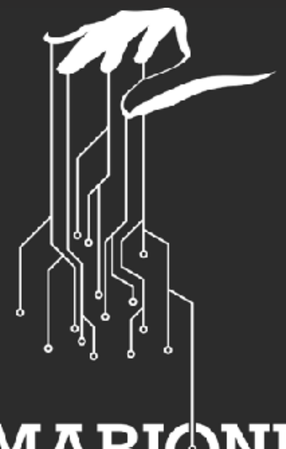
Graphs

From Novice to Graphanista

Dom Davis
@idomdavis

TECH MARIONETTE

“Visualise and control your IT”



TECH MARIONETTE

“Doing bad things to innocent graphs”

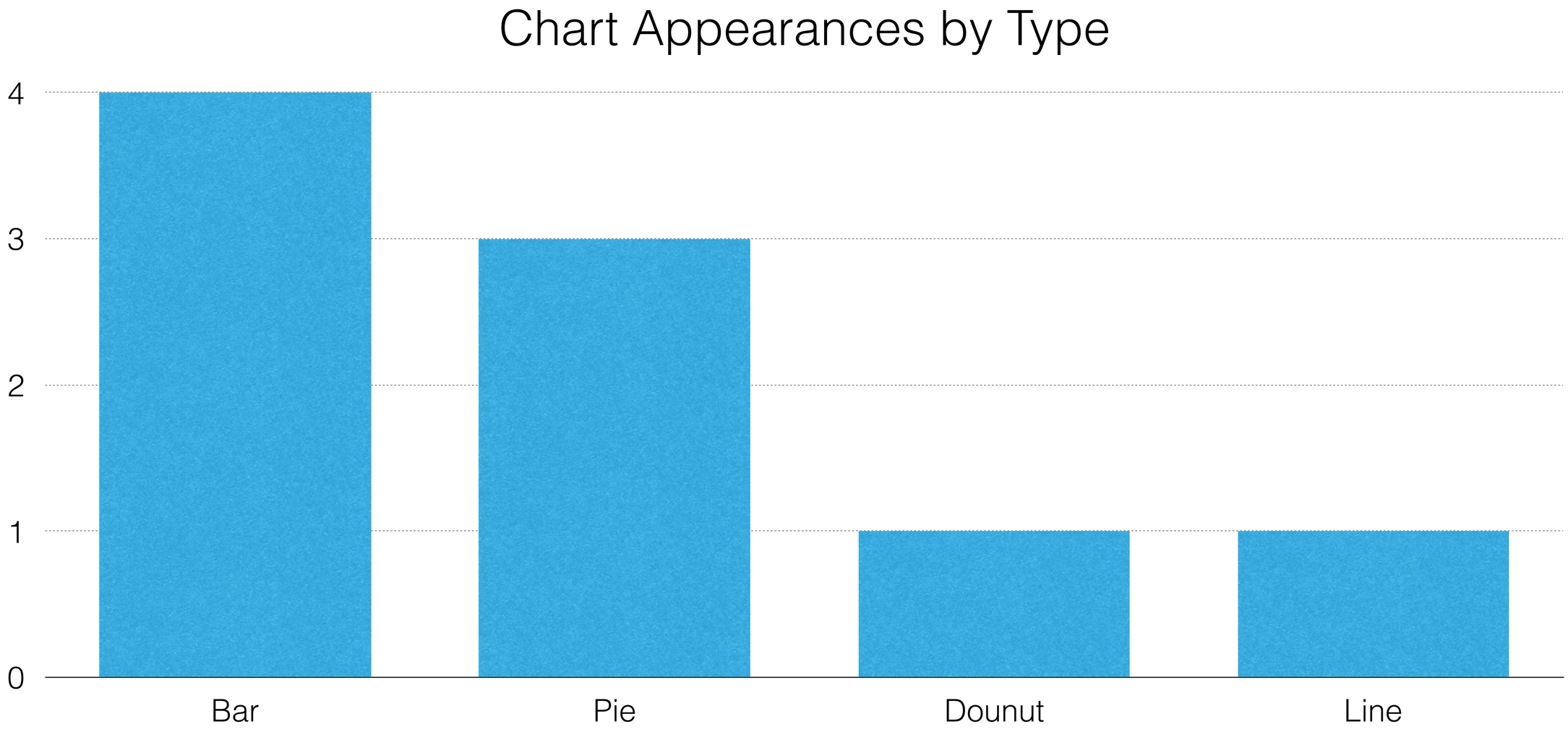


Chart Appearances by Type
4
3
2
1
0
Bar
Pie
Dounut
Line

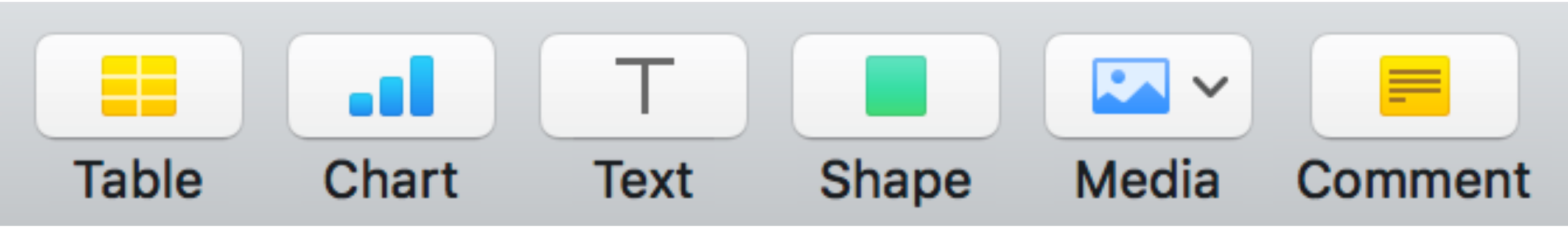


Table
Chart
Text
Shape
Media
Comment

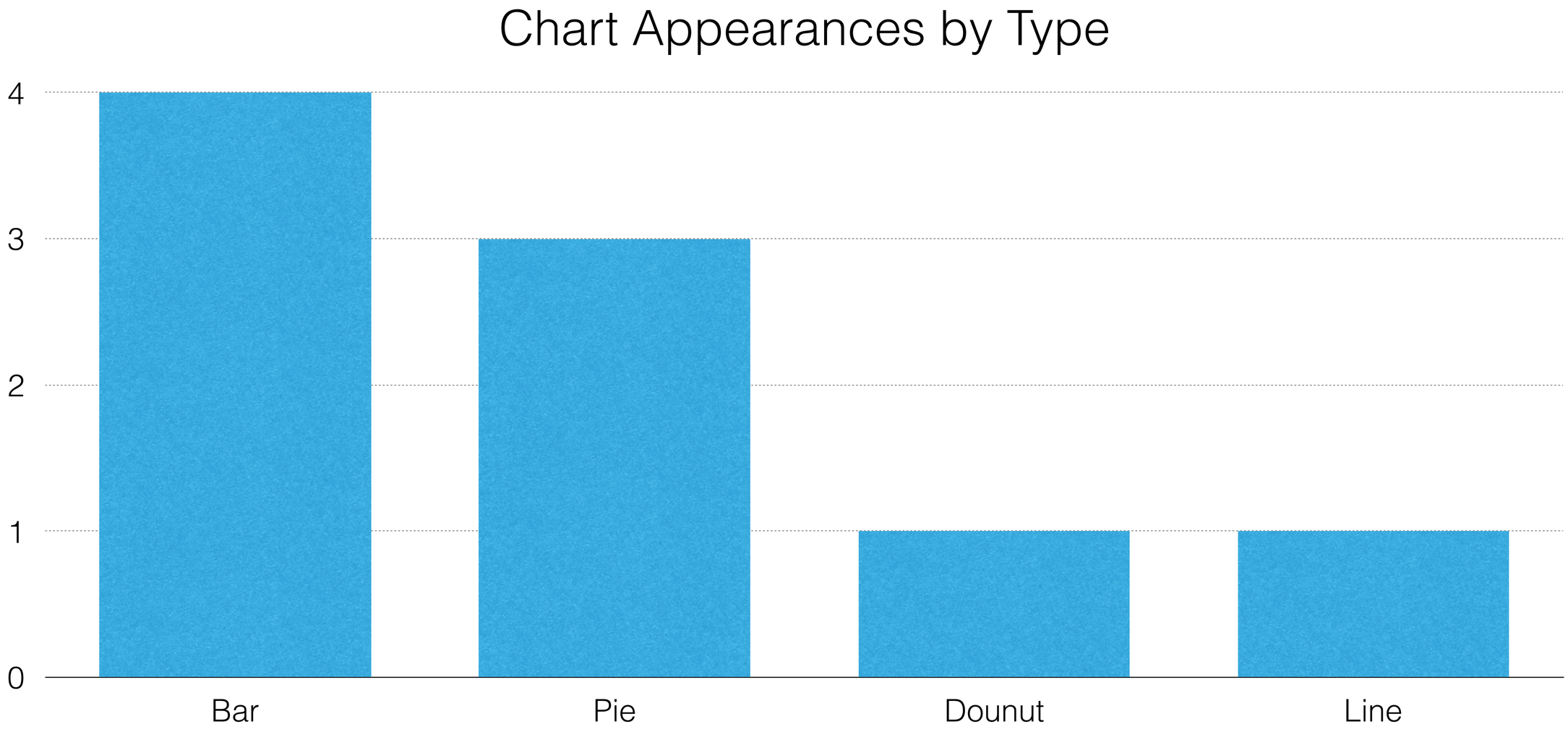
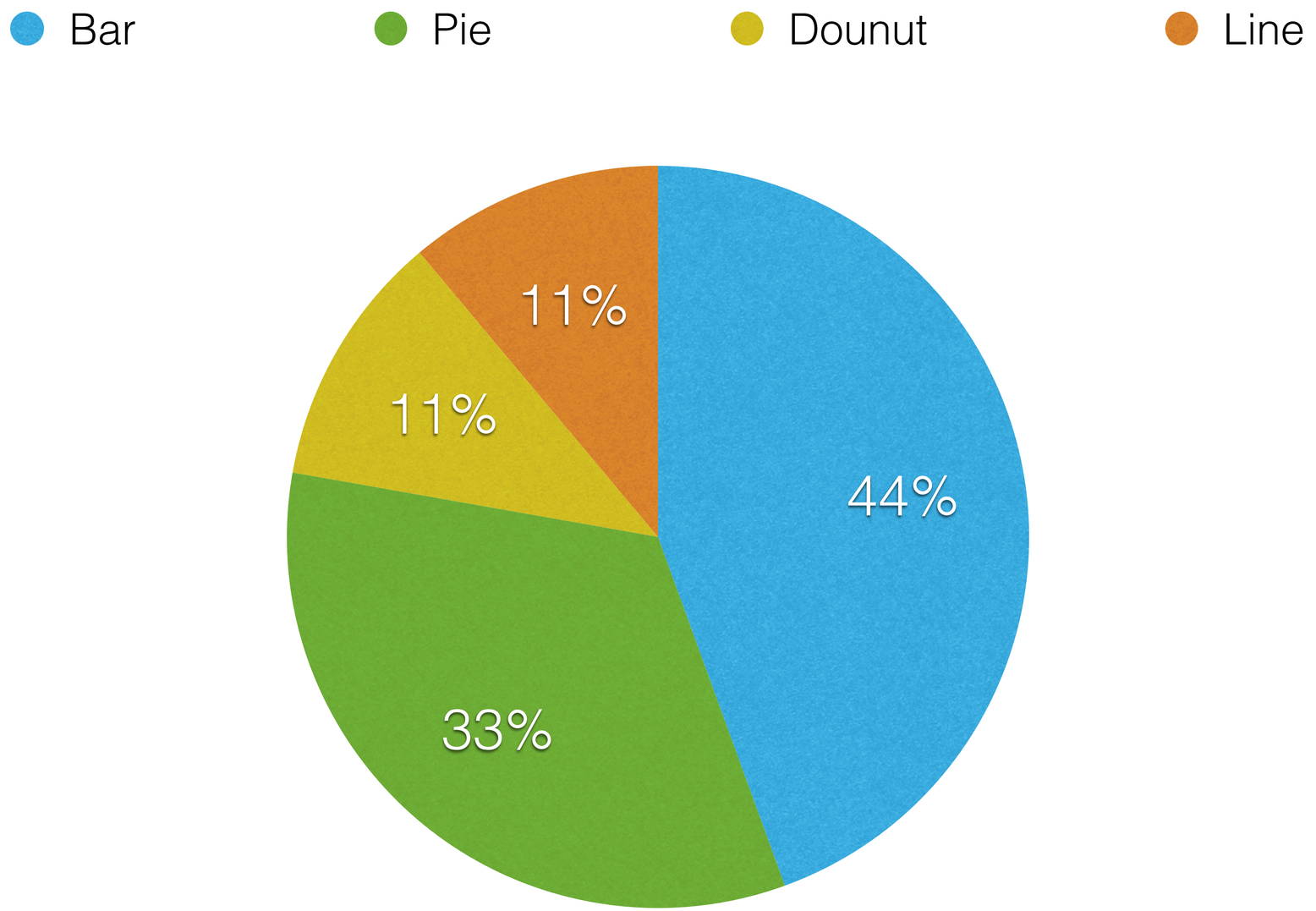
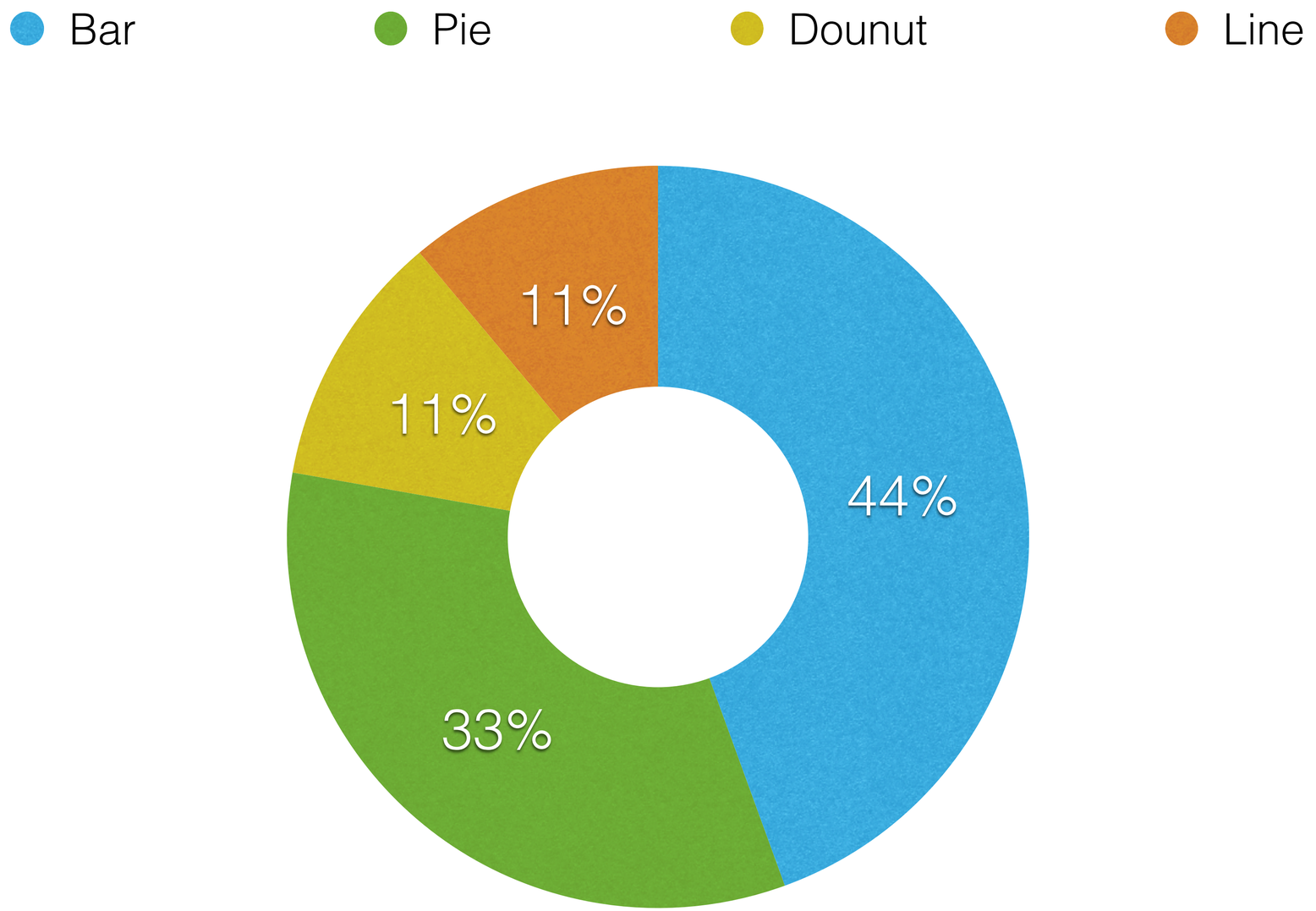
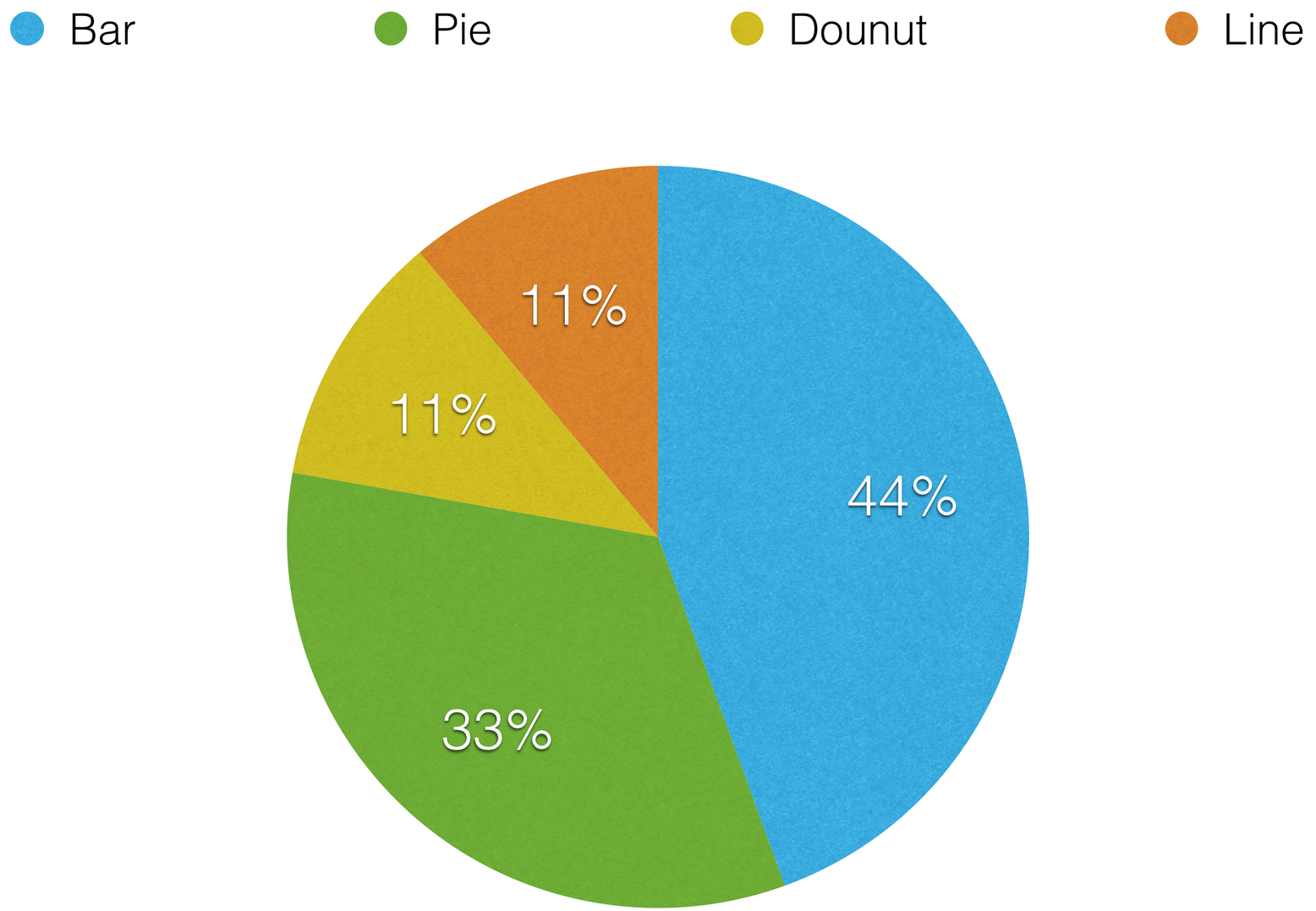


