LDBC Technical Report TR-2021-01

October 2021

Property graphs and paths in GQL: Mathematical definitions

Cite as:

> Green, Alastair; Guagliardo, Paolo and Libkin, Leonid, 2021. “Property graphs and paths in GQL: Mathematical Definitions”, Linked Data Benchmark Council, Technical Report TR-2021-01. https://doi.org/10.54285/ldbc.TZJP7279.

Abstract

This paper provides precise mathematical definitions of a property graph as specified in the proposed GQL international standard, which is an attributed mixed multigraph with loops. It further defines a partially-oriented walk in such a property graph, which is called a path in GQL, as well as restricted classes of such walks (trails, simple/acyclic paths).

Note on publication as an LDBC Technical Report

The original version of this document was submitted to ISO/IEC JTC1 SC32/WG3 (the committee which develops and maintains the SQL and GQL international standards) as document number **WG3:W13-014**, under the title “Definitions of path and subpath in SQL/PGQ and GQL”.[1]

Publication by LDBC does not imply that members of LDBC other than the authors necessarily agree with the content of this report.

> Some of the references (in both documents) to SQL or GQL draft standards or discussion documents produced in the international standards process link to sites which are only accessible to registered process participants. *Any such document can be made available to any member of LDBC*, but we are not able to make them generally available to the public due to the rules of the international standards process conducted by ISO/IEC JTC1.
>
> Please mail info@ldbcouncil.org to request access or to join LDBC.

This report is published both for general interest, and to provide a reference for other works, where the mathematical definitions and historical explanations may be useful.

The report can be read without any reference to the original discussion paper version. However, it may be useful to know that in producing this report:

[1] To obtain a copy of the original paper (whose content is a subset of the content of this report) please email info@ldbcouncil.org or contact the WG3 convenor through the ISO website. Permission is hereby given by LDBC and the authors for WG3 to share that submission with anyone requesting it.

- We have changed the title, and removed a prefatory note on standards procedures.
- We introduce new names for paths with no vertices or with only a single vertex, which we feel will help to avoid ambiguities found historically in the research literature relating to the terms "null path" and "empty path" (which we also used previously)
- We also avoid use of the term "null path" because of possible confusion with the SQL concept of a "null value". This latter point arose in a discussion on the original WG013-014 paper, in the INCITS DM32 Ad-Hoc for SQL/PGQ, and in WG3.

Any contributor to or recipient of this document who is aware of any patent claim that might be infringed by a specification or standard incorporating some or all of the content of this document is invited to notify the LDBC Steering Committee of the details by sending email to info@ldbcouncil.org. At the date of publication LDBC has not been notified of any such patent claims.

Property graphs and paths in GQL: Mathematical definitions

Alastair Green (LDBC Liaison Representative to WG3)
Paolo Guagliardo (UK expert, LDBC GQL Formal Semantics WG)
Leonid Libkin (UK expert, LDBC GQL Formal Semantics WG)

Original WG3:W13-014 discussion paper, 21 June 2021
LDBC Technical Report TR-2021-01, October 2021

Contents

1. References

Reference	Title	Author	Date
[9075_8IWD14-16-PGQ]	“Information technology — Database language SQL — Part 16: Property Graph Queries (SQL/PGQ) Working Draft 14	Eds Stefan Plantikow, Stephen Cannan	4 June 2021
[WG3:W10-019r1]	“Aligning the definitions” r1	Stephen Cannan	8 April 2021
[WG3:W10-026r1]	“Definition of a path in SQL/PGQ and GQL”	Alastair Green	21 April 2021
[WG3:W10-027r1]	“Graph Terminology for GQL and SQL/PGQ: Mathematical definitions”	Leonid Libkin	21 April 2021
[WG3:W10-028]	“Input to discussion on path terminology”	Neo4j Query Languages Standards and Research Team	Uploaded 22 April 2021 [misdated 27 November 2020]
[WG3:W11-029]	“<property reference> proposal”	Fred Zemke	19 May 2021
[GraphHomomorphisms]	“Graphs and Homomorphisms”, Oxford Lecture Series in Mathematics and its Applications 28, OUP, pp 1-3, 4, 6]	Pavol Hell and Jaroslav Nešetřil	2004
[Digraphs]	“Digraphs: Theory, Applications and Algorithms”, Springer-Verlag, Berlin.	Jørgen Bang-Jensen and Gregory Gutin	2000 (1st edition), corrected 2007
[Graph Theory]	“Graph Theory”, Addison-Wesley, Reading, Mass.	Frank Harary	1969
[Structural Models]	“Structural models: an introduction to the theory of directed graphs”	Harary, Norman and Cartwright	1965
[Null Graph]	"Is the Null Graph a Pointless Concept?" In *Graphs and Combinatorics Conference, George Washington University.* New York: Springer-Verlag	Harary, F. and Read, R.	1973

[Empty Graph]	"Empty Graph." From *MathWorld*--A Wolfram Web Resource.	Weisstein, Eric W.	2002

Note on references

Key references for this area in terms of standard concepts and their definitions are [GraphHomomorphisms] and [Digraphs]. (The second edition [2009] of [Digraphs] is less useful for this particular area of mixed (partially directed) graphs.)

2. Introduction

This paper sets out to

1. Provide a formal mathematical definition of a “property graph” and “path in a property graph”, which we view as a *partially-oriented walk in a mixed multigraph*, a concept which is apparently not fully described in the existing literature of graph theory.
2. Provide definitions of associated concepts such as sub-path, path length, and restricted walks. We allow in these definitions for both 1-vertex (single vertex) walks, with a length of 0, that is consisting only of an isolated vertex), and 0-vertex (zero-vertex) walks, which contain no graph elements, and which can arise in operations that attempt to produce paths.
3. Propose corresponding textual definitions for inclusion in the SQL/PGQ and GQL specifications, and some commentary suitable for a concepts section in either of those documents.

3. Varying terminology in graph theory

Many fields have their own distinct cultures. In [Graph Theory] Frank Harary, who played a central role in the development of modern graph theory from the 1950s onwards[2], observed:

> “Most graph theorists use personalized terminology in their books, papers and lectures. In order to avoid quibbling at conferences on graph theory it has been found convenient to adopt the procedure that each man *[sic!]* state in advance the graph theoretic language he would use. Even the very word “graph” is not sacrosanct. Some authors actually define a “graph” as a graph,* *[Footnote: *This is most frequently done by the canonical initial sentence, “In this paper we only consider finite undirected graphs without loops or multiple edges.”]* but others intend such alternatives as multigraph, pseudograph, directed graph or network. We believe that uniformity in graphical terminology will never be attained, and is not necessarily desirable.”

[2] Harary was the major initial theorist of directed or digraphs, codifying his work in [Structural Models], published in 1965. The term “mixed graph” was first used in 1967 in a paper published by Harary.

As we are developing standards, we are compelled to define a uniform terminology, a goal which Stephen Cannan has pursued in [WG3:W10-019r1].

The tradition of multiple terminologies described by Harary is one obstacle, already familiar from the multiple terms used for nodes, points or vertices, and relationships, lines, arcs and edges.

4. Graph theoretical and textual definitions of a walk

For the GQL and SQL standards textual definitions are required. It is desirable that these rest on, or are equivalent to, mathematical definitions that conform as much as possible with standard set theoretical formalizations used in graph theory.

However, a reasonable textual definition may not align exactly with the most acceptable or conventional mathematical definition.

In this case we have an additional factor: although the idea of a mixed multigraph is well-known and studied in graph theory, it appears that the idea of a *partially-oriented path or walk* in such a graph is novel from the standpoint of the published academic literature. Therefore, its mathematical definition will have to extend or synthesize existing partial definitions.

This paper proposes two (equivalent) alternative mathematical approaches, and a set of textual definitions for potential inclusion in the SQL-PGQ (Part 16) working draft (and by editorial extension, to GQL).

We start with a summary of the role of paths in [9075_8IWD14-16-PGQ] and how they relate to graph theoretical concepts.

5. An "extracted path": a partially-oriented walk in a mixed multigraph

DIscussion specific to "graph pattern matching" in the draft SQL/PGQ and GQL standards, is highlighted in [bracketed blue text]*.*

[Path pattern matching in SQL/PGQ and GQL takes as an input a property graph and a <graph pattern> expression, and produces a set of "reduced" path bindings[3].

A *reduced path binding* is a sequence of pairs (***element variable* | *anonymous element symbol*, *graph element***). A graph element is, of course, either a vertex or an edge.

