UNIVERSITÄT LEIPZIG

European Research Council

Semantics in Space: Traditional semantic maps and Graph Theory

Natalia Levshina © 2017

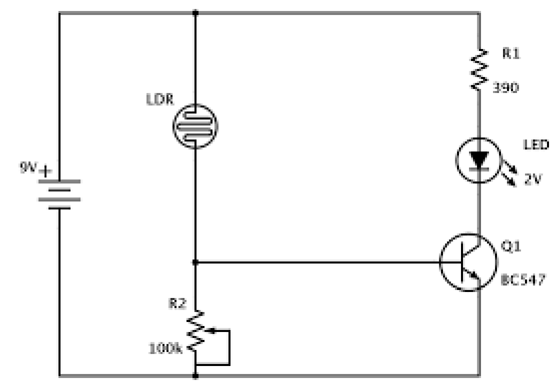
Semantic maps

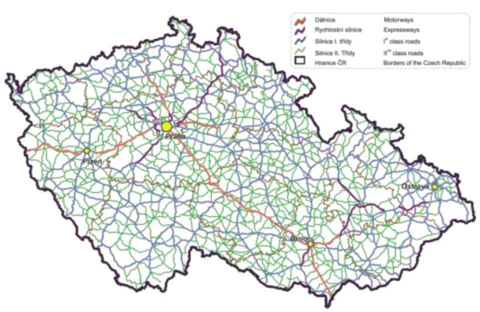
- Represent a convenient tool for comparison of semantic and pragmatic functions across languages
- Based on different kinds of data: grammars and typological databases, parallel corpora and experimental data

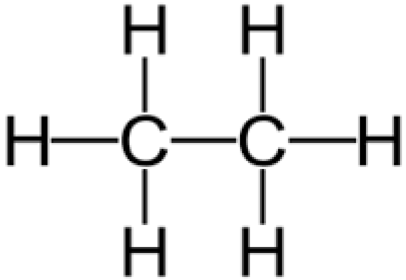
Main types of semantic maps

- Connectivity maps, with meanings as nodes (**vertices**) and links (**edges**) between them. Such maps are called **graphs**.
  - Undirected vs. directed (e.g. diachrony)
  - Weighted vs. unweighted
- Probabilistic maps, with distances between objects (e.g. examples from a parallel corpus or stimuli in an experiment)

Graphs are everywhere!

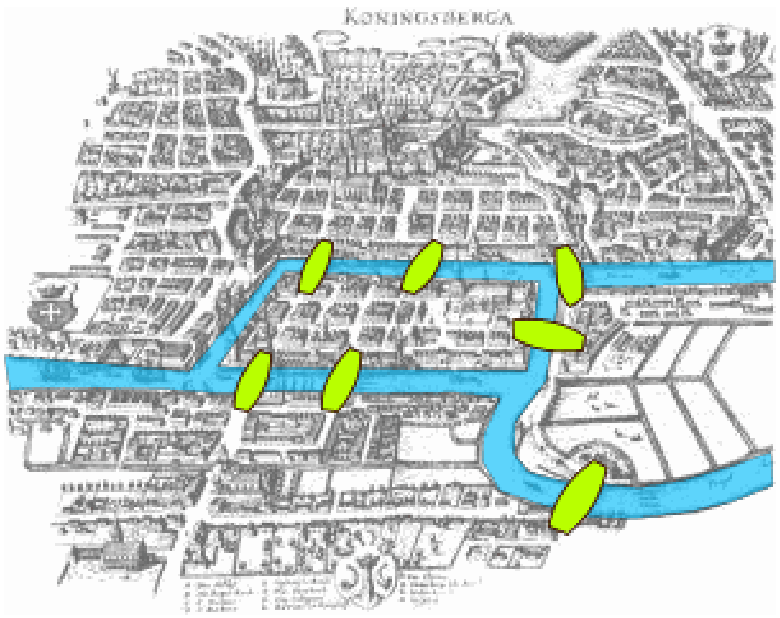






Seven bridges of Königsberg

- Devise a walk through the city that would cross each of the bridges once, and only once.