Chart Appearances by Type
4
3
2
1
0
Bar
Pie
Dounut
Line


Bar
Pie
Dounut
Line
11%
11%
44%
33%


Bar
Pie
Dounut
Line
11%
11%
44%
33%


Bar
Pie
Dounut
Line
11%
11%
44%
33%

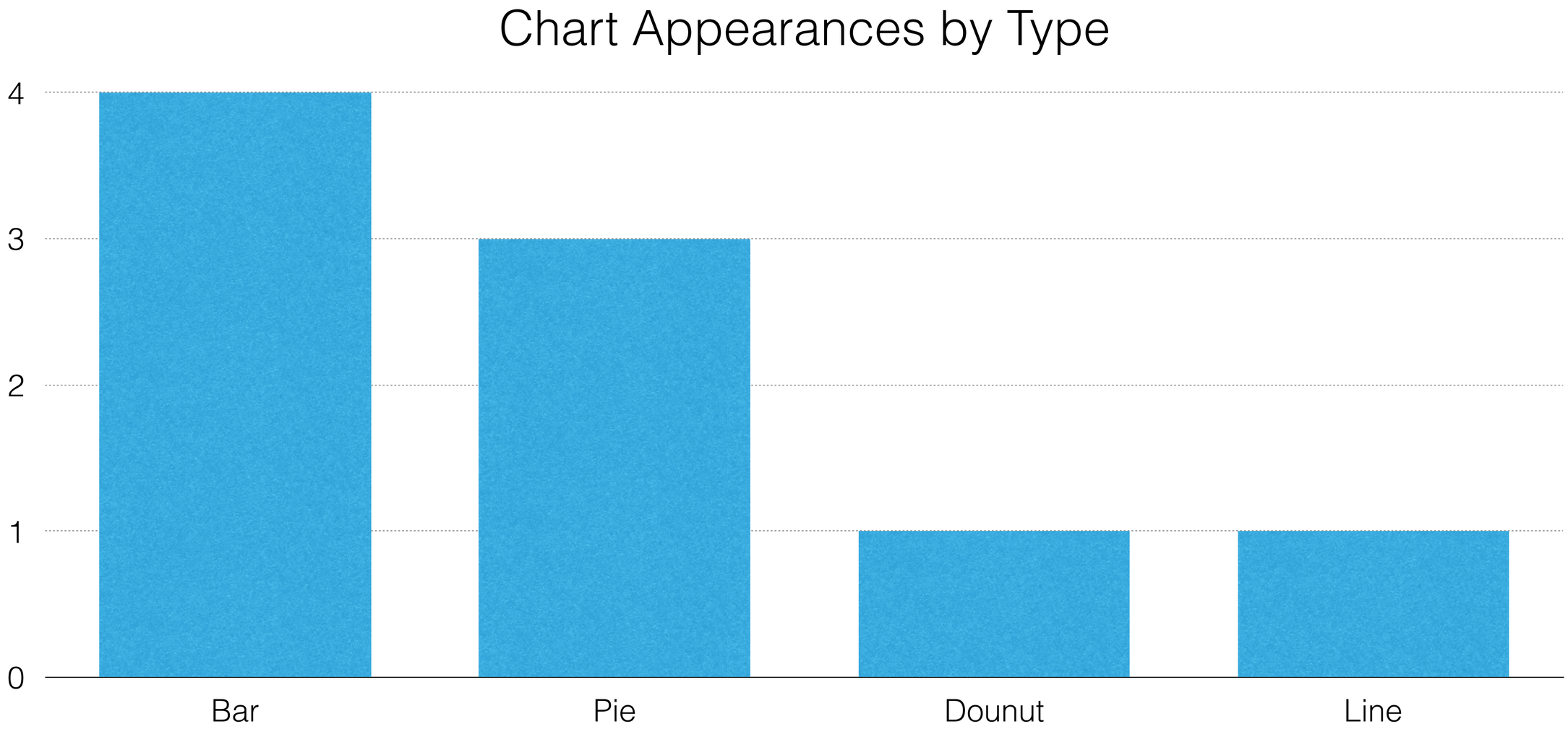
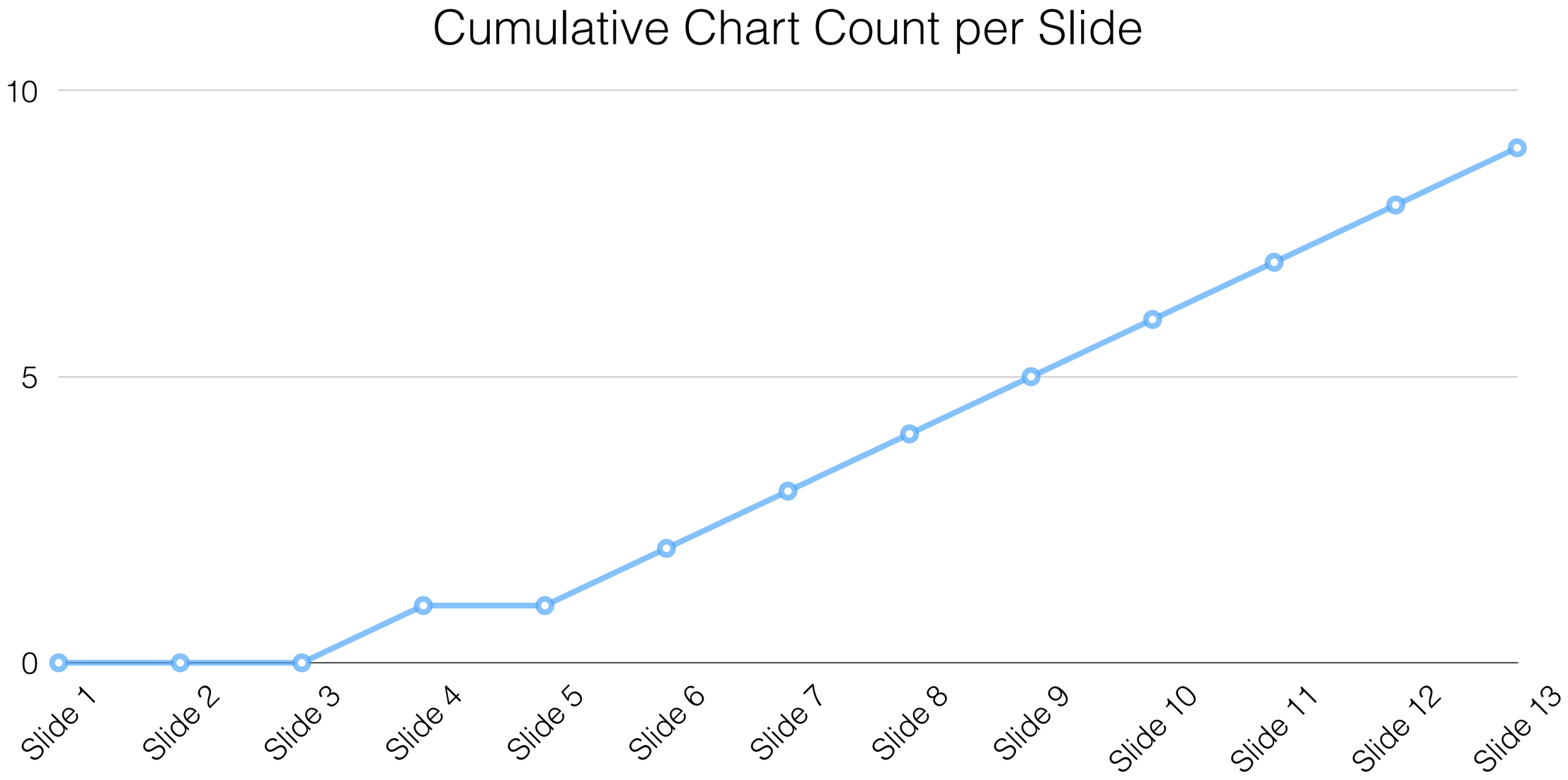


Chart Appearances by Type
4
3
2
1
0
Bar
Pie
Dounut
Line


Cumulative Chart Count per Slide
10
5
0
Slide 1
Slide 2
Slide 3
Slide 4
Slide 5
Slide 6
Slide 7
Slide 8
Slide 9
Slide 10
Slide 11
Slide 12
Slide 13