A sequence formed by replacing each pair in a reduced path binding with its graph element member alone is a sequence of graph elements, which after removing any duplicate vertices, is called an *extracted path*.

[3] See "Appendix: Projection of an extracted path" for a detailed summary of this process, which is defined in two key sections of [9075_8IWD14-16-PGQ]: "9.3 Machinery for graph pattern matching" and "10.2 <graph pattern>".

An extracted path corresponds to the graph theory concept of a walk ***W*** in a graph ***G***.]

A walk ***W*** in a graph ***G*** is a second graph, independent of ***G*** but defined with respect to ***G***, so a vertex in ***W*** or an edge in ***W*** must also be mapped to a vertex or edge in ***G***. This allows two elements in ***W*** to refer to the same element in ***G***.

[In our context of an attributed (property) graph in a database, if the identifier, labels, properties and property values of a node or edge in ***W*** are accessed by a client, then they must be those of the element it references in ***G***, *at the logical point in time when* ***G*** *is extracted* in a read query (SELECT from GRAPH_TABLE in SQL/PGQ). Two elements in a walk may have the same element identifier, therefore.]

A walk is a sequence of graph elements with a distinct structure. It can have no members (be a zero-vertex walk/graph); one member (which must be a vertex, making it a single-vertex walk/graph); or an odd number of members 3, 5, 7 ..., in which case the walk starts with a vertex, which is followed by a subsequence of half-edges (each an ordered pair (***edge***, ***vertex***)), and therefore ends with a vertex as well, making it a multi-vertex walk.

Property graphs are *multigraphs* (there can be multiple edges between two endpoint vertices) and pseudographs (there can be an edge looping from a vertex to itself). They are *mixed* (partially directed) *graphs*, as an edge can be *undirected*, or can have tail and head vertices, in which case it is *directed* from tail to head (or equivalently is *dominated* by the head vertex).

A walk ***W*** in a mixed graph ***M*** traverses vertices, (undirected) edges and arcs (directed edges). Such a walk therefore can be defined to contain information about the role that a vertex plays as an endpoint: whether it is neutral with respect to the direction of an edge, or acts as the head or tail of an arc (as directed edges are sometimes termed). This information gives us the *orientation* of an edge *with respect to the walk*, which (as a sequence) is ordered, and therefore has a first and last element. (A vertex can therefore act as the endpoint of zero, one or two edges in a walk).

An edge in ***W*** may be *unoriented* (if that edge is undirected in ***M***), or it may be *oriented forwards* if the edge is directed and its tail endpoint occurs before its head endpoint in the walk, or *backwards* if the reverse is true.

[The consumer of an extracted path should be able to find the identity and content of each graph element in the path viewed as a walk, its position in the path, and its orientation (if an edge) or role in the orientation of each edge for which it is an endpoint (if a vertex).

An extracted path in SQL/PGQ or GQL is therefore an example of] a walk ***W*** in ***G***, that is, a **partially-oriented walk in a property graph**, which is an attributed, mixed multi/pseudograph[4].

[4] In the academic literature we have found work on wholly-oriented walks in mixed multigraphs (described and referenced in [Digraphs], 8.9 "Orientations of Mixed Graphs"), but as the authors explain, for their purposes "when we speak of orienting a mixed (multi)graph this means that we assign an orientation to every edge and leave the original arcs unchanged". In our case we wish to leave the edges unoriented, and our orientation of the graph (walk) is therefore partial.

A walk where each edge in the path is also a distinct edge in the graph is called a trail; a walk where each vertex in the path is also a distinct vertex in the graph is called a simple path.

6. Two equivalent ways of defining a walk mathematically

There are (at least) two valid approaches to defining a walk in mathematical terms.

To start, we define our notation for sequences and subsequences.

6.1. Sequences and subsequences

A *sequence* $\boldsymbol{s}$ of length $n \in \mathbb{N}_0$ over a set $\boldsymbol{A}$ is a function $\boldsymbol{s} : \{i \in \mathbb{N}_1 \mid 1 \leq i \leq n\} \longrightarrow \boldsymbol{A}$, often written by listing its elements explicitly, e.g., $(s_1, ..., s_n)$. Given $\boldsymbol{s}$, by $\boldsymbol{s}[i]$ we denote its i-th element (in the previous example, s_i). If $n = 0$, this is the *empty* sequence.

Given a sequence $\boldsymbol{s}$ of length n, and bounds b and e (beginning and end) with $1 \leq b \leq e \leq n$, the *subsequence* $\boldsymbol{s'}[b...e]$ of $\boldsymbol{s}$ is the sequence $\boldsymbol{s'} = (s[b], ..., s[e])$, that is, $\boldsymbol{s'}[1] = \boldsymbol{s}[b]$, $\boldsymbol{s'}[2] = \boldsymbol{s}[b+1]$, …, $\boldsymbol{s'}[e-b+1] = \boldsymbol{s}[e]$. The empty sequence is a subsequence of every sequence.

For example, in a sequence $\boldsymbol{ab} = (a_1, b_1, a_2, b_2, a_3)$ we have $\boldsymbol{ab}[1] = a_1$, $\boldsymbol{ab}[2] = b_1$, $\boldsymbol{ab}[3] = a_2$, $\boldsymbol{ab}[4] = b_2$, $\boldsymbol{ab}[5] = a_3$. The subsequence $\boldsymbol{ab'} = \boldsymbol{ab}[2...4]$ is (b_1, a_2, b_2). So, in this sequence, $\boldsymbol{ab'}[1] = \boldsymbol{ab}[2] = b_1$, $\boldsymbol{ab'}[2] = \boldsymbol{s}[3] = a_2$ and $\boldsymbol{ab'}[3] = \boldsymbol{s}[4] = b_2$.

6.2. Approach I. A sequence alternating between nodes and edges

Approach I rests on the concept of a sequence of graph elements alternating between nodes and edges. Cannan, [WG3:W10-019r1], Libkin, [WG3:W10-027r1], Neo4j [WG3:W10-028]. This approach is used in e.g. [Digraphs] by Bang-Jensen and Gutin.

In [9075_8IWD14-16-PGQ] this is formulated textually in "9.3 Machinery for graph pattern matching", p28 as:

> "... either a zero-length string or it is a path of *PG* [the pure property graph]; it begins with a vertex, it alternates between vertices and edges, each edge connects the vertex before and after it in the extracted path, and it ends with a vertex."

(although this treats an empty sequence [one with no elements at all] as not being a path).

[WG3:W10-027r1] rendered this approach formally, and we have expanded the definitions given there, as follows.

A property graph is a tuple $\boldsymbol{G} = \langle \boldsymbol{N}, \boldsymbol{E}_\mathrm{d}, \boldsymbol{E}_\mathrm{u}, \lambda, \textit{ENDPOINTS}, \textit{SRC}, \textit{TGT}, \textit{DV} \rangle$ where

- $\boldsymbol{N}$ is a set of node ids used in G, coming from a set $\mathcal{N}$ of node ids;
- $\boldsymbol{E}_\mathrm{d}$ is a set of directed edge ids used in $\boldsymbol{G}$, coming from a set $\mathcal{E}_\mathrm{d}$ of directed edge ids;
- $\boldsymbol{E}_\mathrm{u}$ is a set of undirected edge ids used in $\boldsymbol{G}$, coming from a set $\mathcal{E}_\mathrm{u}$ of undirected edge ids;

- λ is a labeling function; λ : $\boldsymbol{N} \cup \boldsymbol{E}_{\mathrm{d}} \cup \boldsymbol{E}_{\mathrm{u}} \longrightarrow 2^{\mathcal{L}}$ associates with each of the ids a set (possibly empty) of labels from the set $\mathcal{L}$ of labels[5];
- *ENDPOINTS*: $\boldsymbol{E}_{\mathrm{u}} \longrightarrow 2^{N}$ associates with each edge is $\boldsymbol{e} \in \boldsymbol{E}_{\mathrm{u}}$ a set of its endpoints such that the cardinality of *endpoints*($\boldsymbol{e}$) is 1 or 2;
- *SRC*, *TGT*: $\boldsymbol{E}_{\mathrm{d}} \longrightarrow \boldsymbol{N}$ associate with each directed edge id its source and target;
- *DV*: $\boldsymbol{N} \cup \boldsymbol{E}_{\mathrm{d}} \cup \boldsymbol{E}_{\mathrm{u}} \longrightarrow \mathcal{V}$ associates with each id of a node or an edge some data value from a set $\mathcal{V}$; normally this would be a set of key-value pairs but for definitions of paths it is not important what this set is;
- all sets $\mathcal{N}$, $\mathcal{E}_{\mathrm{d}}$, $\mathcal{E}_{\mathrm{u}}$, $\mathcal{L}$, $\mathcal{V}$ are disjoint.