Graph Theory

- Leonhard Euler (1707 - 1783)

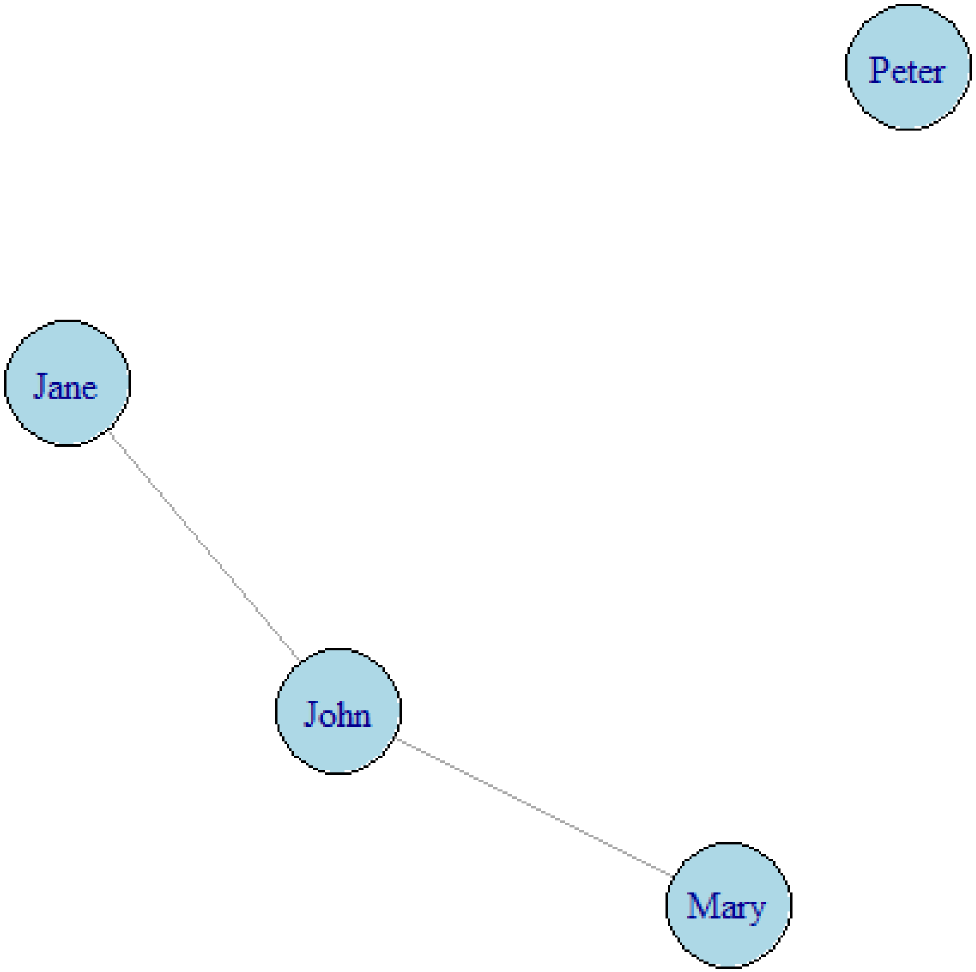
Main types of graphs

- Undirected
- Directed

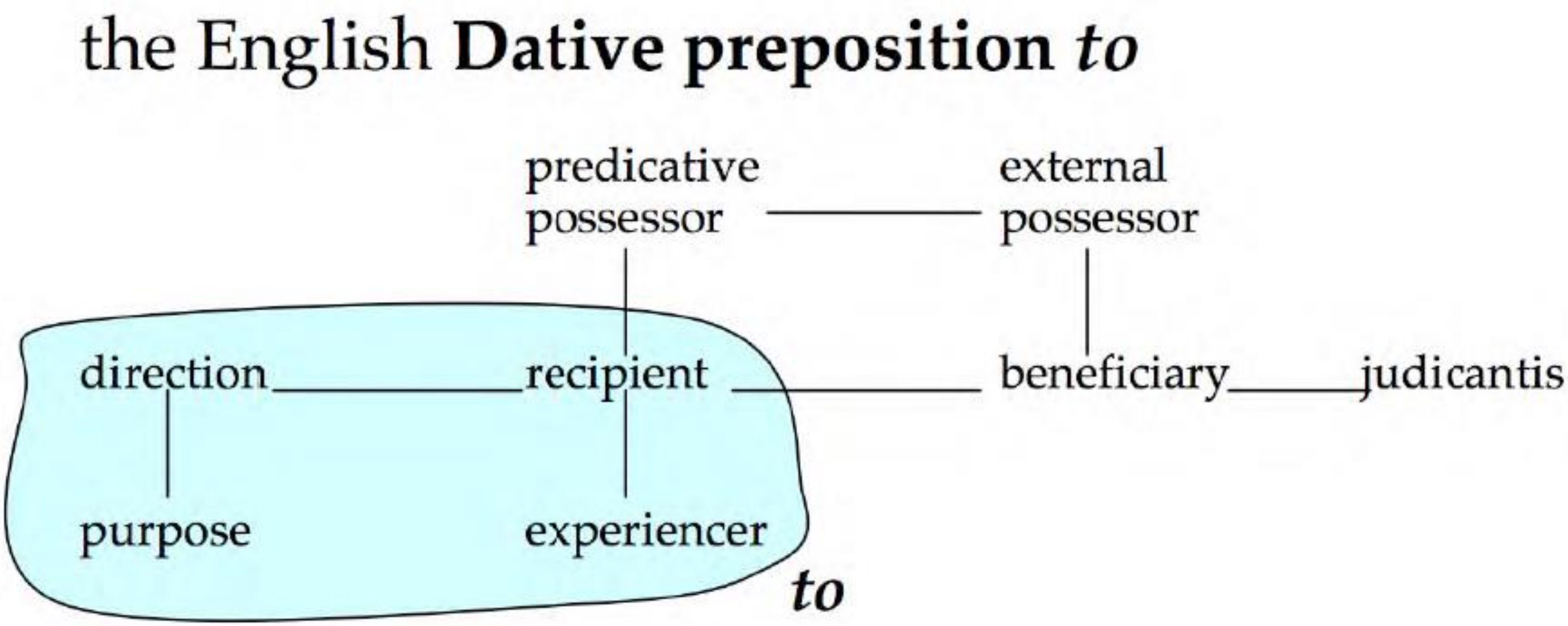
