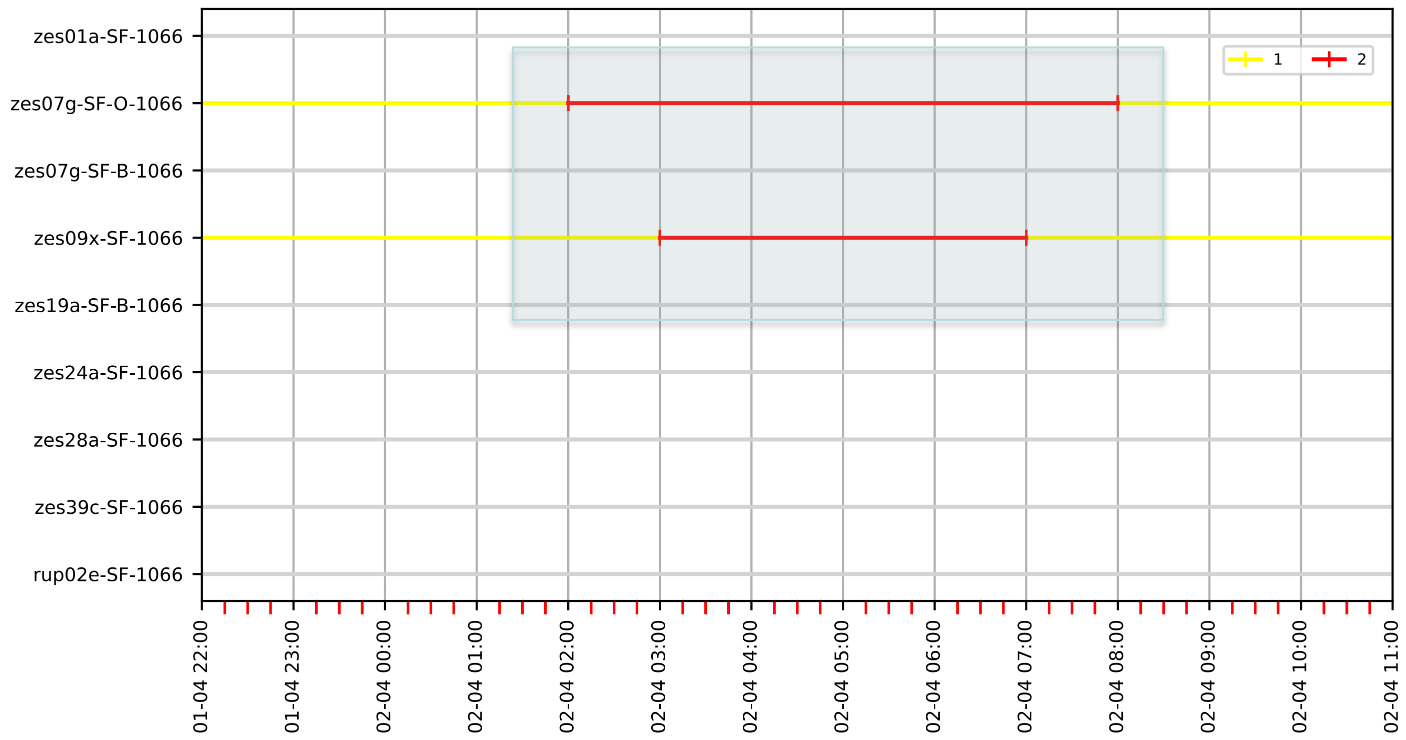
- Finding Paths in **G60**

- The query finds the paths of the kinds that corresponds to the patterns found, in this case, it finds an α_9-path

```
{
"path": [{
  "name": "zes09x - SF -1066", --Sensor S4
  "value": "2",
  "attribute": "ec"},
   {
  "name": "zes07g - SF -O -1066", -- Sensor S2
  "value": "2",
  "attribute": "ec"}],
  "intervals": [
    "2022-04-02 02:00 - 2022-04-02 08:00",
    "2022-04-02 03:00 - 2022-04-02 07:00"
    ],
  "alphas": ["alpha5"] – Relation between S2 and S4
}
```