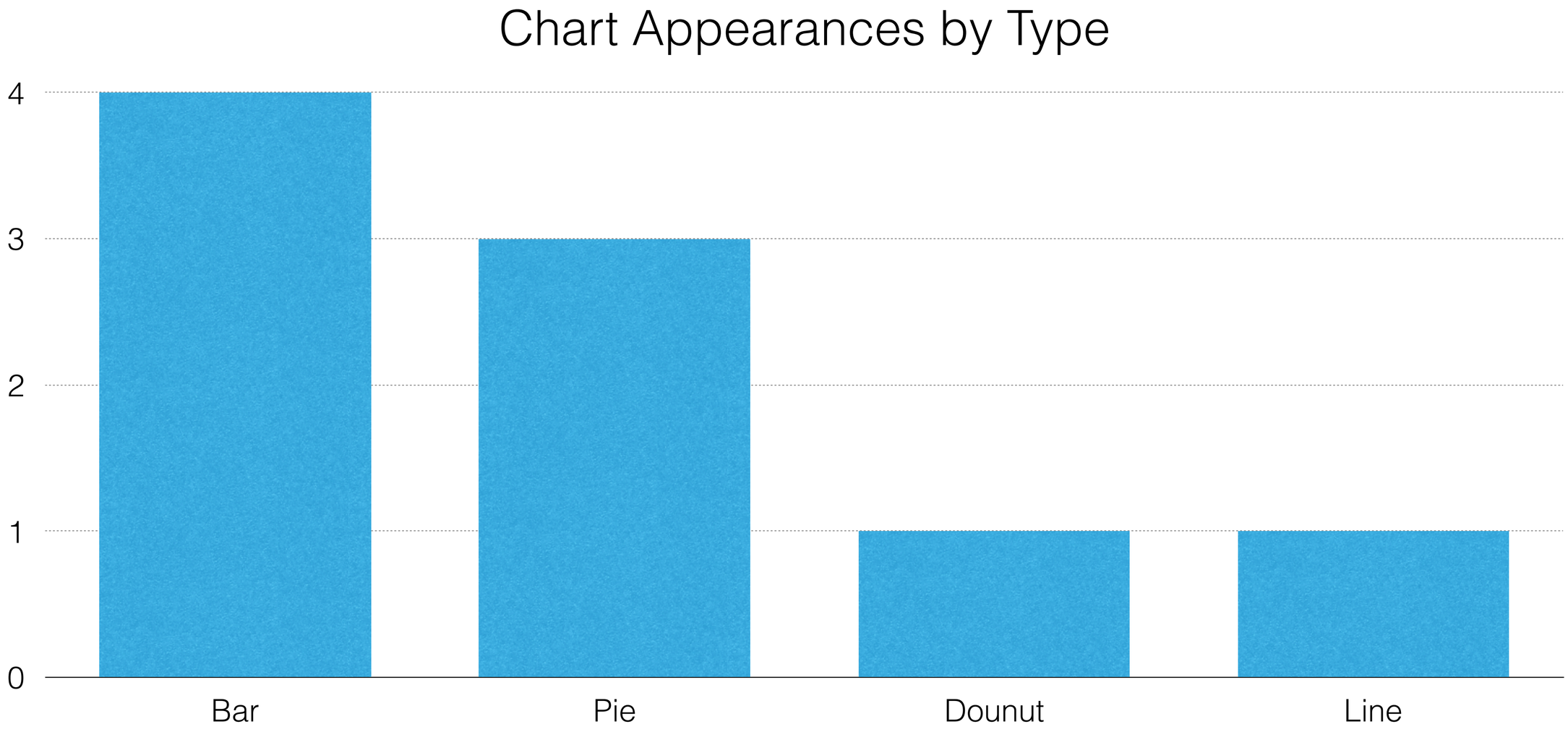
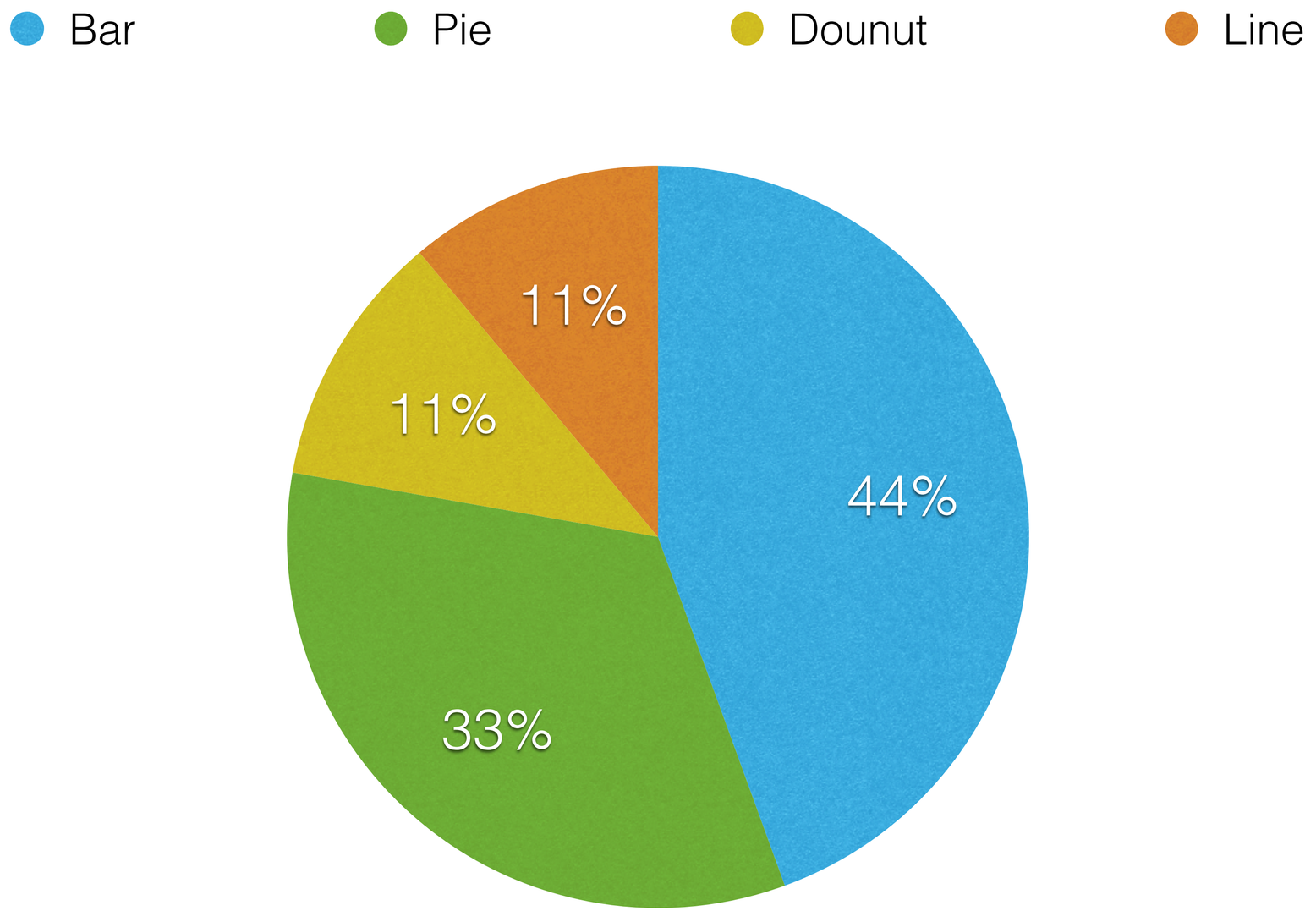
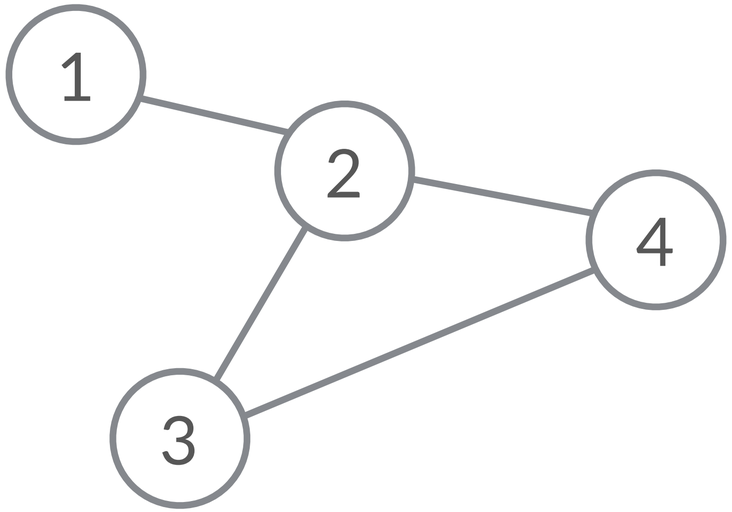
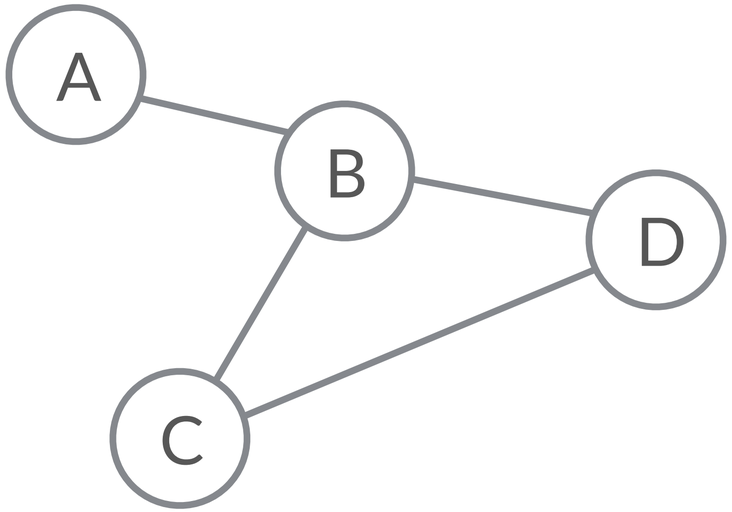


Chart Appearances by Type
4
3
2
1
0
Bar
Pie
Dounut
Line


Bar
Pie
Dounut
Line
11%
11%
44%
33%


1
2
3
4


A
B
C
D

A *directed graph* or *digraph* is a graph in which edges have orientations. It is written as an ordered pair $G = (V, A)$ (sometimes $G = (V, E)$) with

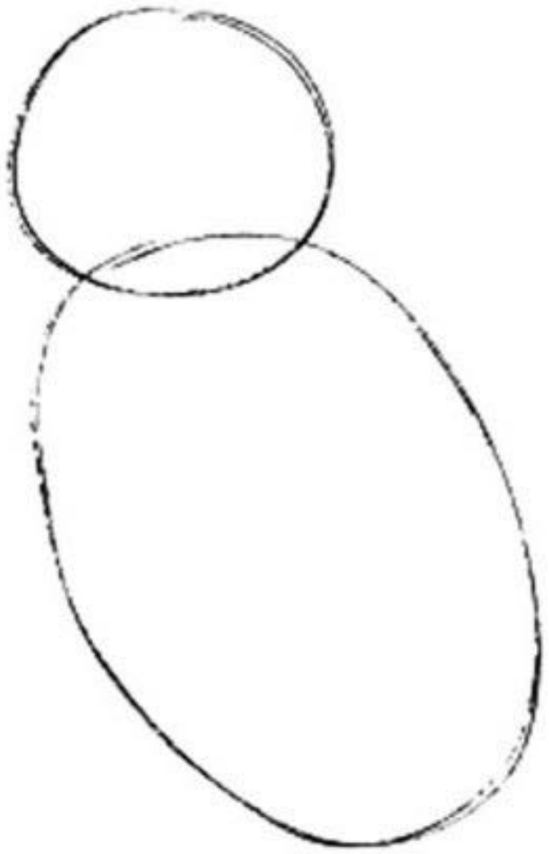
- V a set whose elements are called *vertices*, *nodes*, or *points*;
- A a set of ordered pairs of vertices, called *arrows*, *directed edges* (sometimes simply *edges* with the corresponding set named E instead of A), *directed arcs*, or *directed lines*.

An arrow (x, y) is considered to be directed *from x to y*; y is called the *head* and x is called the *tail* of the arrow; y is said to be a *direct successor* of x and x is said to be a *direct predecessor* of y. If a path leads from x to y, then y is said to be a *successor* of x and *reachable* from x, and x is said to be a *predecessor* of y. The arrow (y, x) is called the *inverted arrow* of (x, y).

A directed graph G is called *symmetric* if, for every arrow in G, the corresponding inverted arrow also belongs to G. A symmetric loopless directed graph $G = (V, A)$ is equivalent to a simple undirected graph $G' = (V, E)$, where the pairs of inverse arrows in A correspond one-to-one with the edges in E; thus the number of edges in G' is $|E| = |A|/2$, that is half the number of arrows in G.

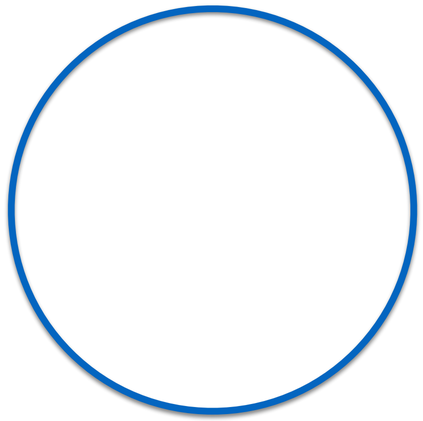
“A graph comprises of vertices and edges, where the edges may be directed or undirected.”