- Unweighted
- Weighted

Undirected graphs: marriage

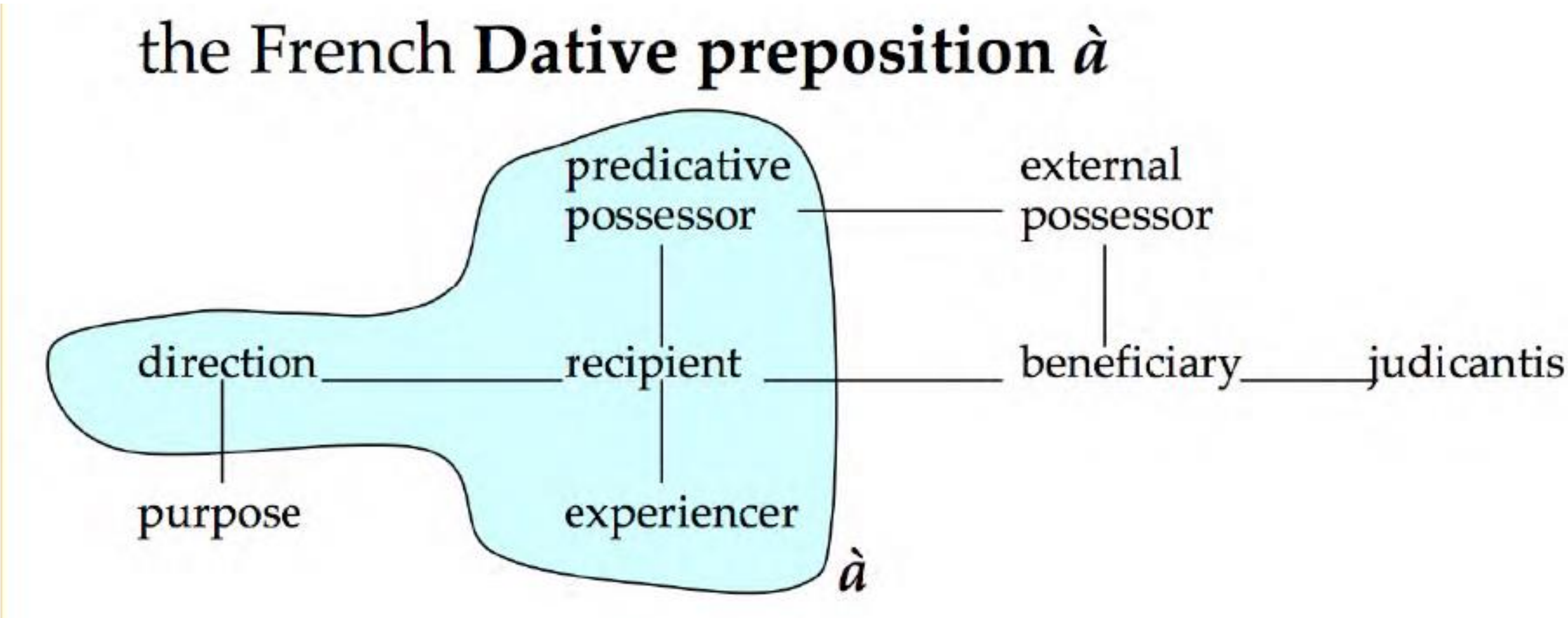
- Mary is John's ex-wife.
- John and Jane are married.
- Peter is single.