Analysis – Finding paths

- Finding Paths in **G60**



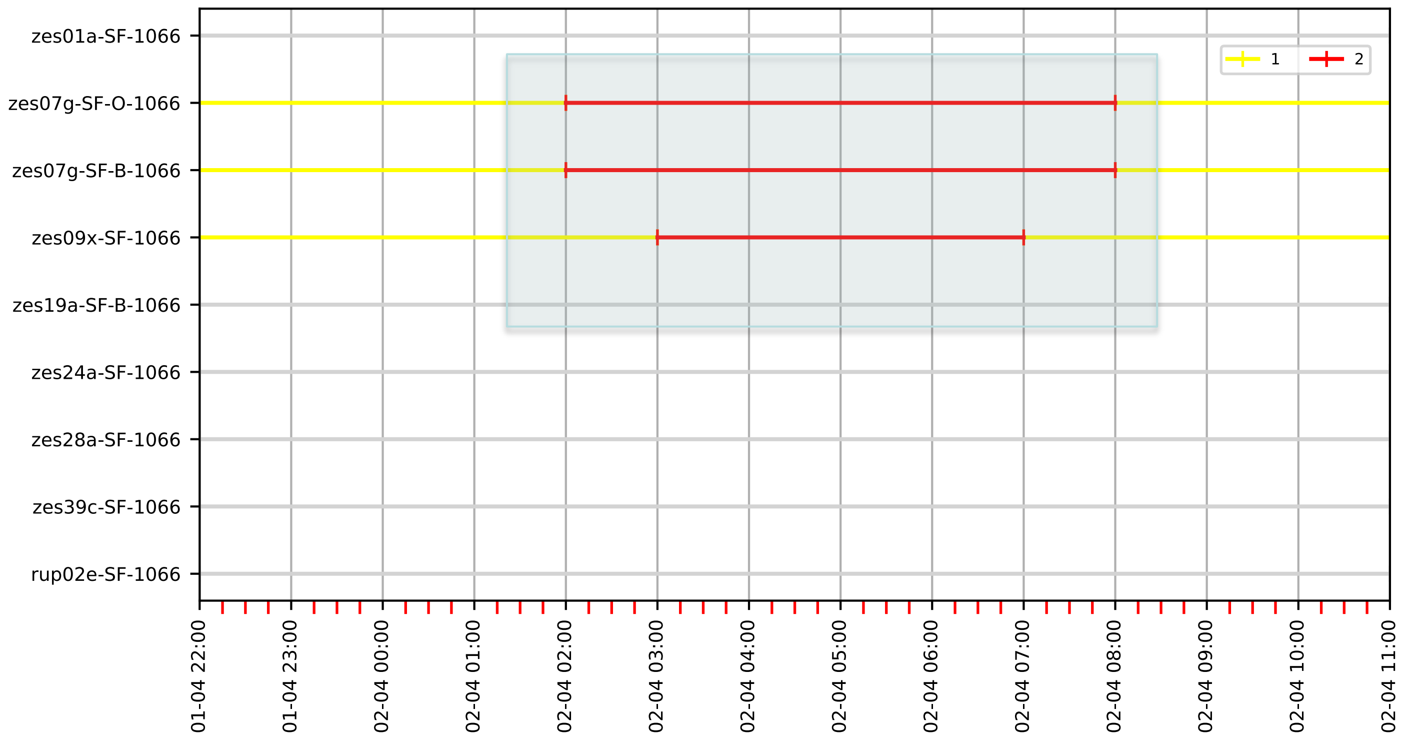
Analysis – Finding paths

- Now we add s_3

```
{
  "path": [{
  "name": "zes09x - SF -1066", -- S4
  "value": "2",
  "attribute": "ec"},
{ "name": "zes07g - SF -B -1066", -- S3
  "value": "2",
  "attribute": "ec"},
{"name": "zes07g - SF -O -1066", -- S2
  "value": "2",
  "attribute": "ec"}
],
  "intervals": [
    "2022-04-02 02:00 - 2022-04-02 08:00",
    "2022-04-02 02:00 - 2022-04-02 08:00",
    "2022-04-02 03:00 - 2022-04-02 07:00” ],
  "alphas": ["alpha5", "alpha7"] – alpha7 between S2 & S3, alpha5
   between S3 & S4}
```

Analysis – Finding paths

- We obtain and $\alpha_{7,5}$-path => stop looking for an α_5-path. Note that s_2 and s_3 are at the same physical location