Notes:

A graph itself can have data values associated with it. This can be modelled by adding an element of $\mathcal{V}$ to the tuple defining $\boldsymbol{G}$. For the purposes of defining paths this makes no difference.

Other possible graph theory terminology for undirected edges: just *edge,* or *line*.
Other possible graph theory terminology for directed edges: *arc*, or *link*.

Two nodes $\boldsymbol{v}$ and $\boldsymbol{v'}$ are *adjacent* if either endpoints(e) = $\{\boldsymbol{v}, \boldsymbol{v'}\}$ for some $\boldsymbol{e} \in \boldsymbol{E}_{\mathrm{u}}$, or $\{\boldsymbol{v}, V'\} = \{$*SRC*$(\boldsymbol{e})$, *TGT*$(\boldsymbol{e})\}$ for some $\boldsymbol{e} \in \boldsymbol{E}_{\mathrm{d}}$. Spelling this out, to help with plain language definitions, means either $\boldsymbol{v}$ = *SRC*($\boldsymbol{e}$), $\boldsymbol{v'}$ = *TGT*(e) or $\boldsymbol{v'}$ = *SRC*(e), $\boldsymbol{v}$ = *TGT*($\boldsymbol{e}$) for some $\boldsymbol{e} \in \boldsymbol{E}_{\mathrm{d}}$.

Definition I.1 (Path) A path in $\boldsymbol{G}$ is a sequence $\boldsymbol{\pi} = (\boldsymbol{v}_1, \boldsymbol{e}_1, \boldsymbol{v}_2, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_{n-1}, \boldsymbol{v}_n)$ where:

1. each $\boldsymbol{v}_i$ is a node id from $\boldsymbol{N}$ and each $\boldsymbol{e}_j$ is an edge id from $\boldsymbol{E}_d \cup \boldsymbol{E}_u$*;*

2. $n \geq 0$*;*

3. for every j between 1 and $n-1$*,* one of the following is true*:*

- $\boldsymbol{e}_j \in \boldsymbol{E}_{\mathrm{u}}$ and *ENDPOINTS*$(\boldsymbol{e}_j) = \{\boldsymbol{v}_j, \boldsymbol{v}_{j+1}\}$ (an *undirected* edge*);*
- $\boldsymbol{e}_j \in \boldsymbol{E}_{\mathrm{d}}$ and *SRC*$(\boldsymbol{e}_j) = \boldsymbol{v}_j$ and *TGT*$(\boldsymbol{e}_j) = \boldsymbol{v}_{j+1}$ (a *forward edge* or *forward arc*)*;*
- $\boldsymbol{e}_j \in \boldsymbol{E}_{\mathrm{d}}$ and *TGT*$(\boldsymbol{e}_j) = \boldsymbol{v}_j$ and *SRC*$(\boldsymbol{e}_j) = \boldsymbol{v}_{j+1}$ (a *backward edge* or *backward arc*).

Note: This definition allows for a walk consisting of an isolated, single vertex (an *empty path*) or a *null path* (one with no vertices), by relying on the fact that if the number of vertices $n = 0 \mid 1$, then there is no positive integer value j between 1 and $n-1 = 0 \mid -1$, and therefore there is no member of the set of edges in the sequence.

We now define the length of a path, a sub-path, and associated concepts:

Definition I.2 (Length of a path)

The length of a path is $n-1$.

Definition I.3 (Zero-edge or single-vertex path: n=1)

[5] Various adjustments are possible, e.g. by saying $|\lambda(\boldsymbol{e})| = 1$for $\boldsymbol{e} \in \boldsymbol{E}_{\mathrm{d}} \cup \boldsymbol{E}_{\mathrm{u}}$ to force single edge labels etc.

A path is zero-edge, or equivalently single-vertex, if n=1; such a path has a length of 0.

Definition I.4 (Zero-vertex path: n=0)

A path is zero-vertex if n=0; such a path has a length of −1.

Definition I.5 (Sub-path)

Given a path $\boldsymbol{\pi} = (\boldsymbol{v}_1, \boldsymbol{e}_1, \boldsymbol{v}_2, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_{n-1}, \boldsymbol{v}_n)$, its subpaths are all subsequences of $\boldsymbol{\pi}$ which are themselves paths*,* i.e., the empty sequence or $(\boldsymbol{v}_i, \boldsymbol{e}_i, \ldots, \boldsymbol{e}_{j-1}, \boldsymbol{v}_j)$ whenever $1 \leq i \leq j \leq n$.

To spell this out, $(\boldsymbol{v}_i, \boldsymbol{e}_i, \ldots, \boldsymbol{e}_{j-1}, \boldsymbol{v}_j) = (\boldsymbol{\pi}[2i-1], \boldsymbol{\pi}[2i], \boldsymbol{\pi}[2i+1], \ldots, \boldsymbol{\pi}[2j-1])$.

Thus, a single node (empty path) is a subpath (when $i = j$) and the null path is always a subpath corresponding to the empty subsequence.

Definitions I.6 (Special types of paths)

A path $\boldsymbol{\pi} = (\boldsymbol{v}_1, \boldsymbol{e}_1, \boldsymbol{v}_2, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_{n-1}, \boldsymbol{v}_n)$ is **simple** if no node repeats except perhaps the first and the last one. That is, if $\{i, j\}$ is not equal to $\{1, n\}$ and i and j are different, then $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ are different.

A path is **acyclic** if it is simple and $\boldsymbol{v}_1$ and $\boldsymbol{v}_n$ are different.

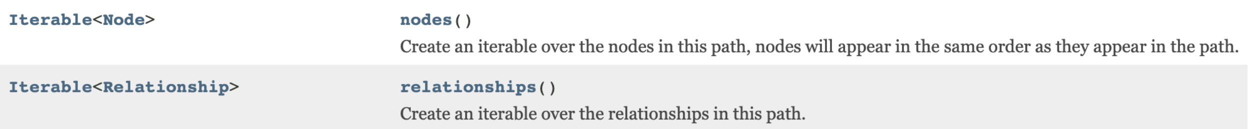
A path is a **trail** if $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$ are different whenever i and j are different.

A path is **closed** if $\boldsymbol{v}_1 = \boldsymbol{v}_{\mathrm{n}}$.

6.3. Approach II. A sequence of nodes, with a corresponding sequence of edges

The second approach defines a walk as a sequence of 0, 1 or more vertices, where each consecutive pair in a sequence of 2 or more vertices maps to an edge in a corresponding sequence of 1 or more edges. The sequence of edges corresponding to a vertex sequence of length one, or nought, is empty.

The interface for Neo4j's client driver libraries for paths shows this approach:



`Iterable<Node>`	`nodes()` Create an iterable over the nodes in this path, nodes will appear in the same order as they appear in the path.
`Iterable<Relationship>`	`relationships()` Create an iterable over the relationships in this path.

First we define a graph using the conventional terminology often seen when treating directed graphs, where undirected lines are edges, and directed ones are arcs. This definition also removes graph element identifiers, adopting an alternative approach of defining multisets of edges and arcs (taken together, a multiset of relationships).

A **property graph** is a partially directed mixed pseudo-multigraph $\boldsymbol{M} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{A})$, where

$\boldsymbol{V}$ is a set of vertices;

$\boldsymbol{E}$ is a multiset of edges where each $\boldsymbol{e} \in \boldsymbol{E}$ is an unordered pair of endpoints $\{\boldsymbol{u}, \boldsymbol{v}\} \subseteq \boldsymbol{V}$;

$\boldsymbol{A}$ is a multiset of arcs, $\boldsymbol{V} \times \boldsymbol{V}$. Each $\boldsymbol{a} \in \boldsymbol{A} = (\boldsymbol{u}, \boldsymbol{v})$ is an ordered pair of endpoints, where the arc is directed from $\boldsymbol{u}$ (its tail) to $\boldsymbol{v}$ (its head);

We term $\boldsymbol{E} \cup \boldsymbol{A}$ as the multiset of relationships $\boldsymbol{R}$: for any pair of endpoints $\{\boldsymbol{u}, \boldsymbol{v}\}$ there may be many parallel edges and parallel arcs (which may be of opposing directions);

$\boldsymbol{M}$ and each member of $\boldsymbol{V} \cup \boldsymbol{R}$ is mapped to a set of attributes (a disjoint union of a set of labels and properties).