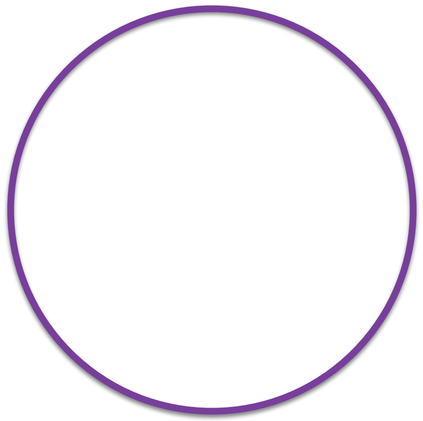
Dom Davis, ACCU Conference 2018



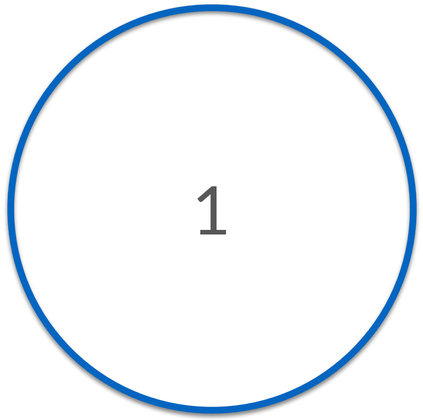
1. Draw two circles

2. Draw the rest of the owl





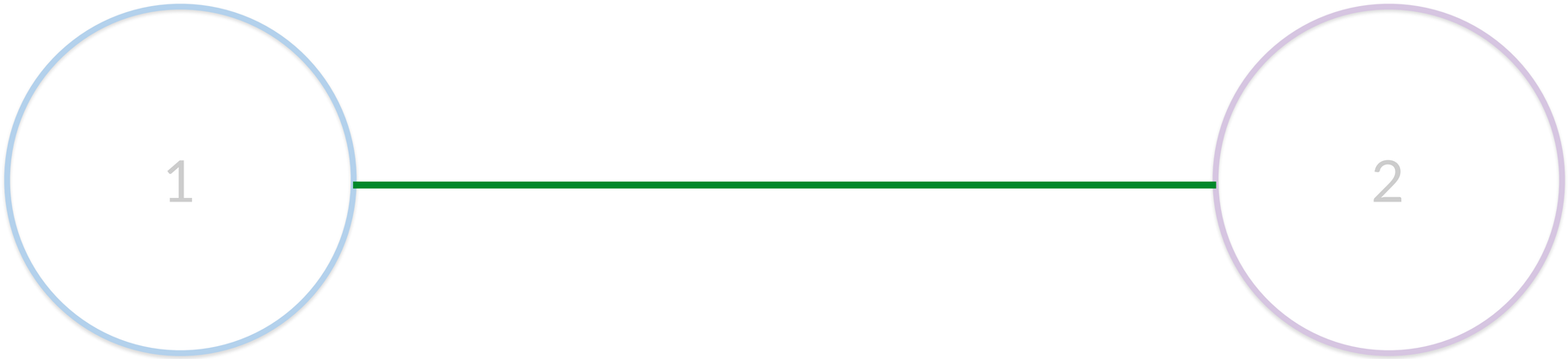

1
2


1

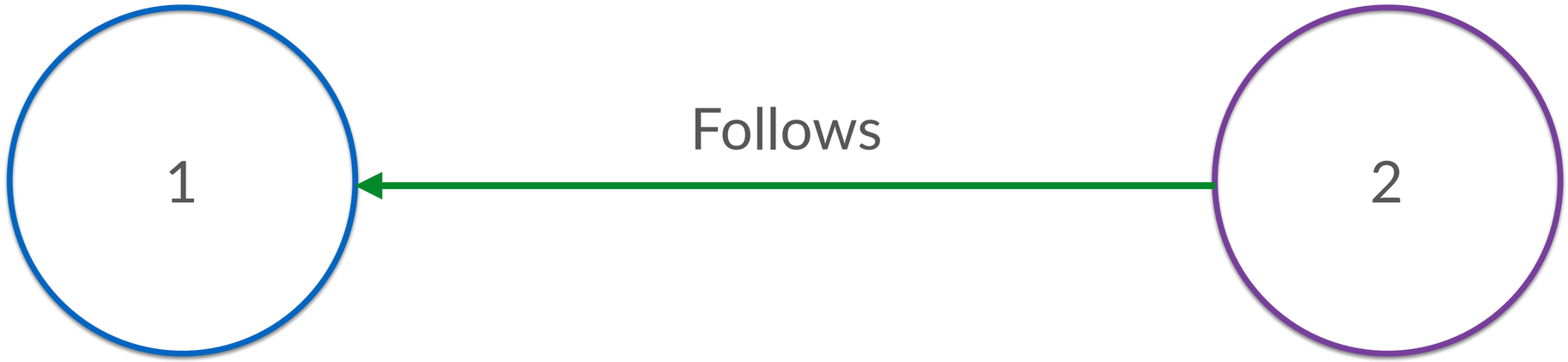

1
2

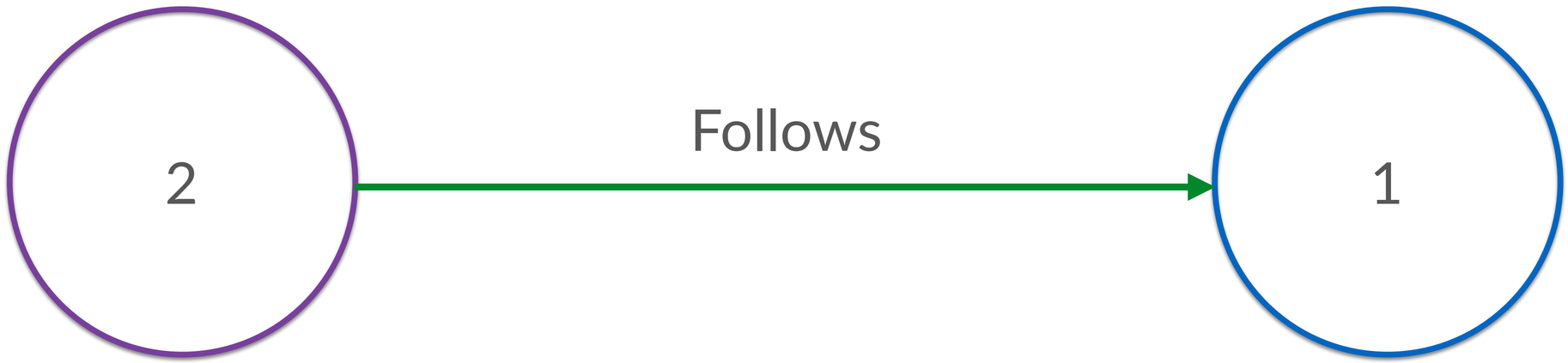

1
2

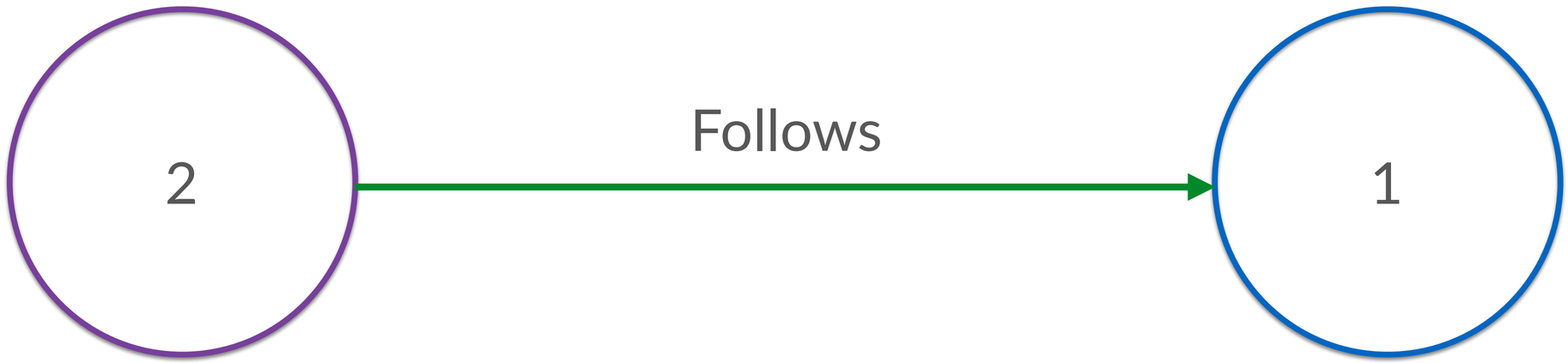

1
2


1
2


1
2


1
Follows
2


2
Follows
1

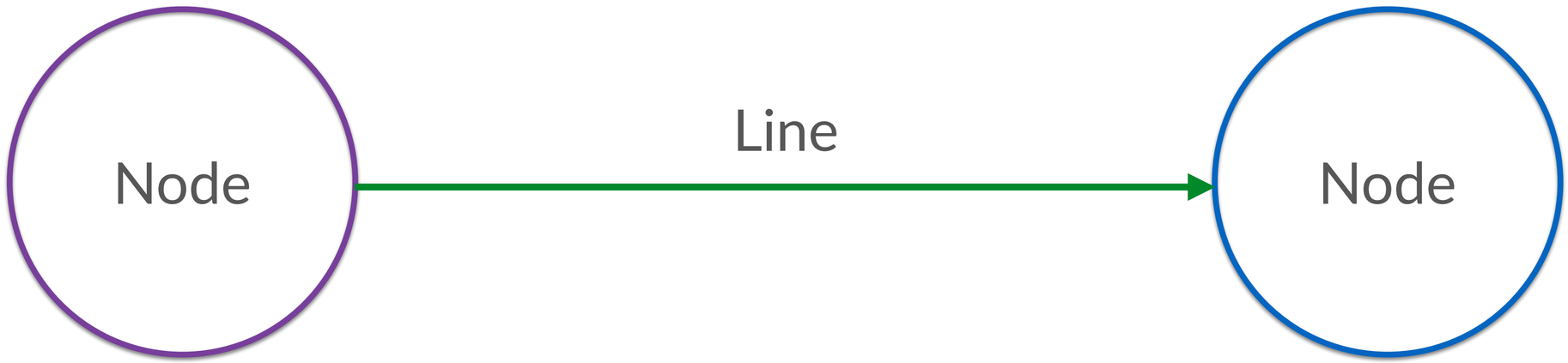

2
Follows
1

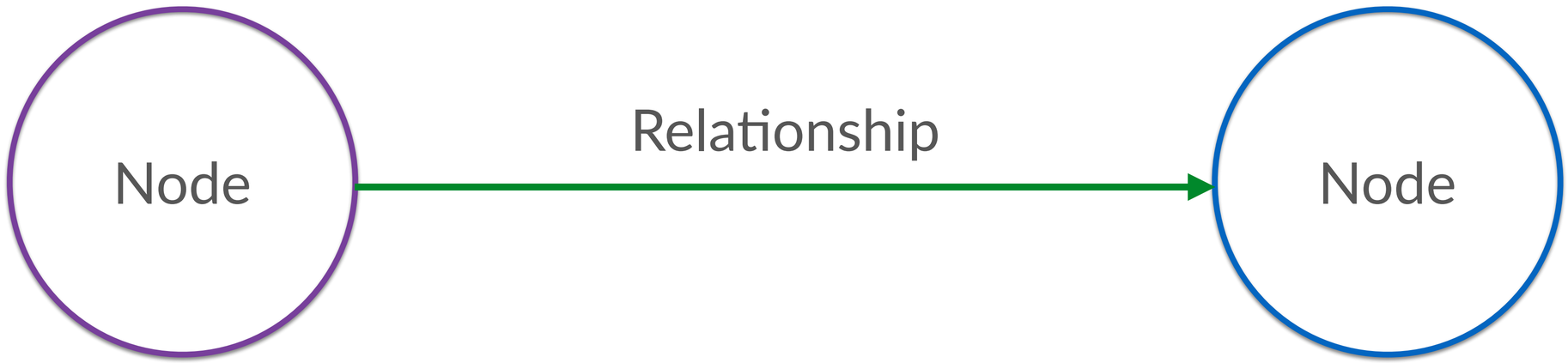

Vertex
Edge
Vertex

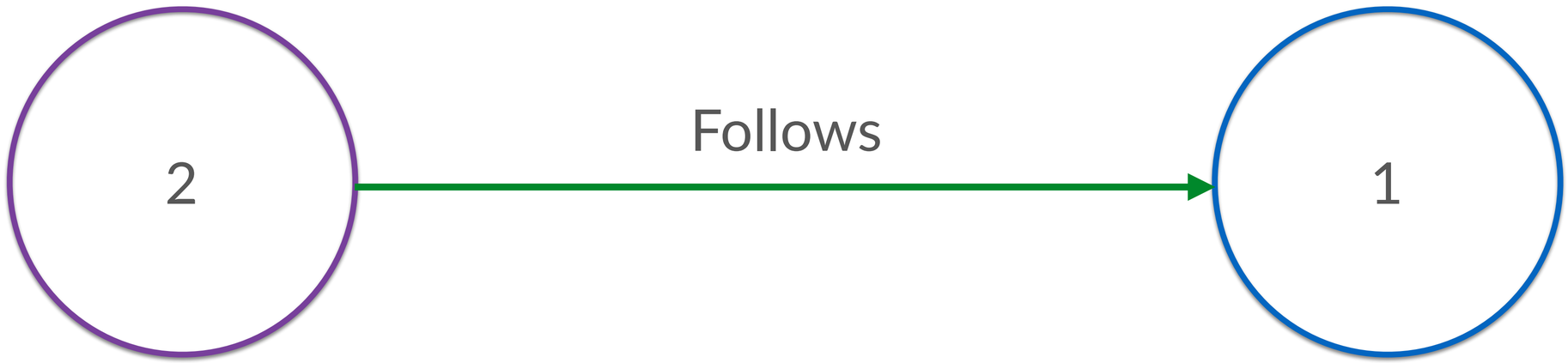

Point
Edge
Point

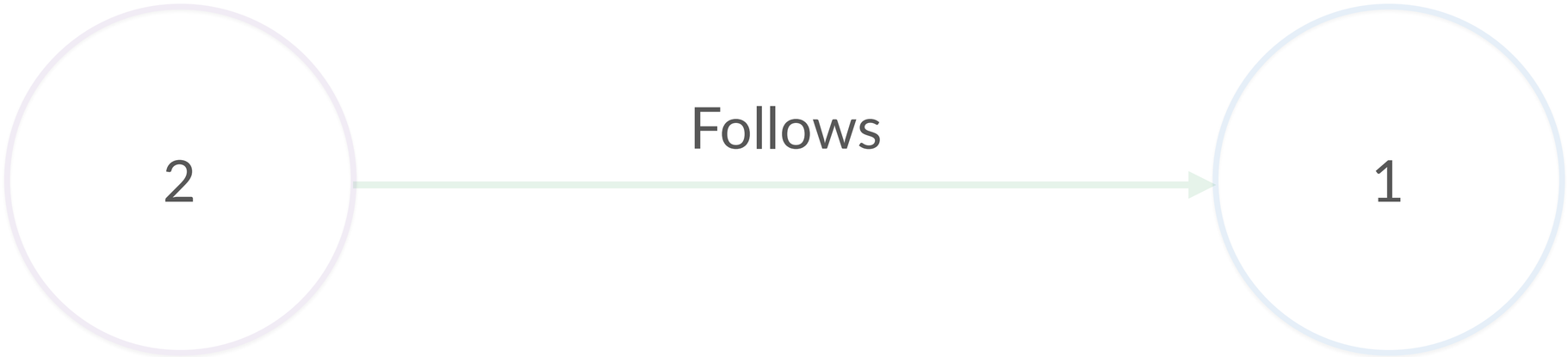

Node
Edge
Node

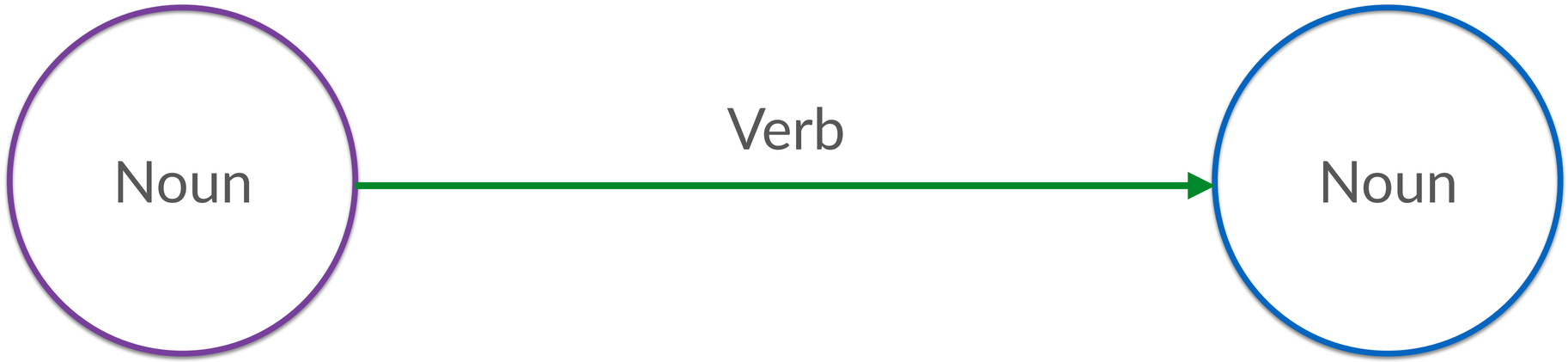

Node
Arc
Node


Node
Line
Node


Node
Relationship
Node

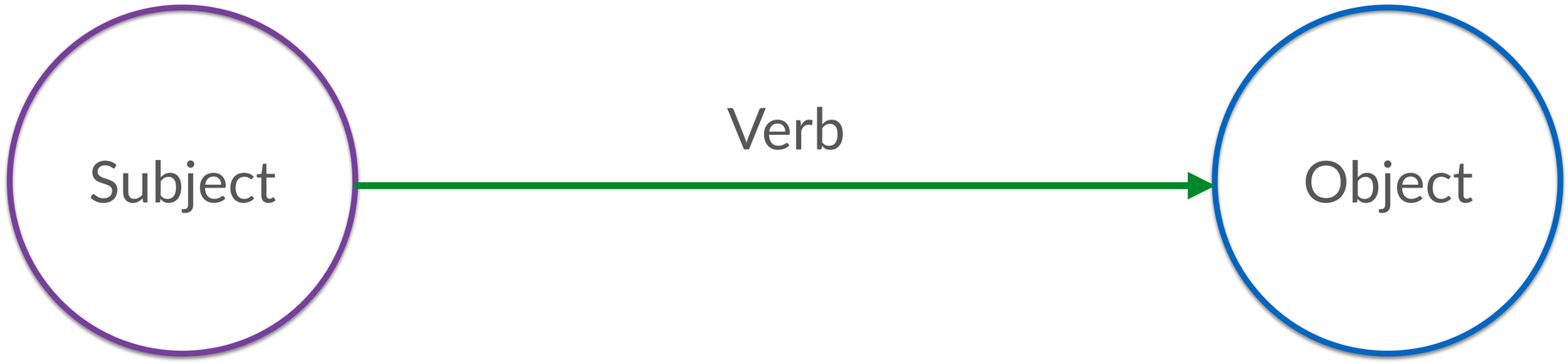

2
Follows
1


2
Follows
1


Noun
Verb
Noun


Graphs
Are
Everywhere


Speaker
Gives
Talk


Subject
Verb
Object

subject - verb - object

subject - object - verb

verb - subject - object

verb - object - subject

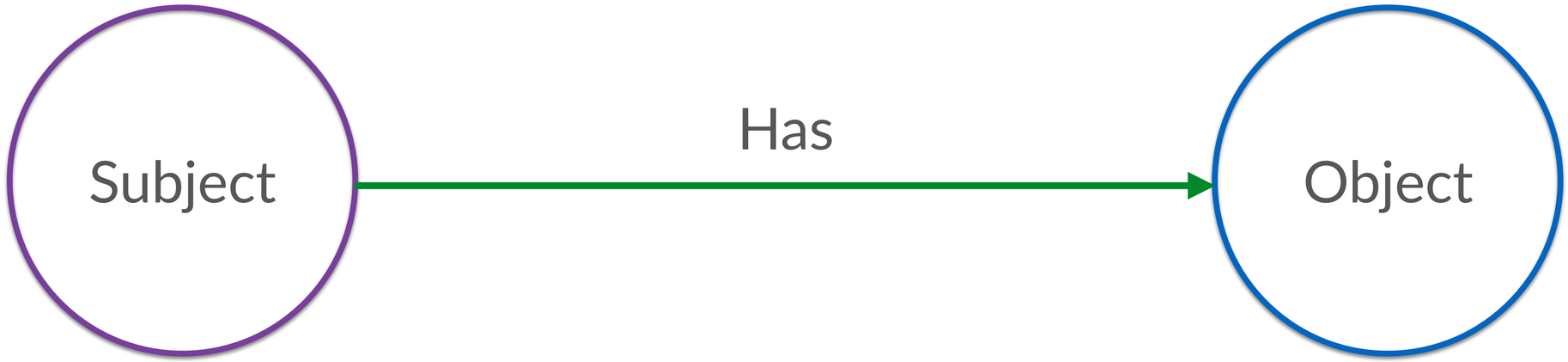
object - verb - subject

object - subject - verb

subject - verb - object

verb(subject, object)

subject.getObject()