Haspelmath 2003: Datives



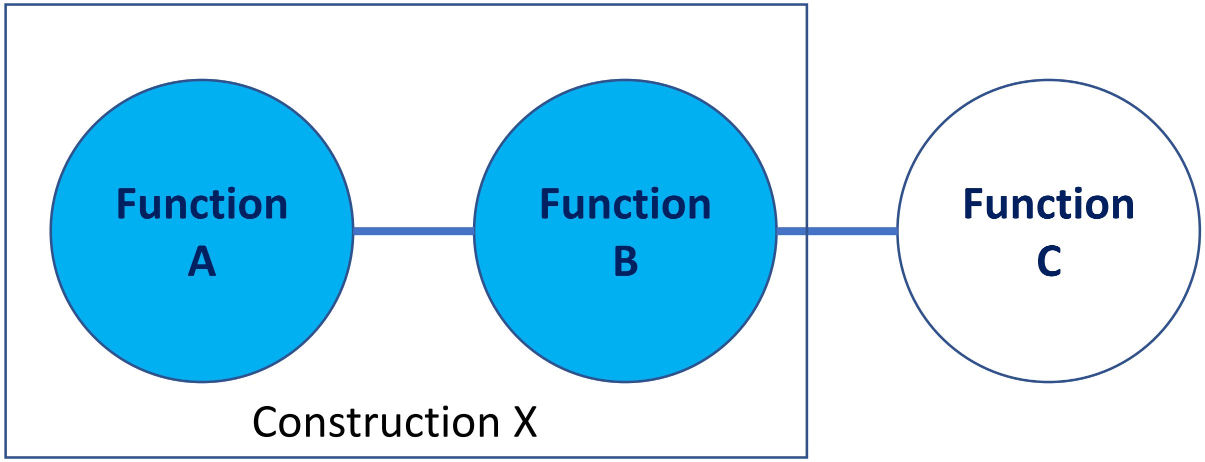
Haspelmath 2003: Datives



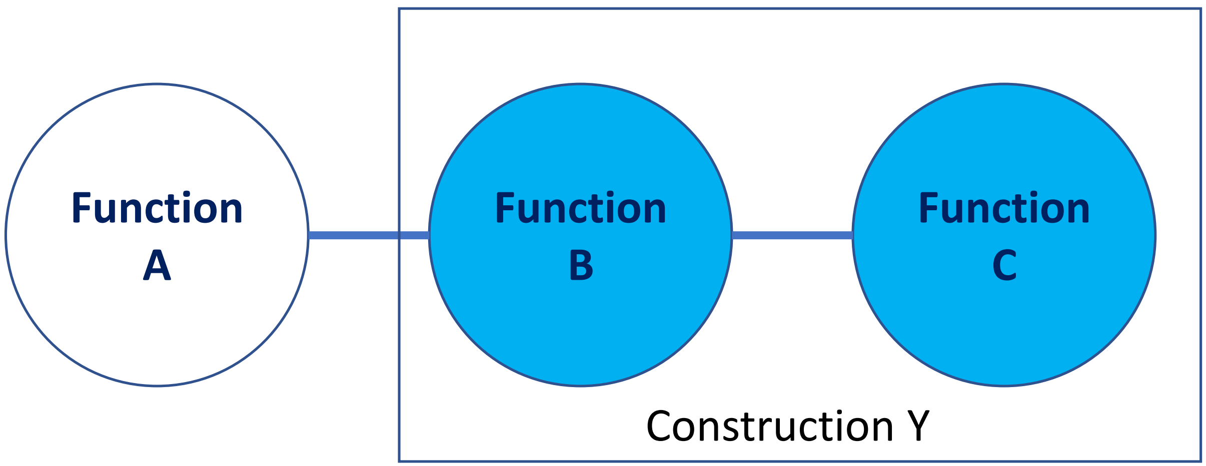
Main principles of semantic maps

- Nodes: A function is put on a map when there's at least one pair of languages which differ wrt. this function (Haspelmath 2003)
- Links: the principle of connectivity (adjacency/contiguity):
  - if a construction has more than one function, they should be connected (see van der Auwera 2013)