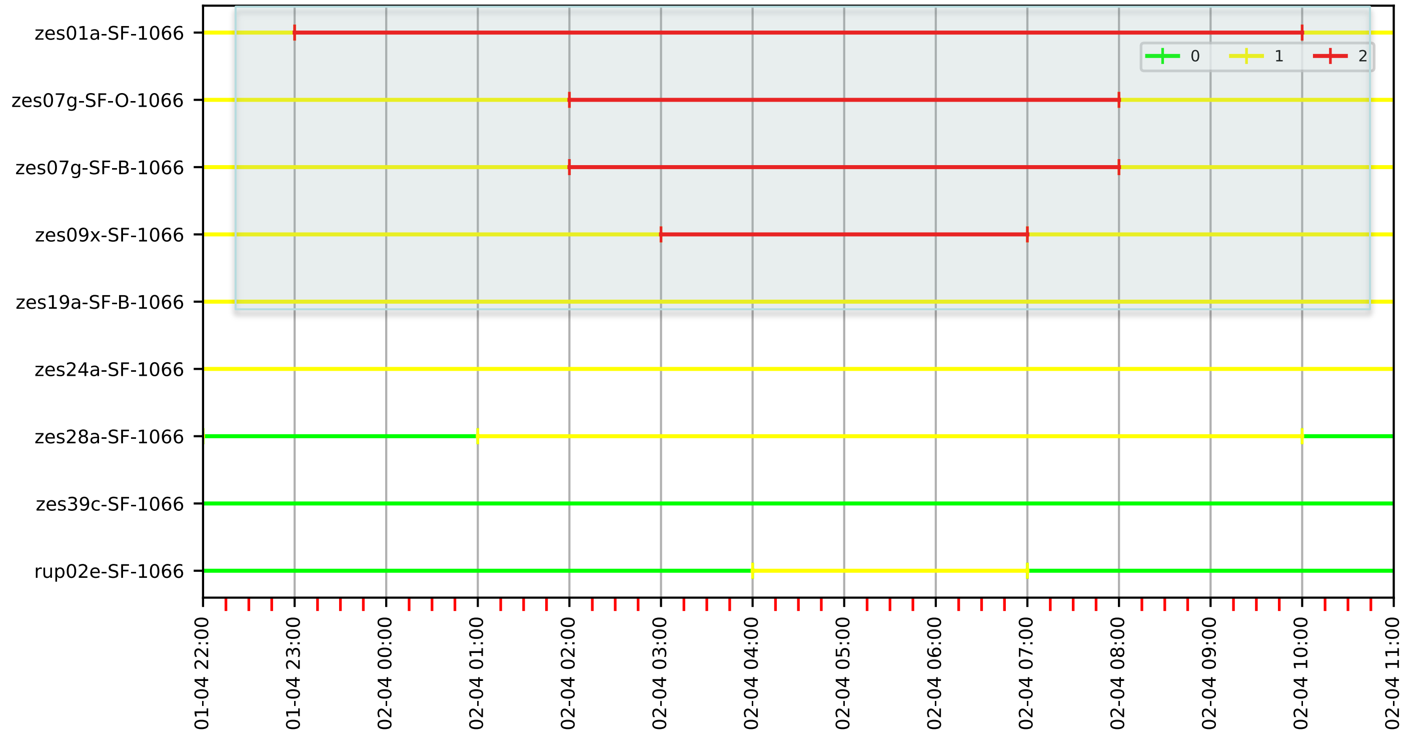
Analysis – Finding paths

- We continue looking for a backward path (we add s1)

```
{
"path": [{
"name": "zes09x - SF -1066", -- S4
"value": "2",
"attribute": "ec"},
{ …
…}
"name": "zes01a - SF -1066", -- S1
"value": "2",
"attribute": "ec"}
],
"intervals": [
  "2022-04-01 23:00 - 2022-04-02 10:00",
  "2022-04-02 02:00 - 2022-04-02 08:00",
  "2022-04-02 02:00 - 2022-04-02 08:00",
  "2022-04-02 03:00 - 2022-04-02 07:00"
  ],
"alphas": ["alpha5", "alpha7", "alpha5"]
}
```

Analysis – Finding paths

- We can see the dissolution effect.
- The same is obtained with G10 => We do not need the 10-minute granularity