Subject
Has
Object

CYPHER

```
MATCH (s)-->(o) RETURN s, o
```

```
MATCH (s)<--(o) RETURN s, o
```

```
MATCH (s)--(o) RETURN s, o
```

```
MATCH (s)-->(o) RETURN s, o
```

```
MATCH (s)-[r]->(o) RETURN s, r, o
```


COVENANT
RITUAL NOISE

We make ritual noise
We weave the fabric of dreams
We make cities of sound
We feel the rhythm of time

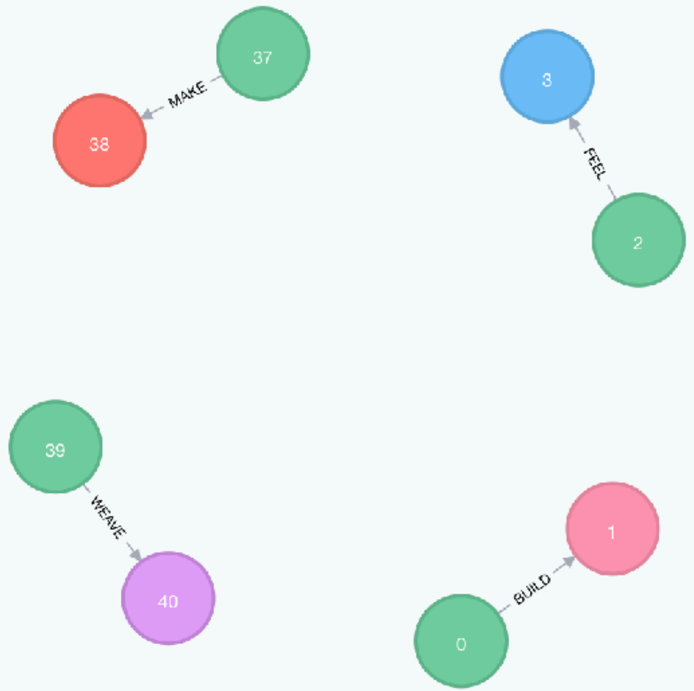
Covenant - Ritual Noise

```
CREATE
  (:We)-[:MAKE]->(:`Ritual noise`),
  (:We)-[:WEAVE]->(:`The fabric of dreams`),
  (:We)-[:BUILD]->(:`Cities of sound`),
  (:We)-[:FEEL]->(:`The rhythm of time`),
```

$ match (n) return n

Graph
Table
Text
Code

*(8) We(4) Cities of sound(1) The rhythm of time(1) Ritual noise(1) The fabric of dreams(1)



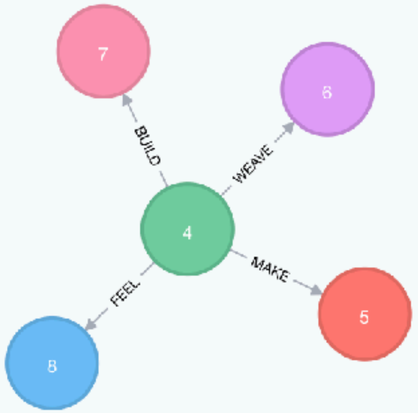
Displaying 8 nodes, 4 relationships.

```
CREATE
  (we:We)-[:MAKE]->(:`Ritual noise`),
  (we)-[:WEAVE]->(:`The fabric of dreams`),
  (we)-[:BUILD]->(:`Cities of sound`),
  (we)-[:FEEL]->(:`The rhythm of time`)
```

$ match (n) return n

Graph | Table | Text | Code

*(5) | We(1) | Ritual noise(1) | The fabric of dreams(1) | Cities of sound(1) | The rhythm of time(1)



Displaying 5 nodes, 4 relationships.

```
CREATE (we:Lyric {words: 'We'})
```

```
CREATE (we:Lyric {words: 'We'}),
  (we)-[:MAKE]->(:Lyric {words: 'ritual noise'}),
  (we)-[:WEAVE]->(:Lyric {words: 'the fabric of dreams'}),
  (we)-[:BUILD]->(:Lyric {words: 'cities of sound'}),
  (we)-[:FEEL]->(:Lyric {words: 'the rhythm of time'})
```

$ match (n) return n

Graph

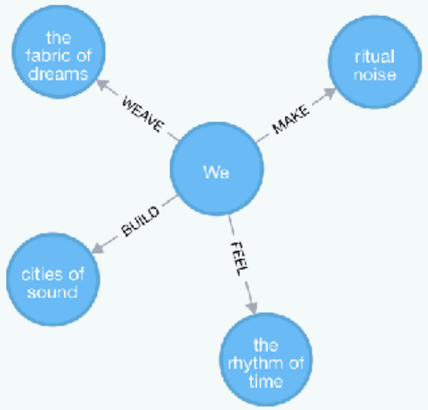
Table

Text

Code

*(5)

Lyric(5)



Displaying 5 nodes, 4 relationships.

We make ritual noise
We weave the fabric of dreams
We make cities of sound
We feel the rhythm of time

Covenant - Ritual Noise

$ match (n) return n

Graph

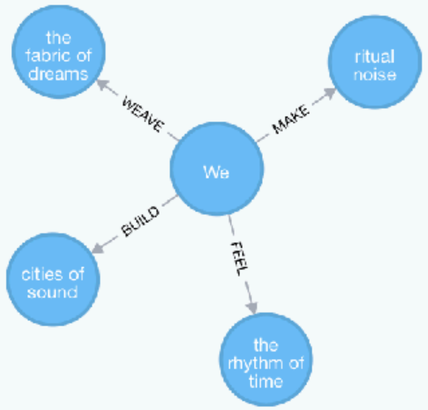
Table

Text

Code

*(5)

Lyric(5)



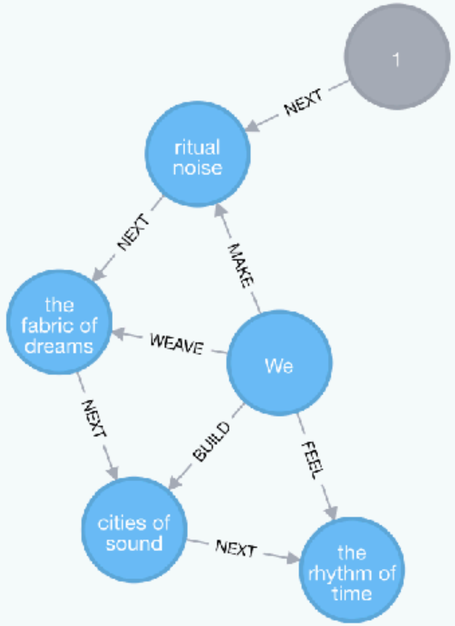
Displaying 5 nodes, 4 relationships.

```
MATCH (l1:Lyric), (l2:Lyric), (l3:Lyric), (l4:Lyric)
  WHERE
  l1.words = 'ritual noise' AND
  l2.words = 'the fabric of dreams' AND
  l3.words = 'cities of sound' AND
  l4.words = 'the rhythm of time'
CREATE
  (:Start)-[:NEXT]->(l1)-[:NEXT]->
  (l2)-[:NEXT]->(l3)-[:NEXT]->(l4)
```

$ match (n) return n

Graph | Table | Text | Code

*(6) Lyric(5) Start(1)



Displaying 6 nodes, 8 relationships.

```
CREATE (we:Lyric {words: 'We'}),
  (we)-[:MAKE {line: 1}]->(:Lyric {words: 'ritual noise'}),
  (we)-[:WEAVE {line: 2}]->(:Lyric {words: 'the fabric of dreams'}),
  (we)-[:BUILD {line: 3}]->(:Lyric {words: 'cities of sound'}),
  (we)-[:FEEL {line: 4}]->(:Lyric {words: 'the rhythm of time'})
```

$ match (n) return n

Graph

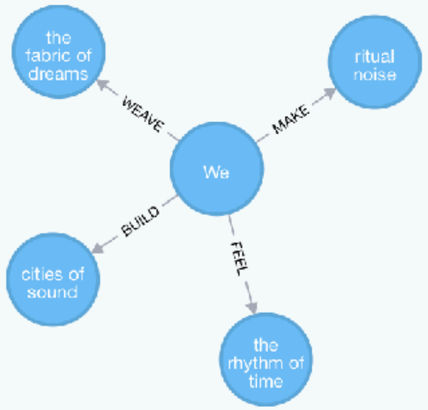
Table

Text

Code

*(5)

Lyric(5)



Displaying 5 nodes, 4 relationships.

```
$ MATCH (s)-[r]->(o) RETURN s.words, toLower(type(r)), o.words ORDER BY r.line
```

Table | Text | Code

s.words	toLower(type(r))	o.words
"We"	"make"	"ritual noise"
"We"	"weave"	"the fabric of dreams"
"We"	"build"	"cities of sound"
"We"	"feel"	"the rhythm of time"

Started streaming 4 records after 2 ms and completed after 2 ms.

```
$ MATCH (s)-[r]->(o) RETURN s.words, toLower(type(r)), o.words ORDER BY r.line
```

Table

Text

Code

s.words	toLower(type(r))	o.words
"We"	"make"	"ritual noise"
"We"	"weave"	"the fabric of dreams"
"We"	"build"	"cities of sound"
"We"	"feel"	"the rhythm of time"

```
MATCH (s)-[r]->(o)
RETURN s.words, toLower(type(r)), o.words
  ORDER BY r.line
```

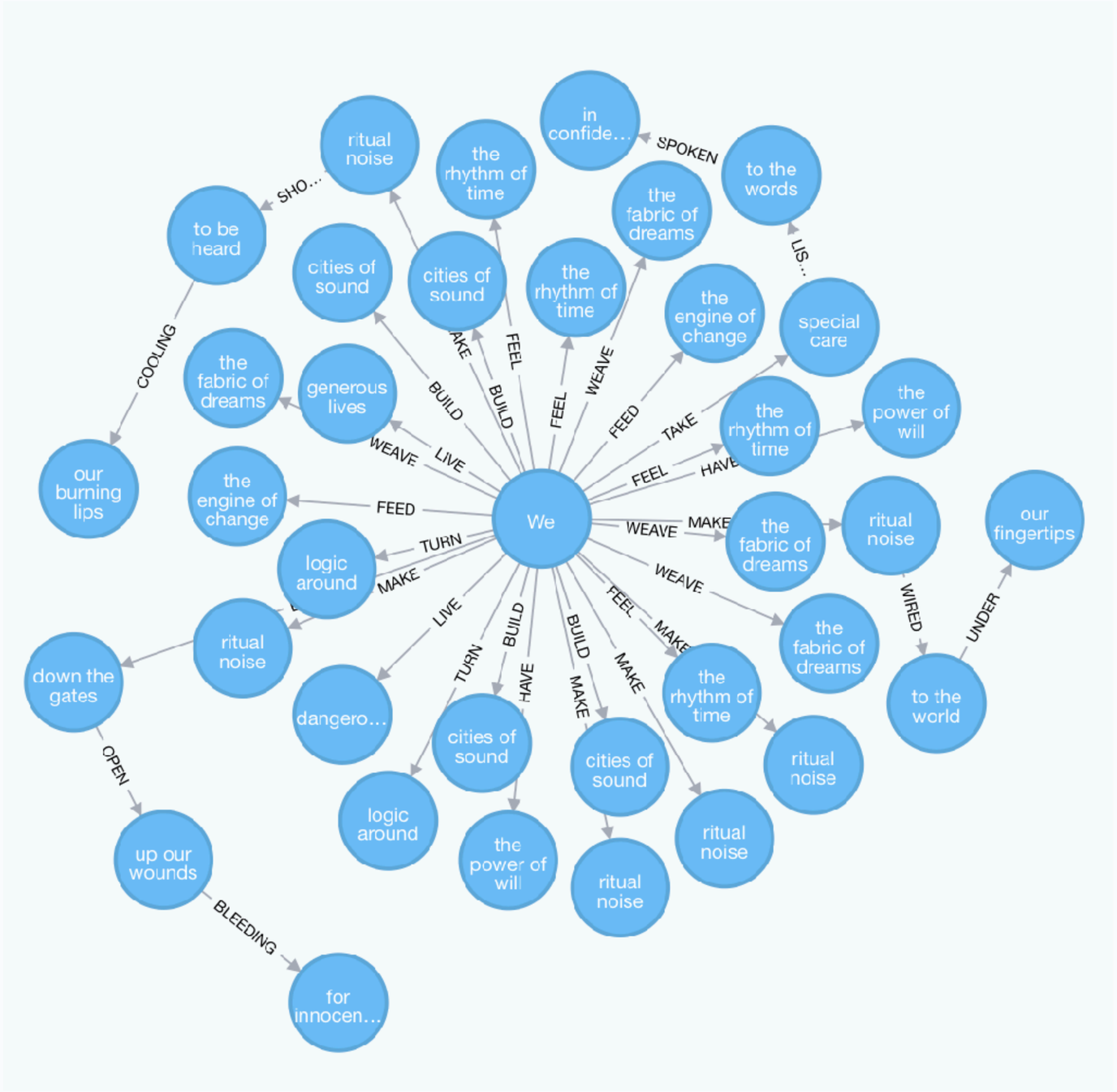
```
$ MATCH (s)-[r]->(o) RETURN s.words, toLower(type(r)), o.words ORDER BY r.line
```

Table

Text

Code

s.words	toLower(type(r))	o.words
"We"	"make"	"ritual noise"
"We"	"weave"	"the fabric of dreams"
"We"	"build"	"cities of sound"
"We"	"feel"	"the rhythm of time"