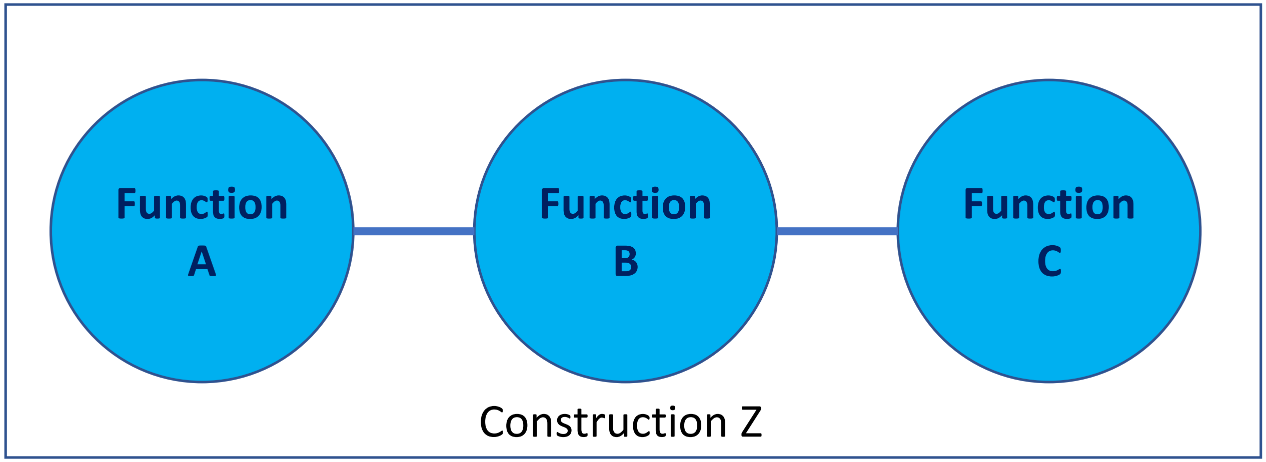
Example 1



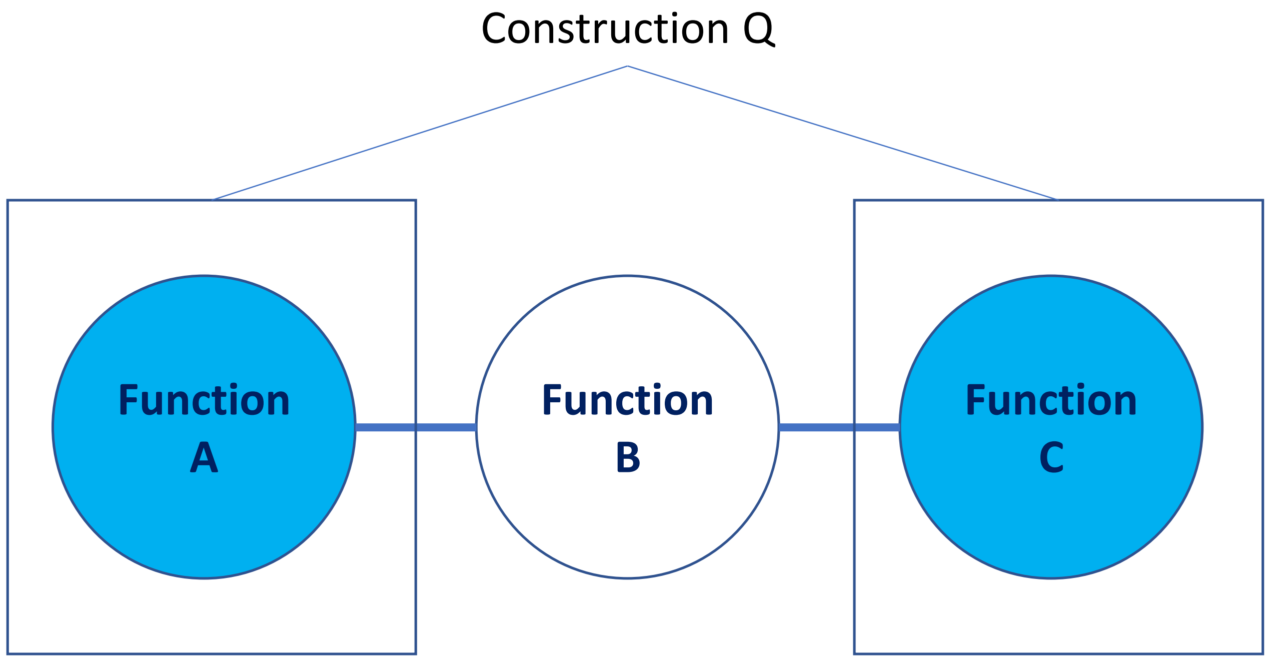