A **partially-oriented walk** $\boldsymbol{W}(n, m)$ in $\boldsymbol{M}$ is a pair of sequences $\langle \boldsymbol{v}, \boldsymbol{r} \rangle$ where:

$n \geq 0$ and $m = max(n-1, 0)$;

$\boldsymbol{v} = (\boldsymbol{v}_i)_{1 \leq i \leq n}$ is a sequence of n vertices from $\boldsymbol{V}$;

$\boldsymbol{r} = (\boldsymbol{r}_i)_{1 \leq i \leq m}$ is a sequence of m relationships from $\boldsymbol{R}$ such that each $\boldsymbol{r}_i$ is one of the following:

1. an edge $\boldsymbol{e} \in \boldsymbol{E}$ whose endpoints are $\{\boldsymbol{v}_i, \boldsymbol{v}_{i+1}\}$
2. a forward arc $\boldsymbol{f} \in \boldsymbol{A}$ whose tail is $\boldsymbol{v}_{i+1}$, and whose head is $\boldsymbol{v}_i$
3. a backward arc $\boldsymbol{b} \in \boldsymbol{A}$ whose tail is $\boldsymbol{v}_i$, and whose head is $\boldsymbol{v}_{i+1}$

Note: as a walk is a graph (separate from the graph $\boldsymbol{M}$ from which it is derived) we can use the conventional notation $\boldsymbol{W}(n, m)$ to express the order (number of vertices) and size (number of edges) in this graph. This allows us to easily note the difference between a 0-vertex walk and a 1-vertex (equivalently, a 0-edge) walk, both of which have a size of 0.

This approach is suggested by the view of a path as centring on a sequence of vertices, used in e.g. [GraphHomomorphisms] by Hell and Nešetřil, and in Green [WG3:W10-026r1].

We now introduce consequential definitions:

Definition II.2 (Length and size of a walk)

The length of a walk $\boldsymbol{W}(n, m)$ is $n-1$, while its size is the number of its edges m.[6]

Definition II.3 (Single-vertex, or zero-edge, walk)

If $n = 1$ then $m = 0$: a walk $\boldsymbol{W}(1, 0)$ is an single-vertex walk (with length 0), containing only an isolated vertex.

Definition II.4 (Zero-vertex walk)

If $n = 0$ then $m = 0$: a walk $\boldsymbol{W}_{\varnothing} = \boldsymbol{W}(0, 0)$ is a zero-vertex walk (with length -1).

Definition II.5 (Subwalk)

A subwalk of a walk $\boldsymbol{W}(n, m) = \langle \boldsymbol{v}, \boldsymbol{r} \rangle$ is either a zero-vertex walk or the walk $\langle \boldsymbol{v'}, \boldsymbol{r'} \rangle$ where, for bounds b and e with $1 \leq b \leq e \leq n$, the sequence $\boldsymbol{v'}$ is $\boldsymbol{v}[b...e]$ and the sequence $\boldsymbol{r'}$ is $\boldsymbol{r}[b...e-1]$.

Definitions II.6 (Special types of walks)

Recall that sequences $\boldsymbol{v}$ and $\boldsymbol{r}$ are functions $\boldsymbol{v} : \{1, \ldots, n\} \longrightarrow \boldsymbol{V}$ and $\boldsymbol{r} : \{1, \ldots, m\} \longrightarrow \boldsymbol{R}$.

[6] Graphs conventionally have sizes, and paths have lengths (both being the number of edges). In our context, where we may have no edges and no vertices in a zero-vertex (sub-)path it is useful to distinguish the size of a path *qua* graph, from its length *qua* path.

A walk is **acyclic** if $\boldsymbol{v}$ is injective.

A walk is **simple** if the restrictions of $\boldsymbol{v}$ to both of $\{1, \ldots, n{-}1\}$ and $\{2, \ldots, n\}$ are injective functions.

A walk is **closed** if $\boldsymbol{v}(1) = \boldsymbol{v}(n)$ (which is equivalent to $\boldsymbol{v}[1] = \boldsymbol{v}[\mathrm{n}]$).

A walk is a **trail** if the function $\boldsymbol{r}$ is injective.

The diagrams in “Appendix: Diagrams” illustrate a walk, a subwalk and these special walks.

6.4. Translating between Approach I and Approach II

Both of these formal definitional approaches describe the same class of mathematical object.

If we have a path

$$\boldsymbol{\pi} = (\boldsymbol{v}_1, \boldsymbol{e}_1, \boldsymbol{v}_2, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_{n-1}, \boldsymbol{v}_n)$$

then the equivalent partially-oriented walk is

$\mathbf{W}(n, m)_{\boldsymbol{\pi}} = \langle \boldsymbol{v}, \boldsymbol{r} \rangle$ where

$m = max(n{-}1, 0)$

$\boldsymbol{v} = (\boldsymbol{\pi}[1], \boldsymbol{\pi}[3], \ldots, \boldsymbol{\pi}[2n{-}1])$

$\boldsymbol{r} = (\boldsymbol{\pi}[2], \boldsymbol{\pi}[4], \ldots, \boldsymbol{\pi}[2n])$.

Given a walk

$$\boldsymbol{W}(n, m)$$

the corresponding path

$$\boldsymbol{\pi}_{W(m,\, n)} = (\boldsymbol{v}[1], \boldsymbol{r}[1], \boldsymbol{v}[2], \boldsymbol{r}[2], \ldots, \boldsymbol{r}[n{-}1], \boldsymbol{v}[n]).$$

The following sections propose textual definitions and descriptions drawing on both of these approaches. We start by proposing some wording on concepts.

7. Proposed concepts section for the SQL/PGQ and GQL specifications

The following text could be included in a concepts section of both the SQL/PGQ and GQL specifications.

“A mathematical definition of a *mixed (partially directed)* graph is given in “Digraphs: Theory, Applications and Algorithms”, Jørgen Bang-Jensen and Gregory Gutin. 2007, Springer-Verlag, Berlin. Section “8.9 Orientations of Mixed Graphs”. In summary:

> Let ***M*** = (***V***, ***A***, ***E***) be a mixed graph. Let ***G*** = (***V***, ***E***) be the undirected part and let ***D*** = (***V***, ***A***) be the directed part of ***M***.

The sets of vertex pairs that define the (undirected) edges in ***E*** and the (directed) arcs in ***A*** can overlap, so at most there can be an edge and two arcs between any two vertices in ***V***.

The authors add:

> A mixed multigraph is the same as a mixed graph, except that we allow parallel arcs and parallel edges as well as arcs that are parallel to edges.

A mathematical definition of an oriented walk in a *directed* graph is given in the "Introduction" to Hell, Pavol, and Jaroslav, Nešetřil. 2004. *Graphs and Homomorphisms*. OUP Oxford, pp 4, 6.[7]

> ... an oriented walk in a digraph G is a sequence of vertices v_0, v_1, ..., v_k of G such that v_{i-1} and v_i are adjacent in G, for each i = 1, 2, ... , k. An arc $v_{i-1}\, v_i \in E(G)$ is called a forward arc of the walk, and an arc $v_i\, v_{i-1}$ is called a backward arc of the walk.

In a database property graph a walk in a *mixed* (partially directed) *multigraph* may also a) traverse parallel arcs and edges, and b) traverse an edge which is undirected: the walk is therefore only *partially oriented*.

In this standard the unqualified term "path" is used as a convenient and familiar synonym for a ***partially-oriented walk in a mixed multigraph*** which manifests itself, for example, as an extracted path resulting from a graph pattern match.

The fact that an edge may be a self-loop (that the graph is a pseudograph) does not affect the concept of a path or walk."

8. Proposed informal definitions relating to "path" in SQL/PGQ and GQL

Definitions 3, 4, 5 and the second sentence of 6 paraphrase or relate to terms specific to "graph pattern matching" in the draft SQL/PGQ and GQL standards, and are highlighted in [bracketed blue text].

1. **Property graph**

   A graph which is

   a. *attributed*: the graph, and each of its vertices and edges, has a (possibly empty) set of attributes, being the disjoint union of a set of labels and a set of properties

   b. *partially directed* (mixed), so each edge may be directed from a source vertex to a destination vertex, or may be undirected (neither of its endpoints is a source or a destination)

   c. a *multigraph*, allowing multiple edges between two vertices

   d. a *pseudograph*, allowing loops, which are edges which join a vertex to itself.