We
ritual noise
to be heard
cities of sound
cities of sound
the rhythm of time
in confide…
the fabric of dreams
to the words
the rhythm of time
the engine of change
special care
the fabric of dreams
generous lives
the rhythm of time
the power of will
our burning lips
the engine of change
logic around
ritual noise
down the gates
dangero…
cities of sound
logic around
the power of will
cities of sound
ritual noise
ritual noise
the rhythm of time
ritual noise
the fabric of dreams
ritual noise
the fabric of dreams
to the world
our fingertips
up our wounds
for innocen…
SHO…
SPOKEN
LIS…
COOLING
BUILD
BUILD
FEEL
WEAVE
FEED
TAKE
FEEL
HAVE
WEAVE
LIVE
FEED
TURN
MAKE
WEAVE
MAKE
WEAVE
FEEL
MAKE
LIVE
TURN
BUILD
HAVE
BUILD
MAKE
MAKE
WIRED
UNDER
OPEN
BLEEDING

```
MATCH (s {words: "We"})-[r1]->(o)
OPTIONAL MATCH (o)-[r2 {line: r1.line}]->(n1)
OPTIONAL MATCH (n1)-[r3 {line: r1.line}]->(n2)
RETURN r1.line, s.words, toLower(type(r1)), o.words, toLower(type(r2)), n1.words,
       toLower(type(r3)), n2.words
  ORDER BY r1.line
```

$ MATCH (s {words: "We"})-[r1]->(o) OPTIONAL MATCH (o)-[r2 {line: r1.line}]->(n1) OPTIONAL MATCH (n1)-[r3 {line: r1.line}]->(n2) …

Table

Text

Code

r1.line	s.words	toLower(type(r1))	o.words	toLower(type(r2))	n1.words	toLower(type(r3))	n2.words
1	"We"	"make"	"ritual noise"	(empty)	(empty)	(empty)	(empty)
2	"We"	"weave"	"the fabric of dreams"	(empty)	(empty)	(empty)	(empty)
3	"We"	"build"	"cities of sound"	(empty)	(empty)	(empty)	(empty)
4	"We"	"feel"	"the rhythm of time"	(empty)	(empty)	(empty)	(empty)
5	"We"	"make"	"ritual noise"	(empty)	(empty)	(empty)	(empty)
6	"We"	"weave"	"the fabric of dreams"	(empty)	(empty)	(empty)	(empty)
7	"We"	"build"	"cities of sound"	(empty)	(empty)	(empty)	(empty)
8	"We"	"feel"	"the rhythm of time"	(empty)	(empty)	(empty)	(empty)
9	"We"	"make"	"ritual noise"	"wired"	"to the world"	"under"	"our fingertips"
10	"We"	"take"	"special care"	"listen"	"to the words"	"spoken"	"in confidence"
11	"We"	"make"	"ritual noise"	"shouting"	"to be heard"	"cooling"	"our burning lips"
12	"We"	"break"	"down the gates"	"open"	"up our wounds"	"bleeding"	"for innocence"
13	"We"	"make"	"ritual noise"	(empty)	(empty)	(empty)	(empty)
14	"We"	"weave"	"the fabric of dreams"	(empty)	(empty)	(empty)	(empty)
15	"We"	"build"	"cities of sound"	(empty)	(empty)	(empty)	(empty)
16	"We"	"feel"	"the rhythm of time"	(empty)	(empty)	(empty)	(empty)

Started streaming 28 records after 5 ms and completed after 5 ms.

https://github.com/domdavis/ritualnoise/

(graphs)-[:ARE]->(everywhere)

```
$ MATCH (s)-[r]->(o) RETURN s.words, toLower(type(r)), o.words ORDER BY r.line
```

Table

Text

Code

s.words	toLower(type(r))	o.words
"We"	"make"	"ritual noise"
"We"	"weave"	"the fabric of dreams"
"We"	"build"	"cities of sound"
"We"	"feel"	"the rhythm of time"

```
{
  "name": "react-app",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "react": "^16.3.0",
    "react-dom": "^16.3.0",
    "react-scripts": "1.1.2"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test --env=jsdom",
    "eject": "react-scripts eject"
  }
}
```

“Typing ability is inversely proportional
to the number of people watching.”