Analysis – Finding paths

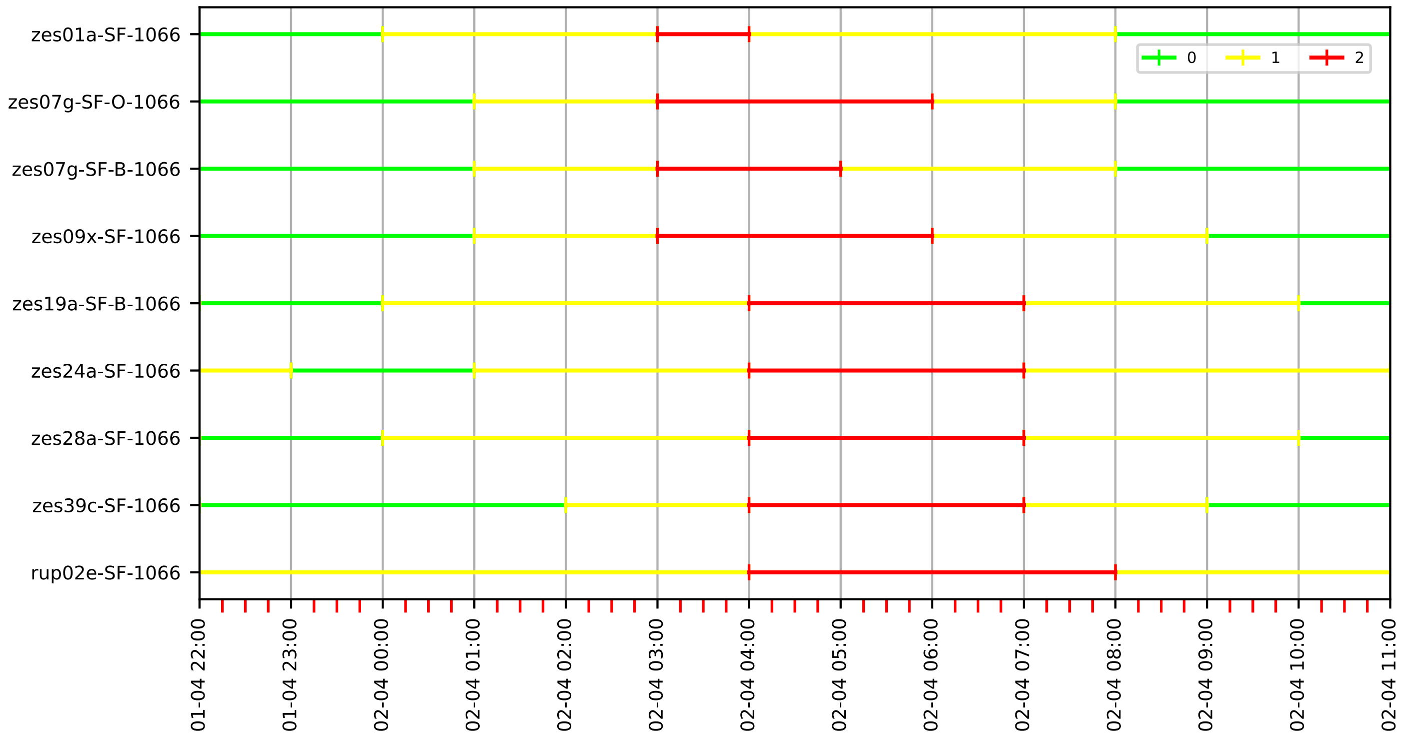
- Finding Paths in **L60, local thresholds with 1-hour granularity**

```
{
  "path": [{
  "name": "zes39c - SF -1066",
 …  "},


  …
{ "name": "zes01a - SF -1066",
  "value": "2",
  "attribute": "ec"}
],
  "intervals": [
    "2022-04-02 03:00 - 2022-04-02 04:00",
     …,
    "2022-04-02 04:00 - 2022-04-02 07:00"],
"alphas":["alpha7","alpha7","alpha7","alpha3","alpha6","alpha8","
alpha6"]
}
…{
```

Analysis – Finding paths

- There are two paths (the last two sensors in different branches)
- L60 does not show dissolution effect, but shows propagation



Agenda

- Introduction and motivation
- Temporal Graph Databases
- Implementation
- Temporal Graphs in Sensor Networks
- **Conclusion**

Conclusion

- A theoretical framework for paths in temporal graphs
  - Data model
  - Query language
  - Characterization of temporal paths based on Allen's algebra
  - High number of combinations
  - Properties to reduce the number of interesting paths
- Extend this theory to sensor networks
- Framework applied to a real-world use case
- Future work:
  - How can we efficiently manage the time series
    - Use time series databases?
  - Other kinds of networks / use cases

* C. Gutiérrez, R. Angles. A Survey on Graph Database Models ACM Computing Surveys, 2008