2. **Partially-oriented walk**

   A walk (as commonly understood in graph theory) in a property graph, whose edges are either unoriented—if an edge has a corresponding edge in the graph which is undirected—or oriented, when an edge reflects the direction of a corresponding directed edge in the graph. The orientation of an oriented edge is defined with respect to the order of the walk, which is a sequence from an initial vertex to a final vertex, if it is not an empty sequence.

   See also **Path,** <Path>**Unoriented edge,** <Path>**Oriented Edge,** <Path>**Null path.**

3. **[Path binding**

[7] https://paperpile.com/shared/TqO0j6

A possibly empty sequence resulting from matching a graph pattern in a property graph, whose terms are each a pair, of the following kinds:

1. A bracket pair, whose first and second members are both either an opening or a closing bracket symbol
2. A subpath delimiter pair, whose first and second members are both the same start or end subpath symbol, unique in the path binding, allowing a subpath to be delimited
3. A vertex pair, whose first member is a vertex variable or an anonymous vertex symbol and whose second member is a vertex in the property graph
4. An edge pair, whose first member is an edge variable or an anonymous edge symbol,, and whose second member is an edge in the property graph

and whose first and last terms must both be vertex pairs.]

4. [<Path binding>**Annotated path**

   A sequence of the second members of the pairs in a binding path, in the same order as the binding path sequence.]

5. [<Annotated path>**Extracted path**

   An extracted path is a subsequence of an annotated path produced by working from beginning to end, removing bracket symbols, subpath symbols and consecutive (duplicate) vertices. Each remaining edge is preceded by a vertex and followed by a vertex, which are its endpoints in the property graph from which the path was extracted. An extracted path may contain only one vertex or no vertices, in which cases it contains no edges. The edges of an extracted path are either oriented or unoriented.]

6. **Path**

   A path is a partially-oriented walk in a property graph.[A path is a sequence with the structure and characteristics of an extracted path.][8]

7. <Path>**Associated property graph**

   The graph with respect to which a walk is defined is the associated property graph of that walk.

8. <Path>**Oriented edge**

   An edge in a path, which is a directed edge in its associated property graph, is *forward-oriented* if the vertex to its left in the path is its source endpoint, or *backward-oriented* if the vertex to its left is its destination endpoint.

9. <Path>**Unoriented edge**

   An *unoriented* edge in a path is an undirected edge in its associated property graph.

[8] [It is possible that a path may be created by other means than pattern matching, for example in the course of processing a GQL INSERT or DELETE statement. A statement making this explicit might be appropriate as a note in the GQL specification.]

10. <Path>**Length**

    The length of a path is one less than the number of its vertices. If a path contains edges then its length is the number of its edges.

11. <Path>**Zero-vertex path**

    A path with no vertices (and whose length is therefore −1).

12. <Path>**One-vertex path**

    A path with one vertex (and whose length is therefore 0).

13. <Path>**Closed**

    A path is closed (synonymously, is a cycle) if its first and last members are the same vertex in its associated property graph.

14. <Path>**Subpath**

    A subpath is a path which is a contiguous subsequence of another path, and may be an empty sequence, i.e a zero-vertex path.

15. <Path>**Walk**

    The unqualified term walk is a synonym for path.

16. <Path>**Trail**

    A walk such that no two of its edges are the same edge in its associated property graph.

17. <Path>**Simple path**

    A walk such that only the first and last terms may be the same vertex in its associated property graph.

18. <Path>**Acyclic**

    A path is acyclic if no two of its vertices are the same vertex in its associated property graph. It is therefore a simple path.

9. Appendix: On evolution of terminology

The original version of this paper was discussed in meetings of the U.S. standards committee INCITS DM32 Ad-Hoc for SQL/PGQ (Property Graph Query), and of the international standards committee ISO/IEC JTC1 SC32/WG3, in July 2021. It was pointed out that the term “null” has a very strong connotation in SQL of a null value (of a cell in a table or of an expression), and that the term “null walk” could be confusing.

On reflection, we can also see that walks with no content, and walks with one vertex but no edges could both be described as “empty”, if by that we mean edgeless. To make things worse, there is a standing ambiguity in the academic literature on the meaning of “null graph”, which is reflected in this entry in the Wolfram MathWorld online dictionary of mathematics:[Empty Graph].

There is an old debate, summarized in [Null Graph] about the validity of the concept of a graph (of which a walk is a special case) with no vertices.

In the context of path pattern matching in a graph, a pattern may yield no matches (producing a graph with no elements). If a path pattern contains a sub-path pattern, and the sub-path pattern has no matches, then the path pattern as a whole may still produce matches. In this context we do need to define a matching operation that can yield a walk that is completely void (has no elements). For this reason we see value in defining special walks with no vertices and with one vertex.

Therefore we have decided to avoid the terms “empty walk” and “null walk” altogether, in favour of the terms "zero-vertex walk" and "single-vertex walk", which are unambiguous and self-descriptive.

Viewed as amendments to our original definitions (using Approach II, Section x.3), where we define

> A **partially-oriented walk** $\boldsymbol{W}(n, m)$ in $\boldsymbol{M}$ ...

we see the following changes to supplementary definitions:

> **Definition (1) (~~Empty walk~~ Single-vertex or zero-edge walk)**
>
> If $n = 1$ then $m = 0$: a walk $\boldsymbol{W}(1, 0)$ is a single-vertex ~~empty~~ walk (with length 0), containing only an isolated vertex.
>
> **Definition (2) (~~Null walk~~ Zero-vertex walk)**
>
> If $n = 0$ then $m = 0$: a walk $\boldsymbol{W}_{\varnothing} = \boldsymbol{W}(0, 0)$ is a zero-vertex ~~null~~ walk (with length −1).

We could then further say:

> **Definition (3) (Multi-vertex walk)**
>
> If $n > 1$ then $m = n - 1$, and a walk $\boldsymbol{W}(n, m)$ is a multi-vertex walk (with length m).

We can equivalently talk about 1-vertex, 0-vertex, 2-vertex (edge) and M-vertex walks.

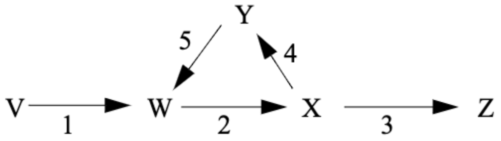
10. Appendix: Projection of an extracted path

This appendix describes the process of "graph pattern matching" in the draft SQL/PGQ and GQL standards. The concepts and definitions in this report are stated independently of this motivating use-case of "path extraction".

An extracted path is produced (logically) by the application of a sequence of procedures operating on a **pure property graph** and a <path pattern list> from a **<graph pattern>**.

These steps are described in two key sections of [9075_8IWD14-16-PGQ]: "9.3 Machinery for graph pattern matching" and "10.2 <graph pattern>"

On p. 36 of "<property reference> proposal" [WG3:W11-029], Fred Zemke gives a useful example which helps to illustrate the projection of an extracted path. It is repeated here, with emphasis added in a couple of places:

> Consider the pattern
>
> ```
> [(A) -> [(B) -[E]-> (C) WHERE E.X = A.X+D.X]+ -> (D)]{2}
> ```
>
> to be evaluated in this graph:
>
> 
>
>
> I will use $[_1\ ]_1$ as the bracket symbols for the outer <parenthesized path pattern expression>, and $[_2\ ]_2$ for the inner group.
>
> The candidate solution I consider is V→ W→ X → Y→ W→ X→ Z, with the following **path binding**:
>
> $[_1$ A - $[_2$ B E C $]_2$ - D $]_1$ $[_1$ A - $[_2$ B E C $]_2$ - D $]_1$
>
> **$[_1$ V 1 $[_2$ W 2 X $]_2$ 4 Y $]_1$ $[_1$ Y 5 $[_2$ W 2 X $]_2$ 3 Z $]_1$**

The first sequence shown is the **word** of the path binding; the second is its **annotated path**.

A path binding arises when an implementation finds paths in the pure property graph that match a path pattern. There is a list of path patterns in a graph pattern: each path pattern has a corresponding path binding.

This structure binds element variables in the pattern to graph elements; it also introduces anonymous element symbols associated with graph elements that are not bound by an element variable, and indexed brackets to denote the nesting level and inclusion of bracketed expressions.