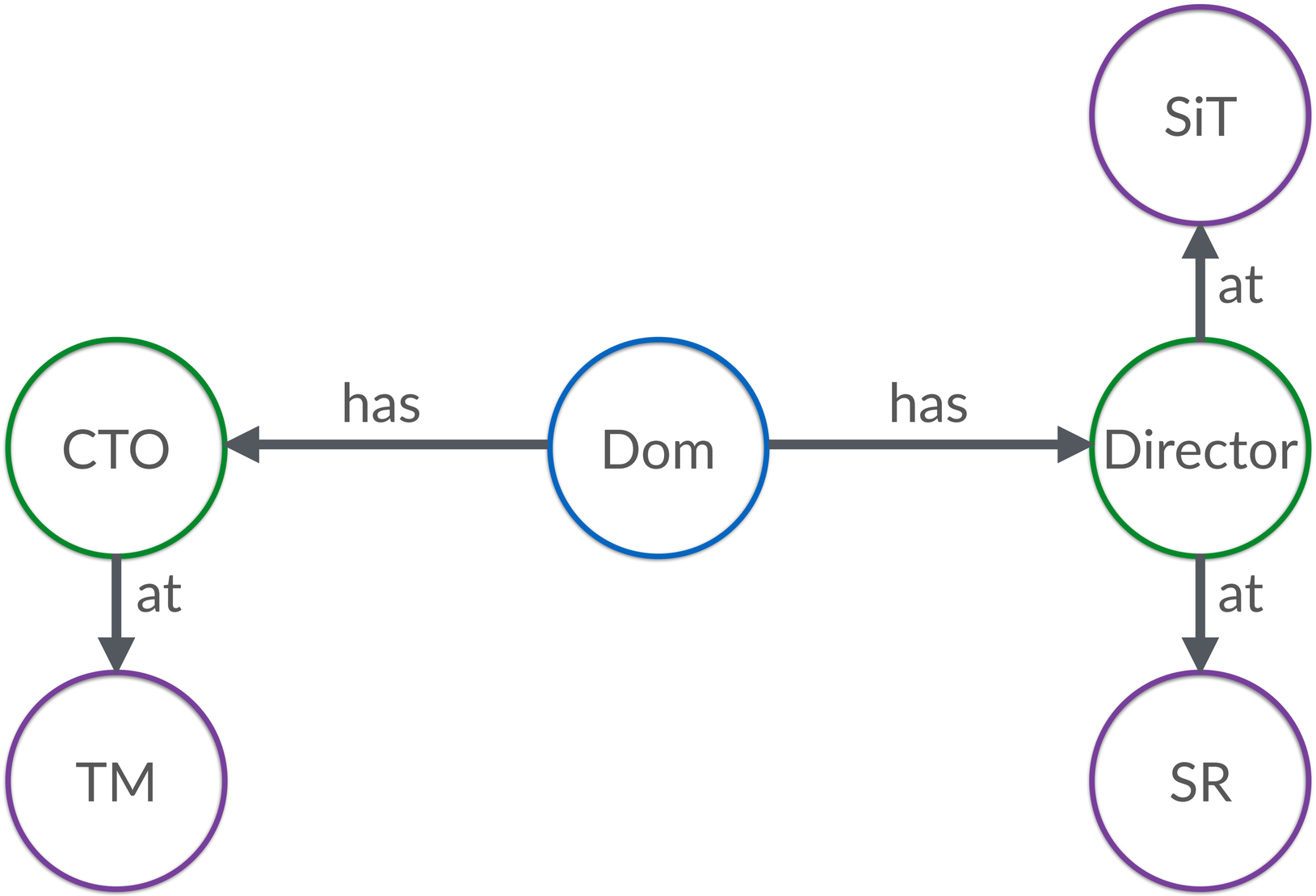
Dom’s first Law

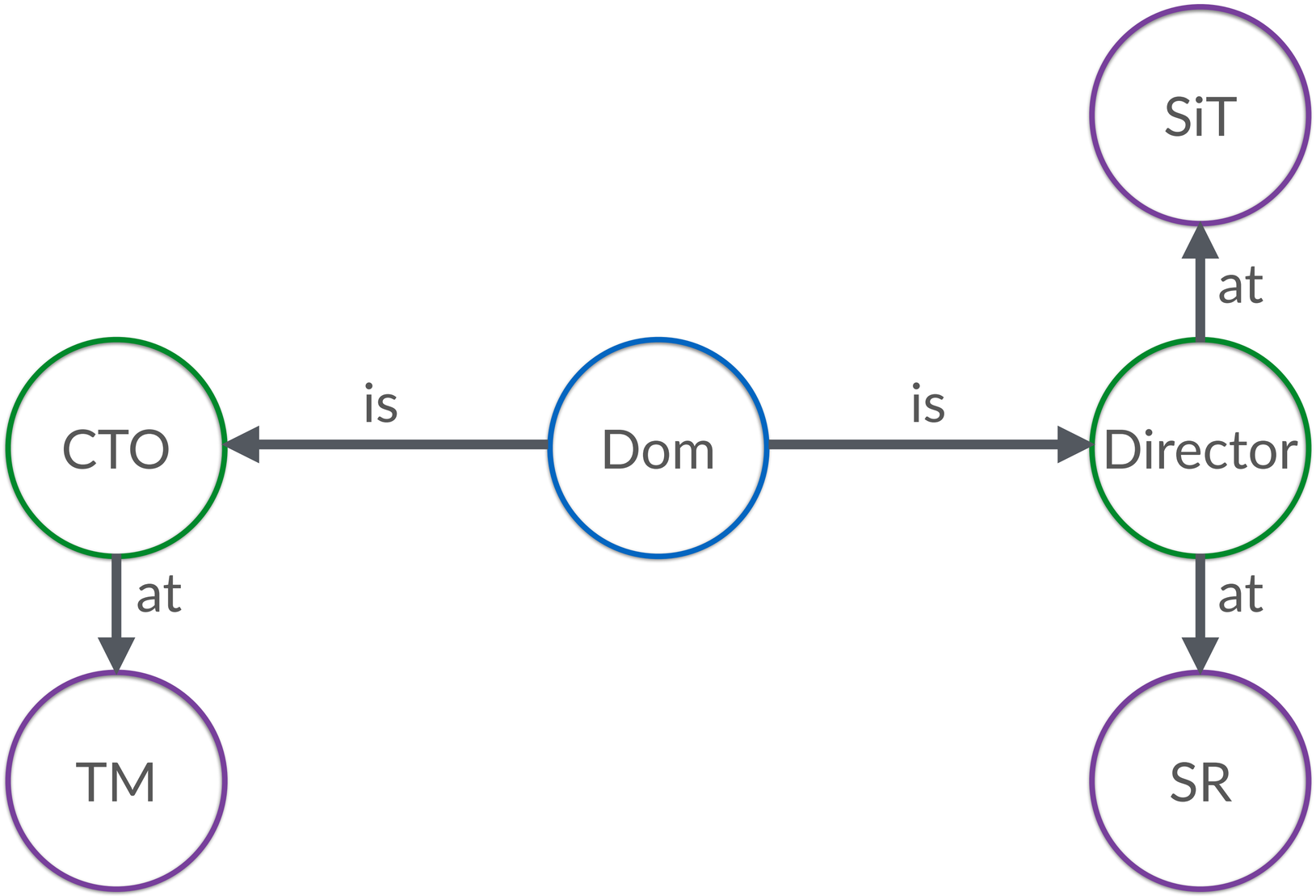
https://github.com/domdavis/accu

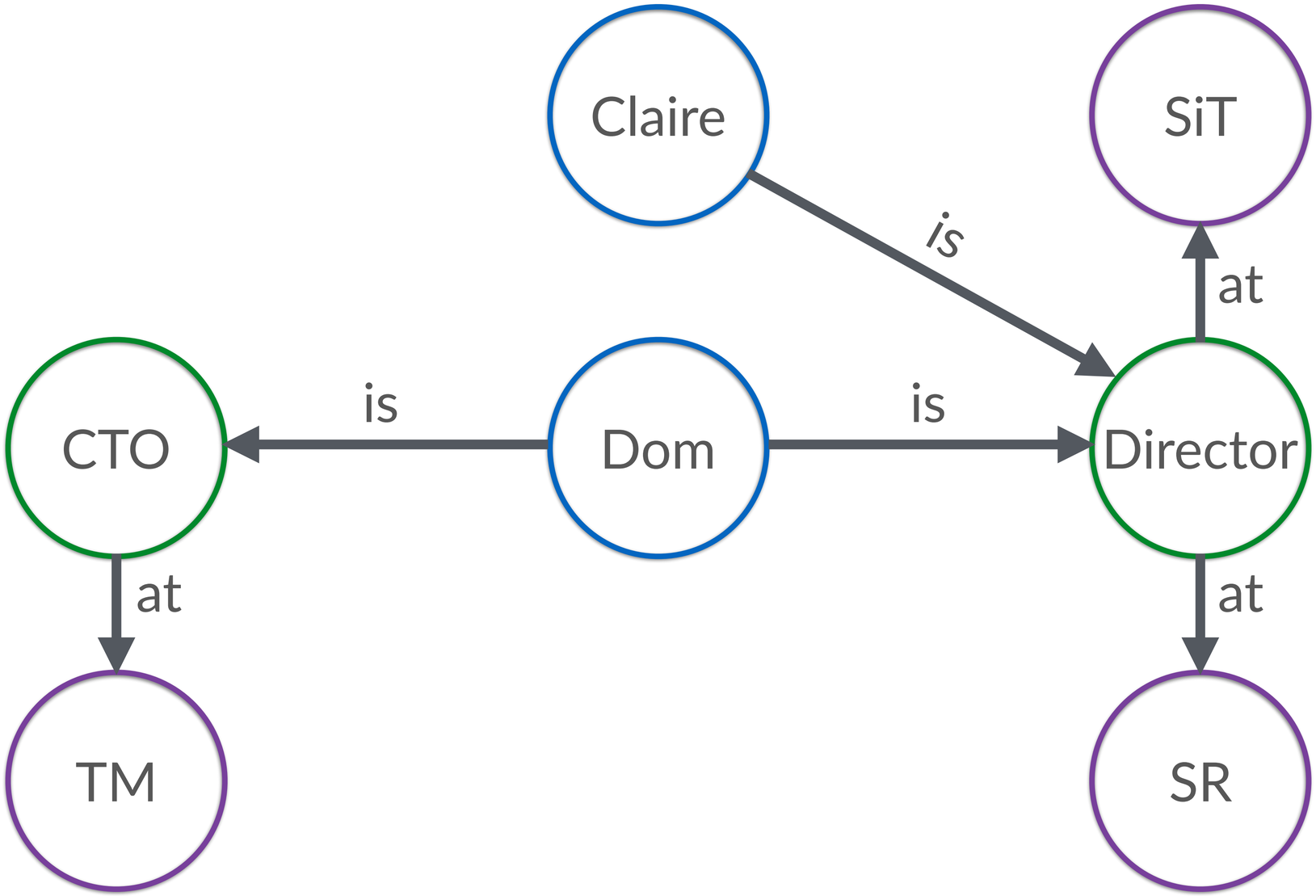
MODELLING

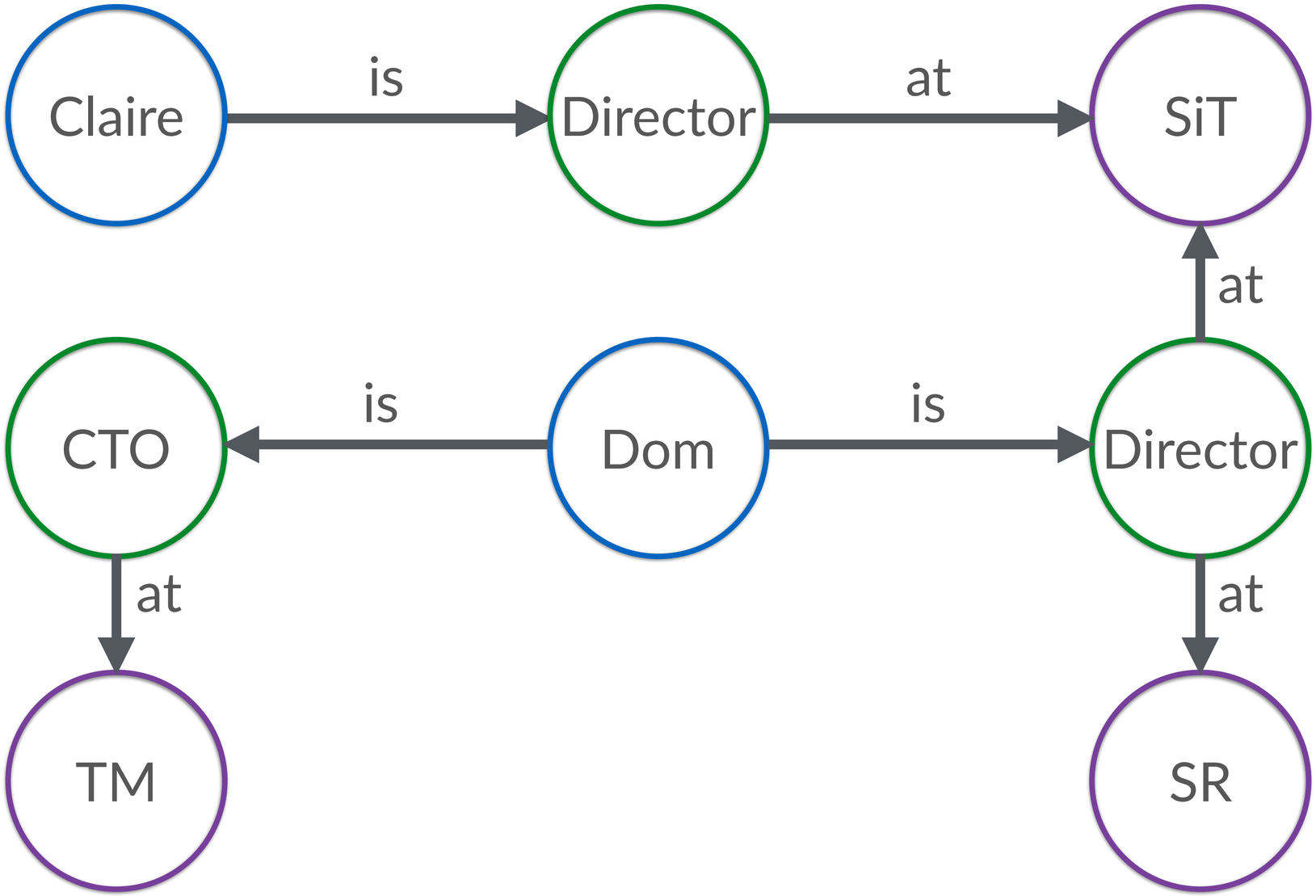
“A person has a position at a company.”