A path pattern can contain a sub-path variable, which generates a matched pair of subpath start and end delimiters with a unique identity corresponding to the variable, which surround a subword of the word, and a corresponding contiguous subsequence of the annotated path. Subpaths are not illustrated in the example.

A path binding can also be defined, therefore as a sequence of pairs: in the example:

$(([_1, [_1), (A, V), (-, 1), ..., (]_1,]_1))$

Brackets are paired with brackets; element variables like A are paired with a vertex or an edge, and anonymous element symbols () and – are paired with vertices and edges respectively. Subpath start and end identifiers are paired with themselves, in the manner of brackets.

An annotated path is the second sequence shown in the example,

$[_1$ V 1 $[_2$ W 2 X $]_2$ 4 Y $]_1$ $[_1$ Y 5 $[_2$ W 2 X $]_2$ 3 Z $]_1$

that is, a sequence of the second element of each pair in the sequence previously shown, in the same order:

$([_1$, V, 1, ..., $]_1)$

An **extracted path** is an annotated path, with all brackets and sub-path delimiters removed:

After step 1: V 1 W 2 X 4 Y Y 5 W 2 X 3 Z

and all consecutive vertices replaced by a single vertex.

After step 2: V 1 W 2 X 4 Y 5 W 2 X 3 Z

An extracted path conforms to the rule 9.3 (14), that is, it is

> ... either a zero-length string or it is a path of *PG* [the pure property graph]; that is, it begins with a vertex, it alternates between vertices and edges, each edge connects the vertex before and after it in the extracted path, and it ends with a vertex.

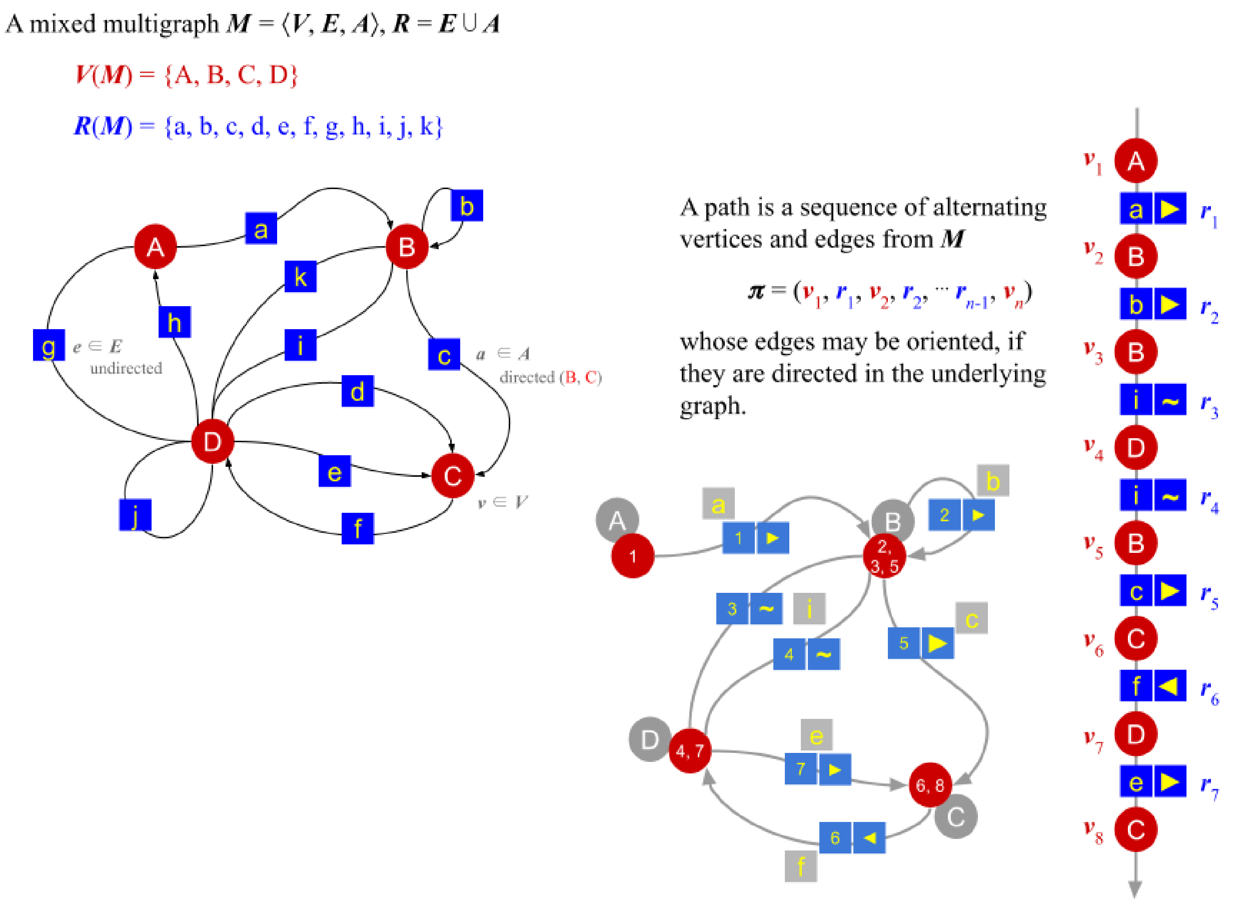
The function *REDUCE* defined in 9.3 removes brackets and sub-path delimiters from a path binding.

A “reduced path binding” is therefore a sequence of pairs, and a sequence of the second elements in those pairs appears to be equivalent to an annotated path, after the application of step 1 above, or to an extracted path, before the application of step 2. It may be that closer reading of [9075_8IWD14-16-PGQ] will show that this apparent anomaly is our misunderstanding. This has no bearing on the definition of an extracted path, which can be obtained by applying steps 1 and 2 to an annotated path, or step 2 to a sequence of the second elements of the pairs of a reduced path binding.

11. Appendix: Diagrams

11.1. Walks using Approach I

This diagram uses the definition of a mixed multigraph for Approach II, to illustrate the definition of a path using Approach I.



A mixed multigraph $\boldsymbol{M} = \langle \boldsymbol{V}, \boldsymbol{E}, \boldsymbol{A} \rangle$, $\boldsymbol{R} = \boldsymbol{E} \cup \boldsymbol{A}$

$\boldsymbol{V}(\boldsymbol{M}) = \{\mathrm{A, B, C, D}\}$

$\boldsymbol{R}(\boldsymbol{M}) = \{\mathrm{a, b, c, d, e, f, g, h, i, j, k}\}$

A path is a sequence of alternating vertices and edges from $\boldsymbol{M}$

$$\boldsymbol{\pi} = (\boldsymbol{v}_1, \boldsymbol{r}_1, \boldsymbol{v}_2, \boldsymbol{r}_2, \cdots \boldsymbol{r}_{n\text{-}1}, \boldsymbol{v}_n)$$

whose edges may be oriented, if they are directed in the underlying graph.

11.2. The same walk using Approach II

This diagram shows the equivalent definition of a walk in $\boldsymbol{M}$ using Approach II.

A mixed multigraph $\boldsymbol{M} = \langle V, \boldsymbol{E}, \boldsymbol{A}\rangle$, $\boldsymbol{R} = \boldsymbol{E} \cup \boldsymbol{A}$

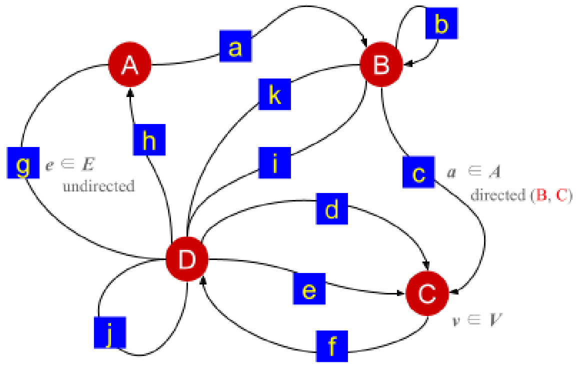
$V(\boldsymbol{M}) = \{\text{A, B, C, D}\}$

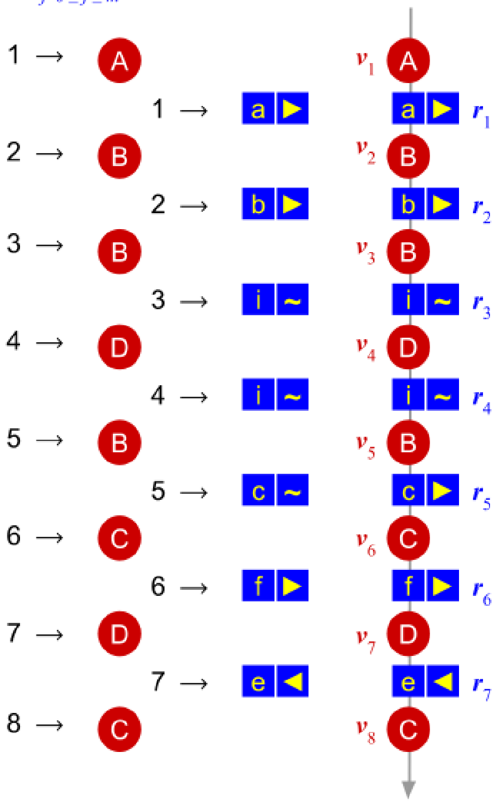
$\boldsymbol{R}(\boldsymbol{M}) = \{\text{a, b, c, d, e, f, g, h, i, j, k}\}$

A walk $\boldsymbol{W}(n, m)$ in $\boldsymbol{M} = \langle (\boldsymbol{v}_i)_{0 \le i \le n}, (\boldsymbol{r}_j)_{0 \le j \le m} \rangle$

$(\boldsymbol{v}_i)_{0 \le i \le n} : \mathbb{N} \rightarrow V(\boldsymbol{M})$

$(\boldsymbol{r}_j)_{0 \le j \le m} : \mathbb{N} \rightarrow \boldsymbol{R}(\boldsymbol{M})$





$\boldsymbol{W}(8, 7)$ in $\boldsymbol{M} = \langle \boldsymbol{v} = (\boldsymbol{v}_n)_{n=8}, \boldsymbol{r} = (\boldsymbol{r}_m)_{m=n-1} \rangle$

$\boldsymbol{v} = (\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4, \boldsymbol{v}_5, \boldsymbol{v}_6, \boldsymbol{v}_7, \boldsymbol{v}_8) = (\text{A, B, B, D, B, C, D, C})$

$\boldsymbol{r} = (\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3, \boldsymbol{r}_4, \boldsymbol{r}_5, \boldsymbol{r}_6, \boldsymbol{r}_7) = (\text{a, b, i, i, c, f, e})$

There may be no relationships in a walk: m does not always equal $n - 1$.

$m = max(n - 1, 0)$.
$\boldsymbol{W}(0, 0) = \boldsymbol{W}_\varnothing$ is a *zero-vertex walk.*
$\boldsymbol{W}(1, 0)$ is a *one-vertex walk.*

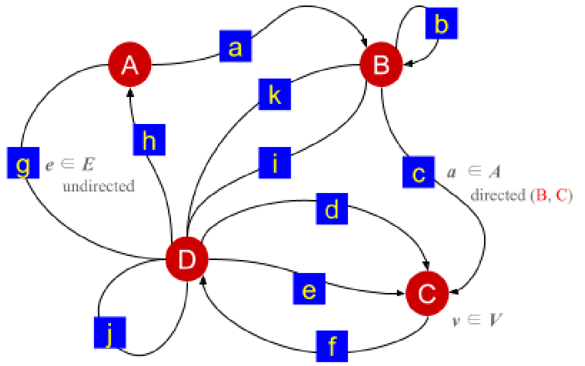
11.3. Subwalk of the same walk

This diagram shows a subwalk (which is a walk) of $W(n, m)$ in $\boldsymbol{M}$.

A mixed multigraph $\boldsymbol{M} = \langle \boldsymbol{V}, \boldsymbol{E}, \boldsymbol{A} \rangle$, $\boldsymbol{R} = \boldsymbol{E} \cup \boldsymbol{A}$

$\boldsymbol{V}(\boldsymbol{M}) = \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}\}$

$\boldsymbol{R}(\boldsymbol{M}) = \{\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}, \mathrm{e}, \mathrm{f}, \mathrm{g}, \mathrm{h}, \mathrm{i}, \mathrm{j}, \mathrm{k}\}$



For a walk $\boldsymbol{w} = \boldsymbol{W}(8, 7)$ in $\boldsymbol{M}$

$\boldsymbol{w} = \langle \boldsymbol{v} = (\boldsymbol{v}_n)_{n=8}, \boldsymbol{r} = (\boldsymbol{r}_m)_{m=n-1} \rangle$

$\boldsymbol{v} = (\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4, \boldsymbol{v}_5, \boldsymbol{v}_6, \boldsymbol{v}_7, \boldsymbol{v}_8) = (\mathrm{A}, \mathrm{B}, \mathrm{B}, \mathrm{D}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{C})$

$\boldsymbol{r} = (\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3, \boldsymbol{r}_4, \boldsymbol{r}_5 , \boldsymbol{r}_6, \boldsymbol{r}_7) = (\mathrm{a}, \mathrm{b}, \mathrm{i}, \mathrm{i}, \mathrm{c}, \mathrm{f}, \mathrm{e})$

the subwalk

$\boldsymbol{sw}_w = \boldsymbol{w}[2 \ldots 4] = \langle (\boldsymbol{v}' = \boldsymbol{v}[2 \ldots 4], \boldsymbol{r}' = \boldsymbol{r}[2 \ldots 3] \rangle$

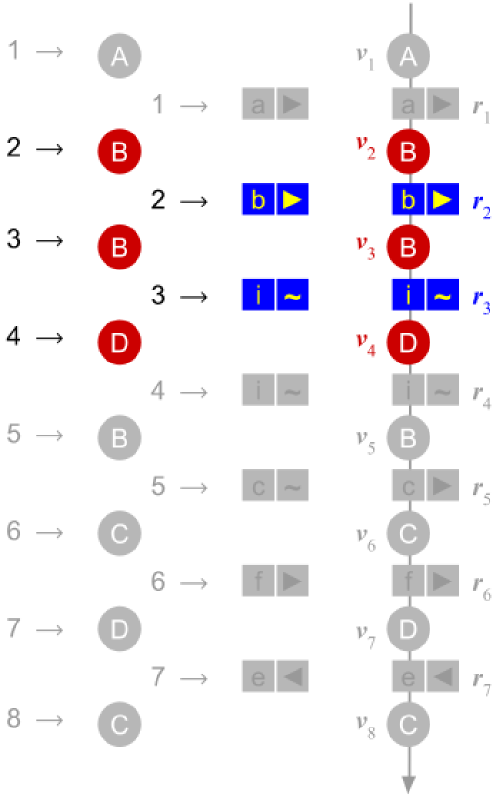
$\boldsymbol{v}' = (\boldsymbol{v}'[1 \ldots 3]) = (\boldsymbol{v}[2 \ldots 4]) = (\boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4) = (\mathrm{B}, \mathrm{B}, \mathrm{D})$

$\boldsymbol{r}' = (\boldsymbol{r}'[1 \ldots 2]) = (\boldsymbol{r}[2 \ldots 3]) = (\boldsymbol{r}_2, \boldsymbol{r}_3) = (\mathrm{b}, \mathrm{i})$

$\boldsymbol{W}_{\varnothing}$ is a subwalk of every walk.