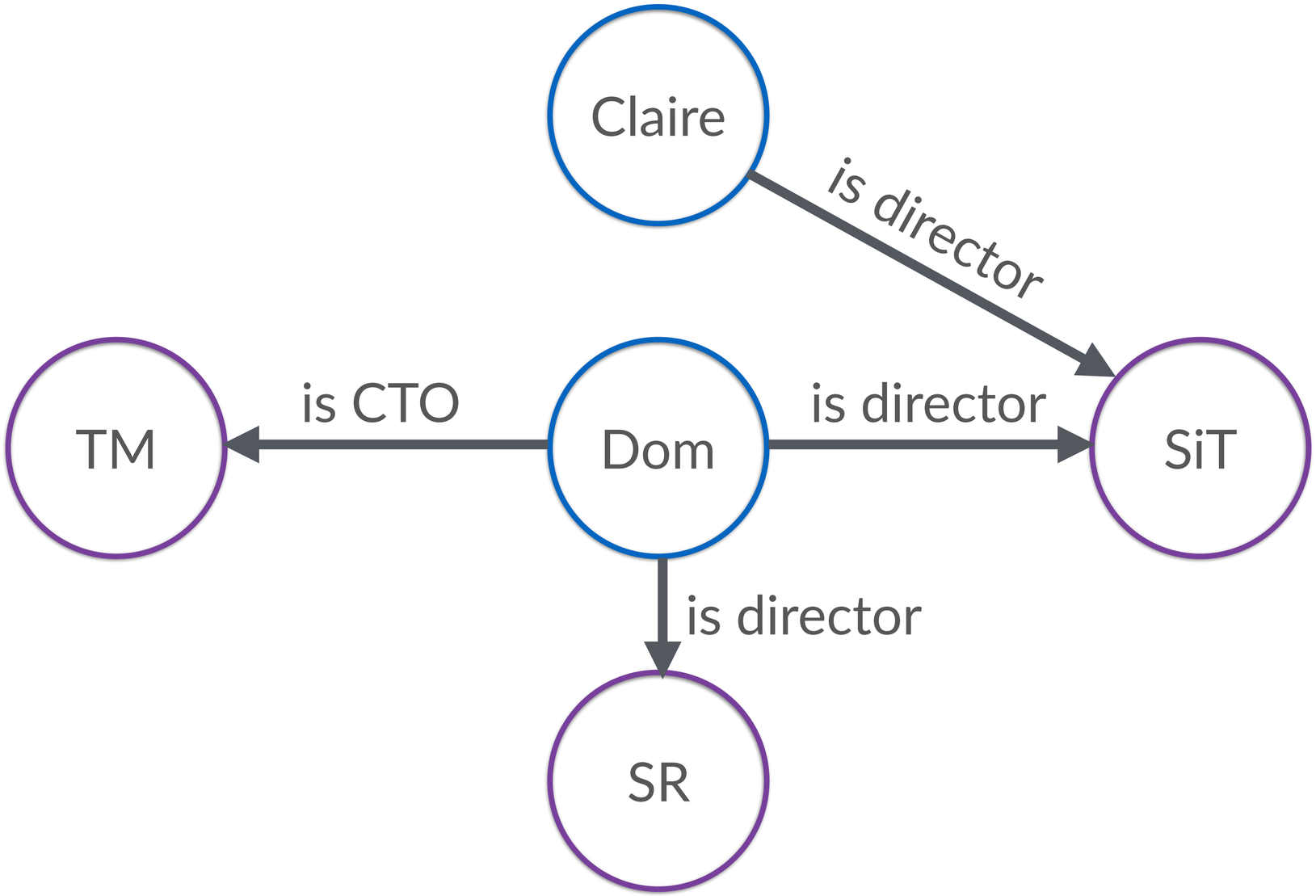

Person
has
Position
at
Company

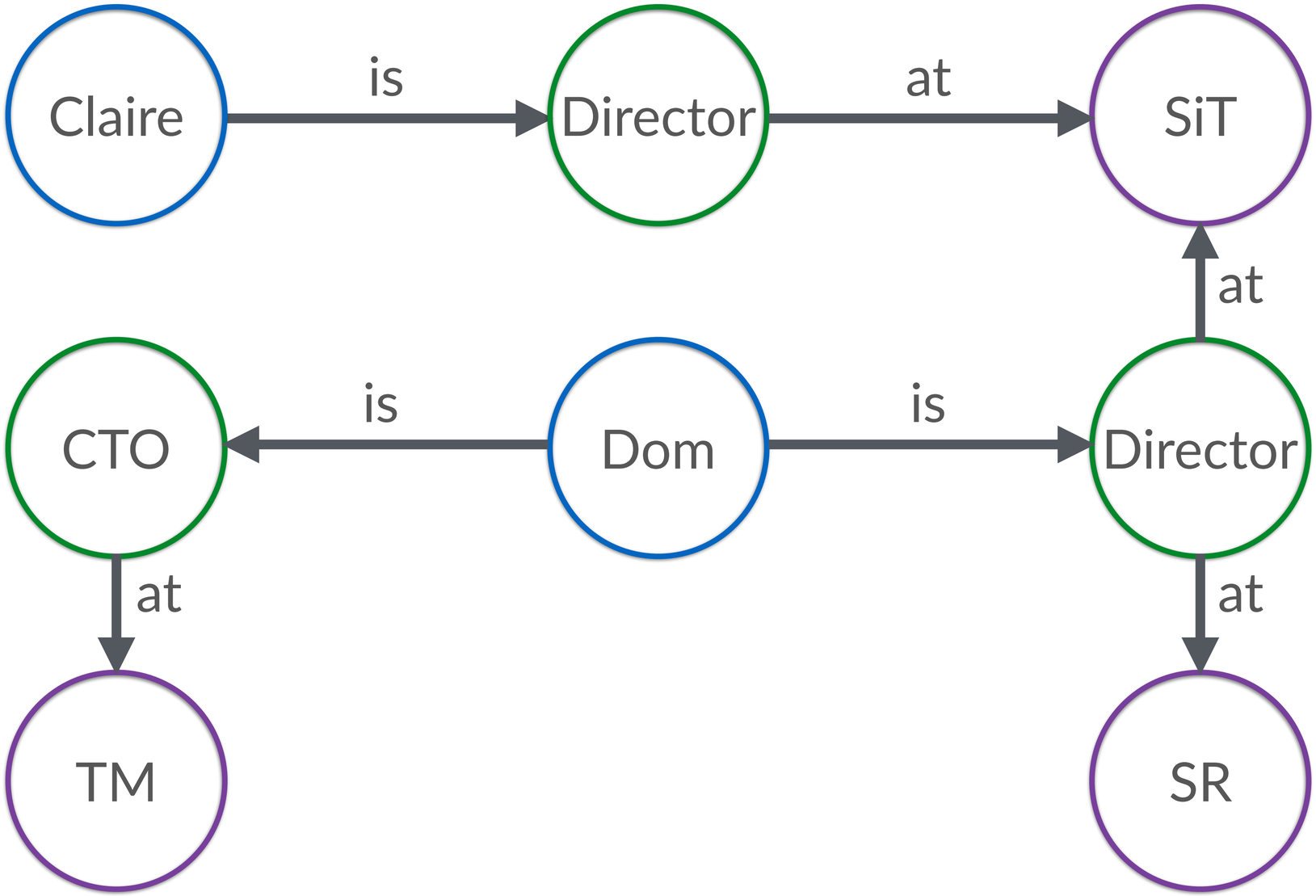

SiT
at
CTO
has
Dom
has
Director
at
at
TM
SR

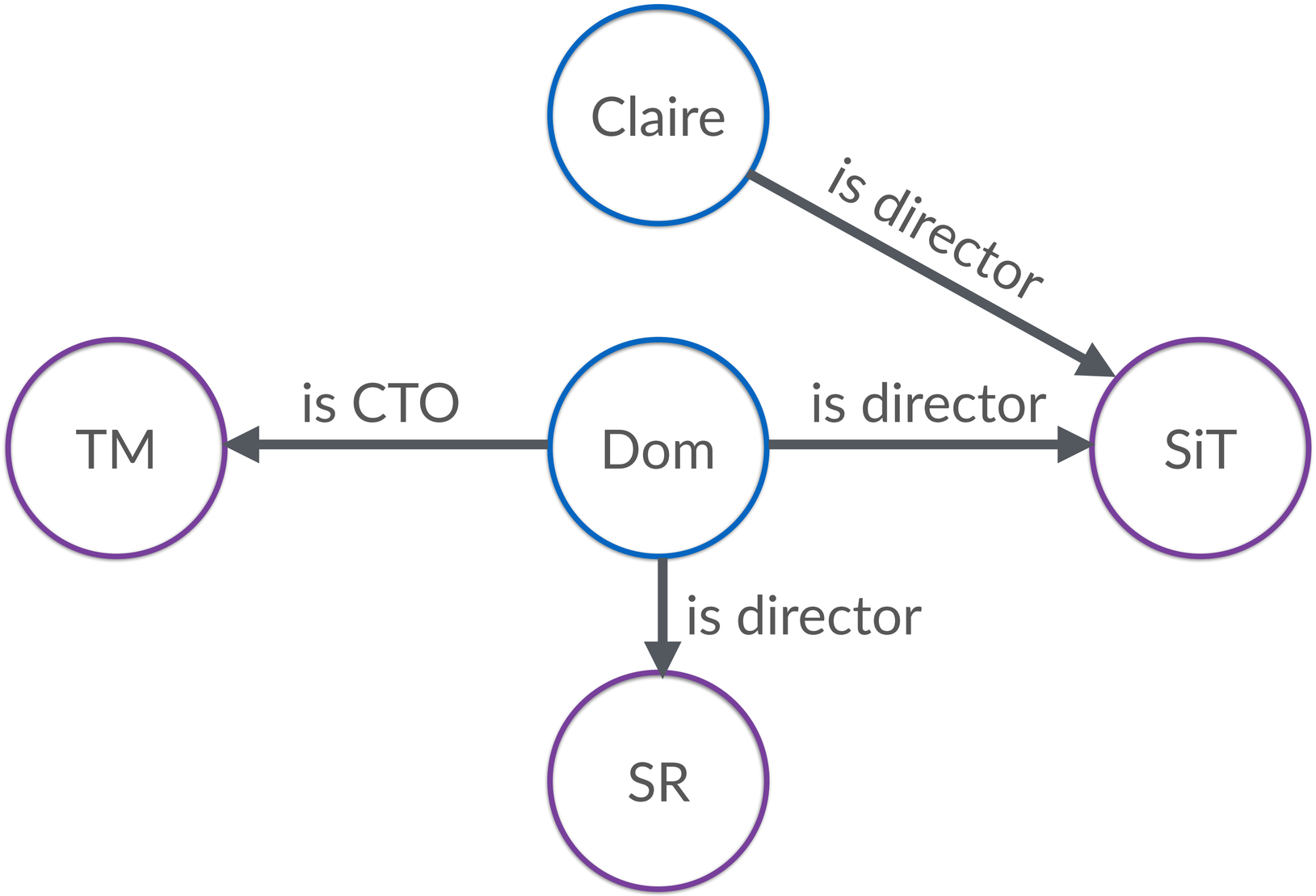

SiT
at
CTO
is
Dom
is
Director
at
at
TM
SR

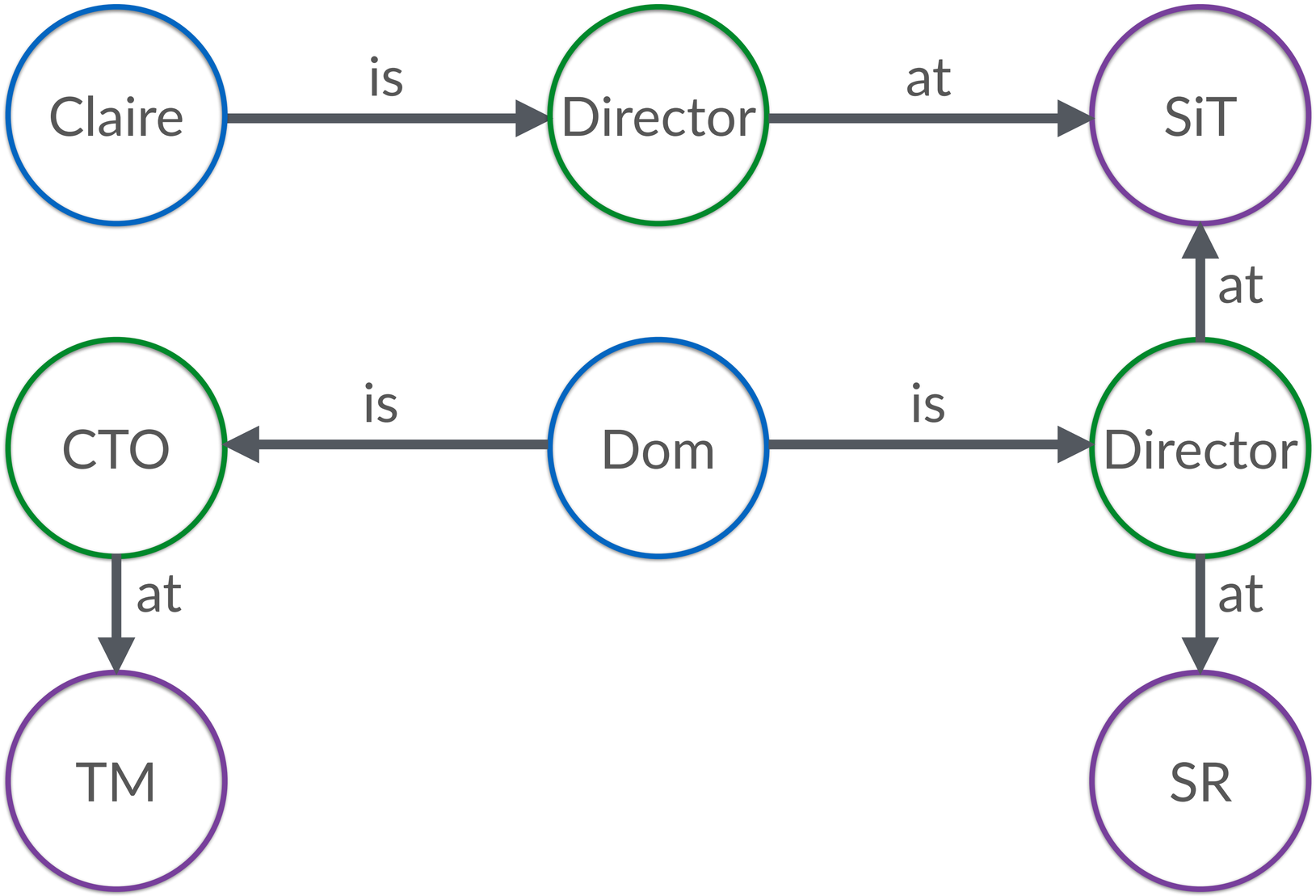

Claire
SiT
is
at
CTO
is
Dom
is
Director
at
at
TM
SR

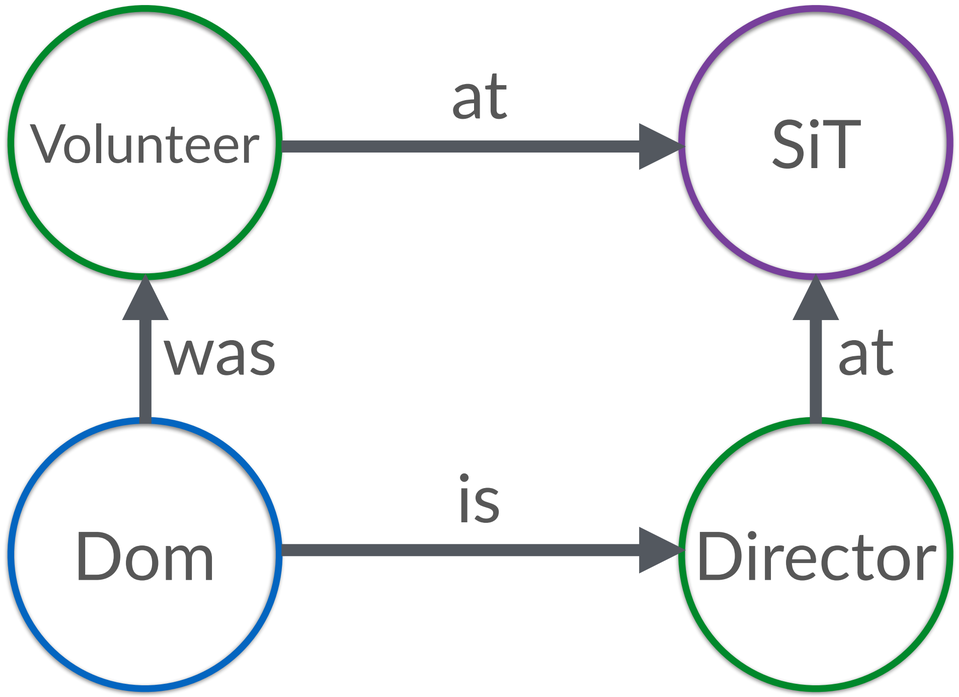

Claire
is
Director
at
SiT
at
CTO
is
Dom
is
Director
at
at
TM
SR

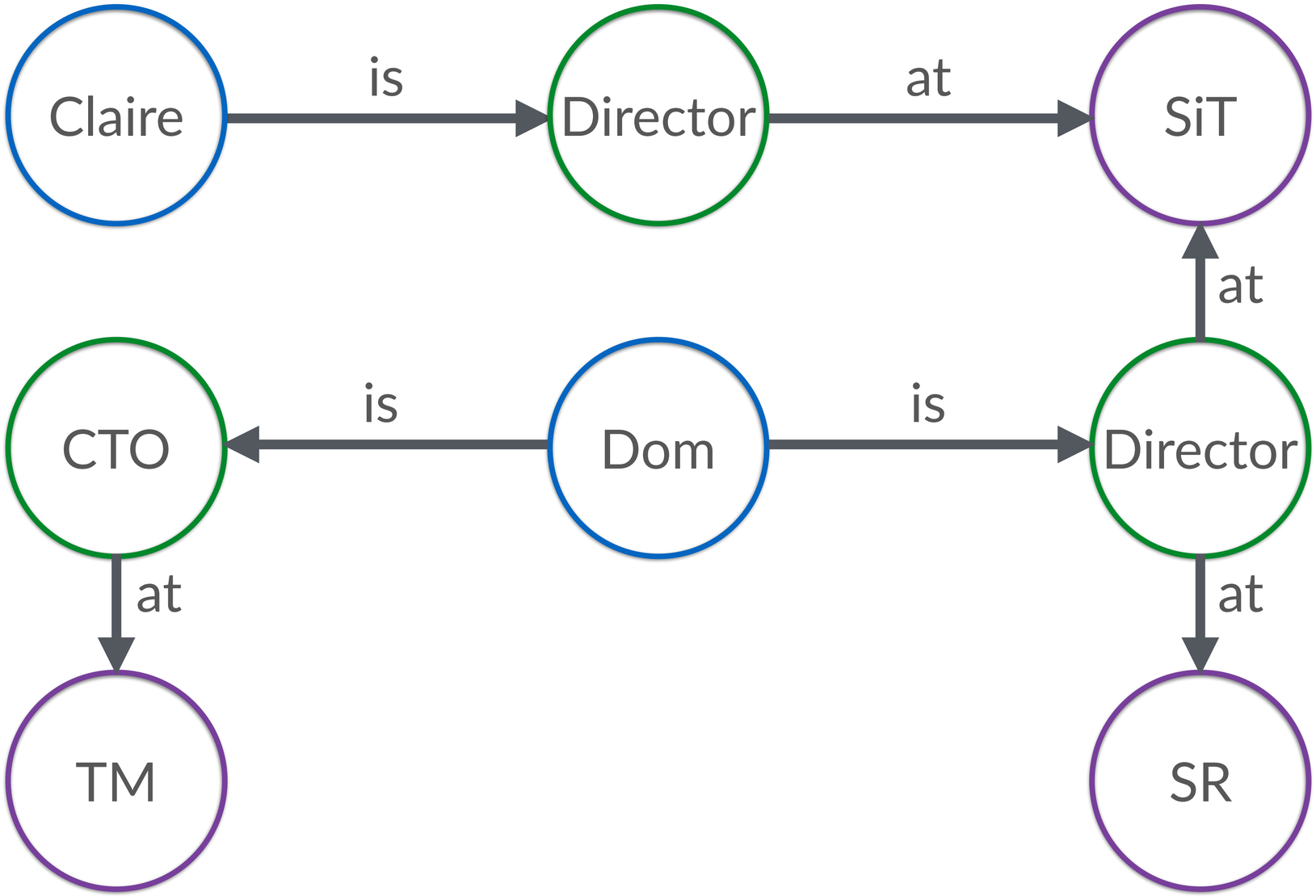

Claire
is director
TM
is CTO
Dom
is director
SiT
is director
SR


Claire
is
Director
at
SiT
at
CTO
is
Dom
is
Director
at
at
TM
SR


Claire
is director
TM
is CTO
Dom
is director
SiT
is director
SR


Claire
is
Director
at
SiT
at
CTO
is
Dom
is
Director
at
at
TM
SR


Volunteer
at
SiT
was
at
Dom
is
Director


Claire
is
Director
at
SiT
at
CTO
is
Dom
is
Director
at
at
TM
SR

```
(:Person {name: "Dom Davis"})
```

```
(:Person)-[:HAS_NAME]->(:`Dom Davis`)
```

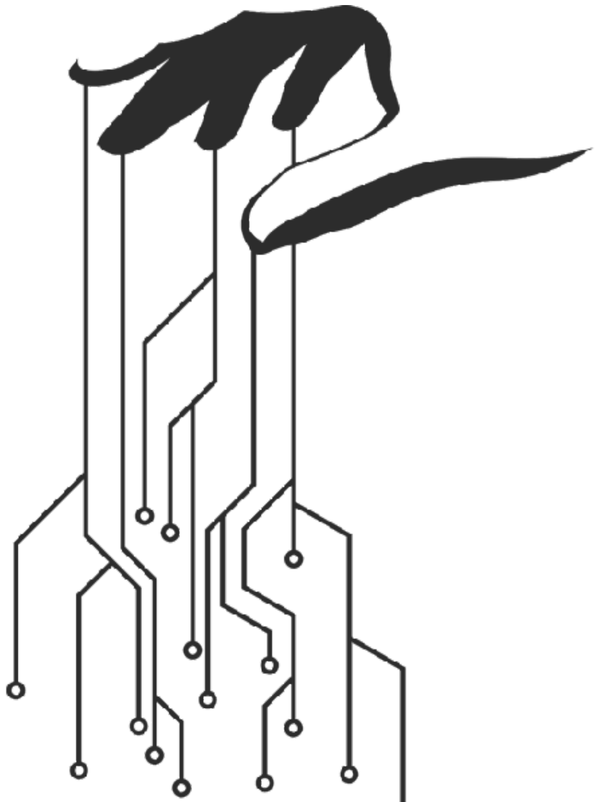
```
(:Person {name: "Dom Davis"})
          -[:HAS_ROLE {type: "Primary"}]->(:Role {title: "CTO"})
          -[:IN_COMPANY]->(:Company {name: "Tech Marionette"})
```

```
-[:HAS_ROLE {type: "Primary"}]->
```

```
(:Person {name: "Dom Davis"})
          -[:HAS_PRIMARY_ROLE]->(:Role {title: "CTO"})
          -[:IN_COMPANY]->(:Company {name: "Tech Marionette"})
```

```
(r:Role {title: "CTO"}),
(:Person {name: "Dom Davis"})-[:HAS_ROLE]->(r)
          -[:IN_COMPANY]->(:Company {name: "Tech Marionette"}),
(r)-[:TYPE]->(:Primary)
```

Drive the model from the language of the domain

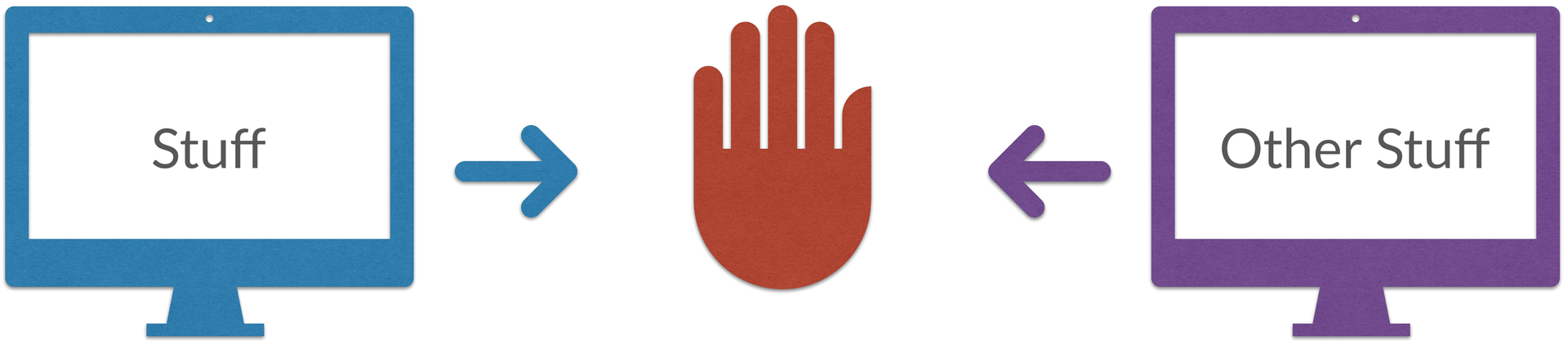


TECH MARIONETTE


Stuff

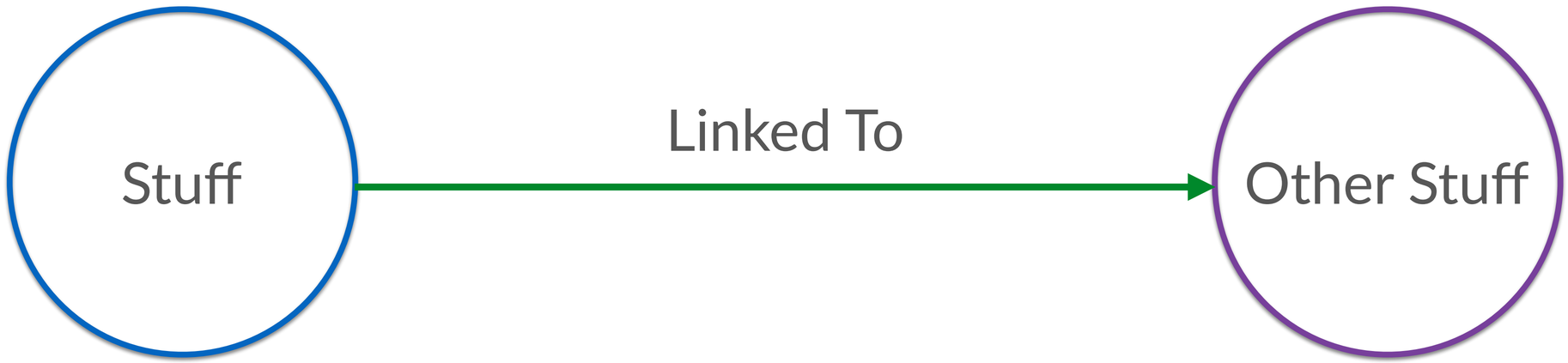

Stuff


Other Stuff


Stuff
Other Stuff


Stuff
Insight
Other Stuff


Stuff
Other Stuff


Stuff
Linked To
Other Stuff

```
(:Stuff {
  property1: "some value",
  // : : :
  propertyN: "some other value”
})
```

```
(:Concept {
  properties: [“A”, "B", "C"]
})
```

Stuff has properties

```
(:Stuff)-[:HAS]->(p:Property)
SET p.Name = “A”, p.Value = “foo”
```

```
(:Stuff)-[:ALIAS]->(:Property)
```

```
(:Thing)-[:ALIAS]->(:Thing)
```

```
(s)-[:ALIAS {name: "Dom"}]->(o),
(s)-[:ALIAS {name: "Dominic"}]->(o),
(s)-[:ALIAS {name: “@idomdavis”}]->(o)
```

(g:Graph)-[:DESCRIBED_BY]->(g)

NO SQL

Dom Davis

@idomdavis

about.me/idomdavis