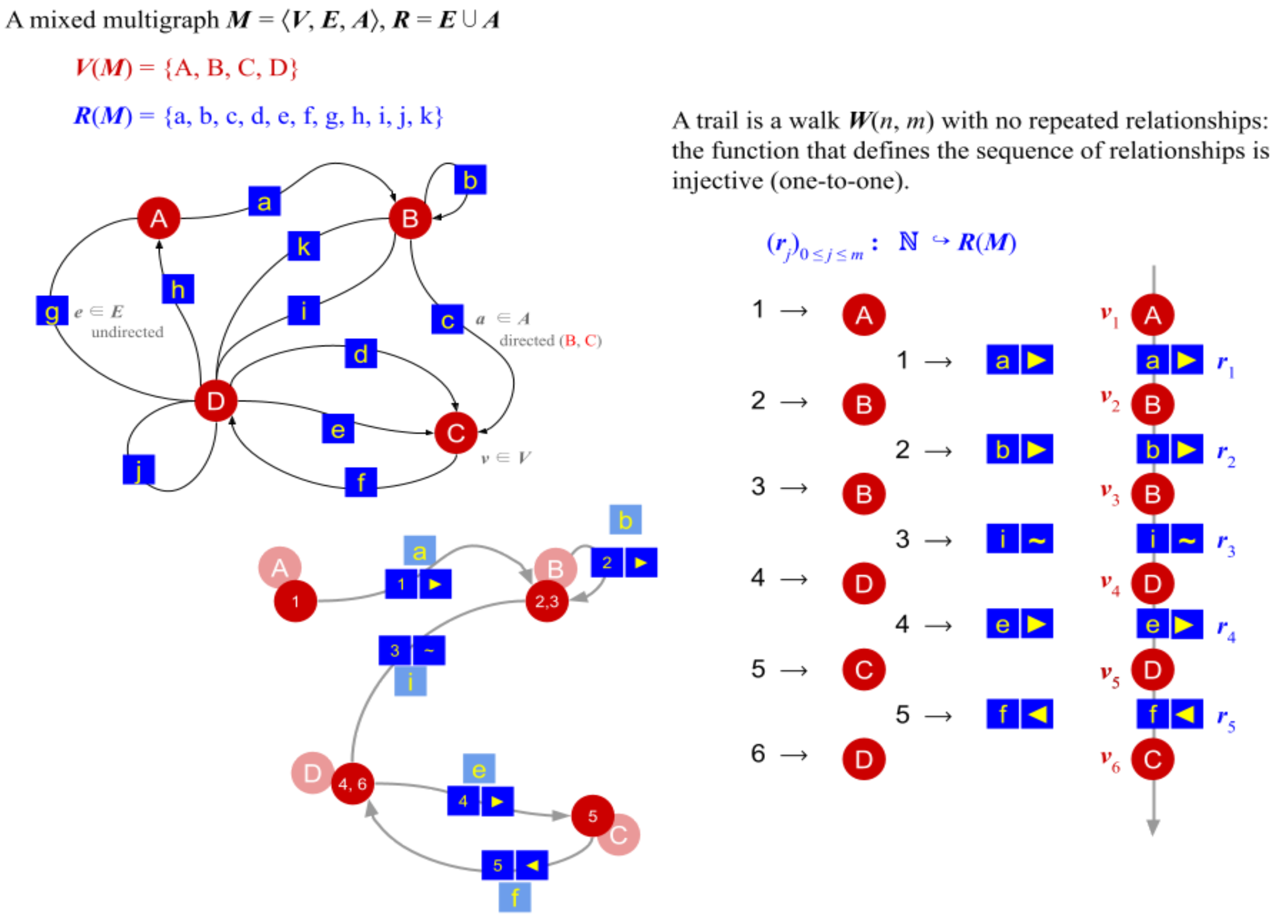
A bounded subwalk $\boldsymbol{W}(n, m)[\mathrm{b} \ldots \mathrm{e}]$ in $\boldsymbol{M}$

$= \langle (\boldsymbol{v}_i)_{0 \le i \le n}[\mathrm{b} \ldots \mathrm{e}], (\boldsymbol{r}_j)_{0 \le j \le m}[\mathrm{b} \ldots \mathrm{e}-1] \rangle.$



11.4. Trail

This diagram shows a trail in $\boldsymbol{M}$.



A mixed multigraph $\boldsymbol{M} = \langle \boldsymbol{V}, \boldsymbol{E}, \boldsymbol{A} \rangle$, $\boldsymbol{R} = \boldsymbol{E} \cup \boldsymbol{A}$

$\boldsymbol{V}(\boldsymbol{M}) = \{\text{A, B, C, D}\}$

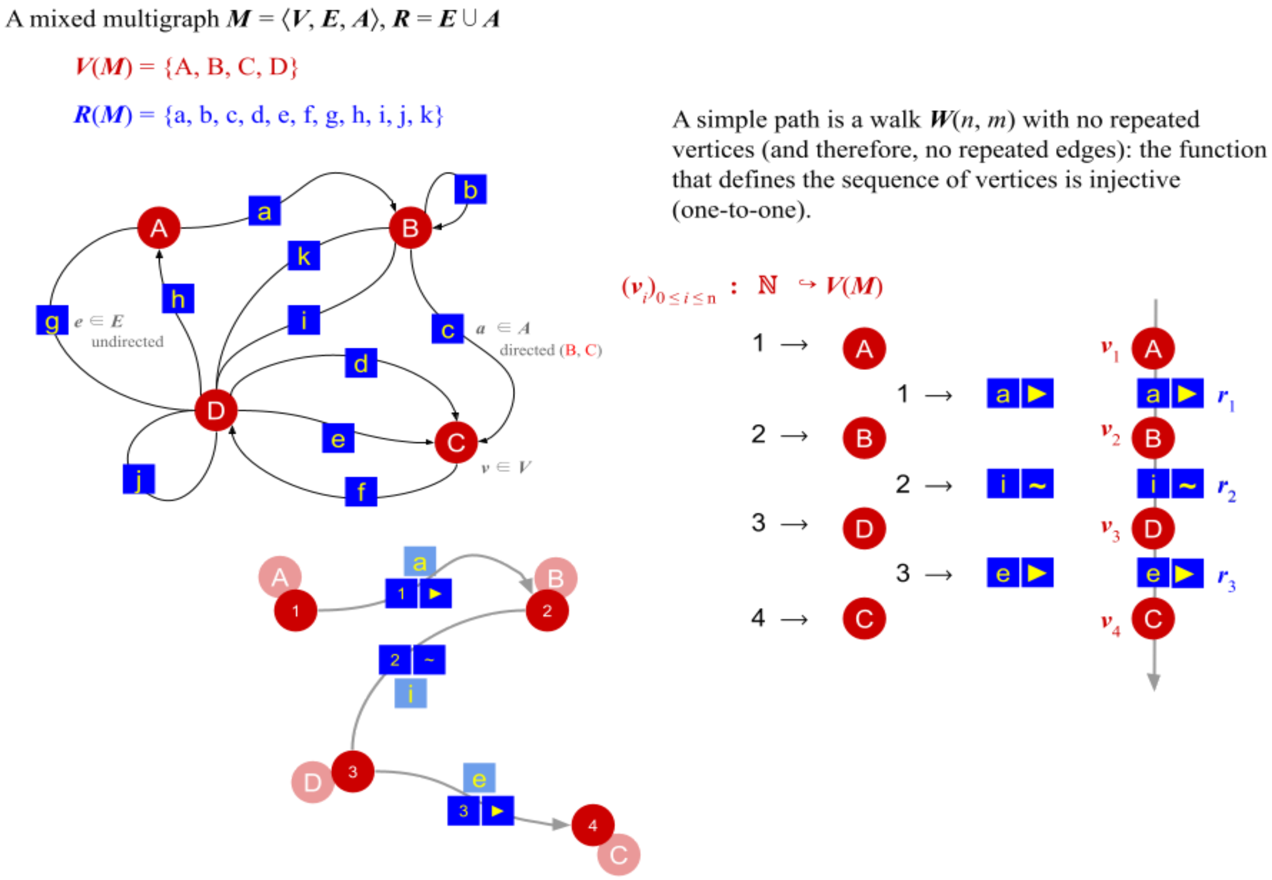
$\boldsymbol{R}(\boldsymbol{M}) = \{\text{a, b, c, d, e, f, g, h, i, j, k}\}$

A trail is a walk $\boldsymbol{W}(n, m)$ with no repeated relationships: the function that defines the sequence of relationships is injective (one-to-one).

$(\boldsymbol{r}_j)_{0 \le j \le m} : \ \mathbb{N} \hookrightarrow \boldsymbol{R}(\boldsymbol{M})$

11.5. Simple path

This diagram shows a simple path in $\boldsymbol{M}$.



A mixed multigraph $\boldsymbol{M} = \langle \boldsymbol{V}, \boldsymbol{E}, \boldsymbol{A} \rangle$, $\boldsymbol{R} = \boldsymbol{E} \cup \boldsymbol{A}$

$\boldsymbol{V}(\boldsymbol{M}) = \{\mathrm{A, B, C, D}\}$

$\boldsymbol{R}(\boldsymbol{M}) = \{\mathrm{a, b, c, d, e, f, g, h, i, j, k}\}$

A simple path is a walk $\boldsymbol{W}(n, m)$ with no repeated vertices (and therefore, no repeated edges): the function that defines the sequence of vertices is injective (one-to-one).

$(\boldsymbol{v}_i)_{0 \le i \le n} : \mathbb{N} \hookrightarrow \boldsymbol{V}(\boldsymbol{M})$