Example 2



Example 3

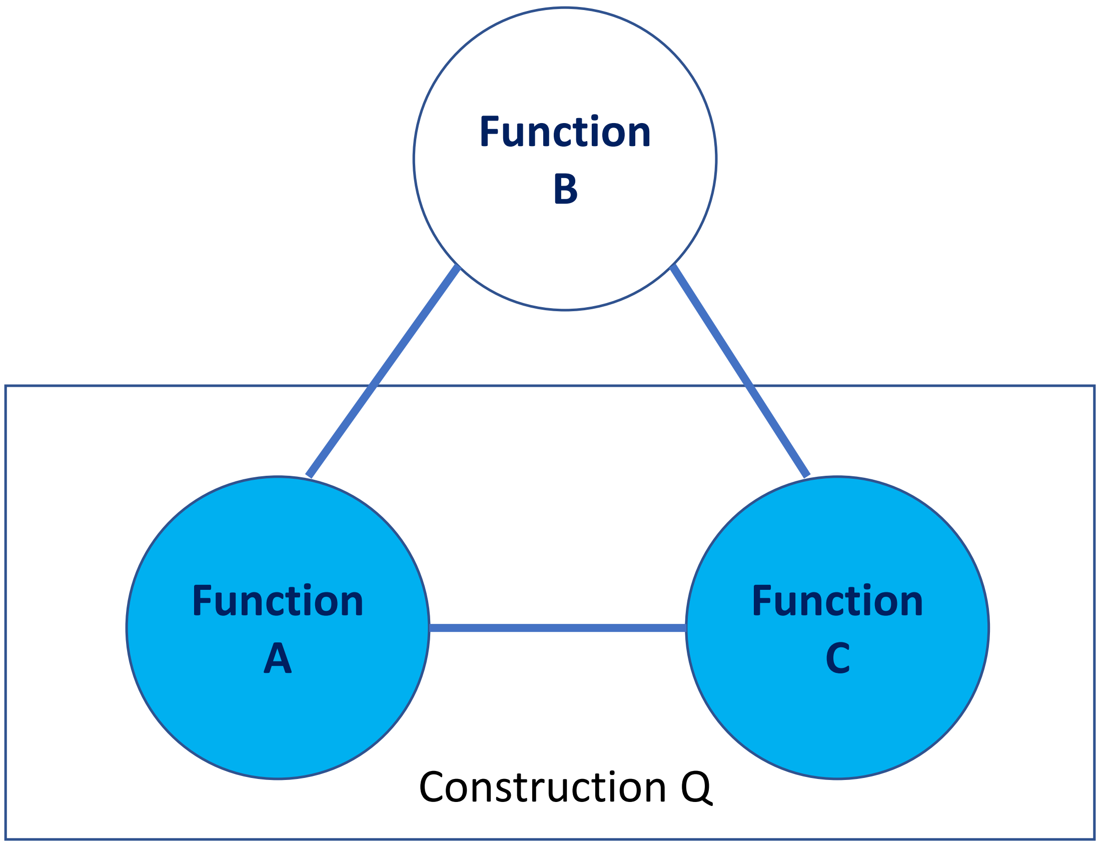


Example 4



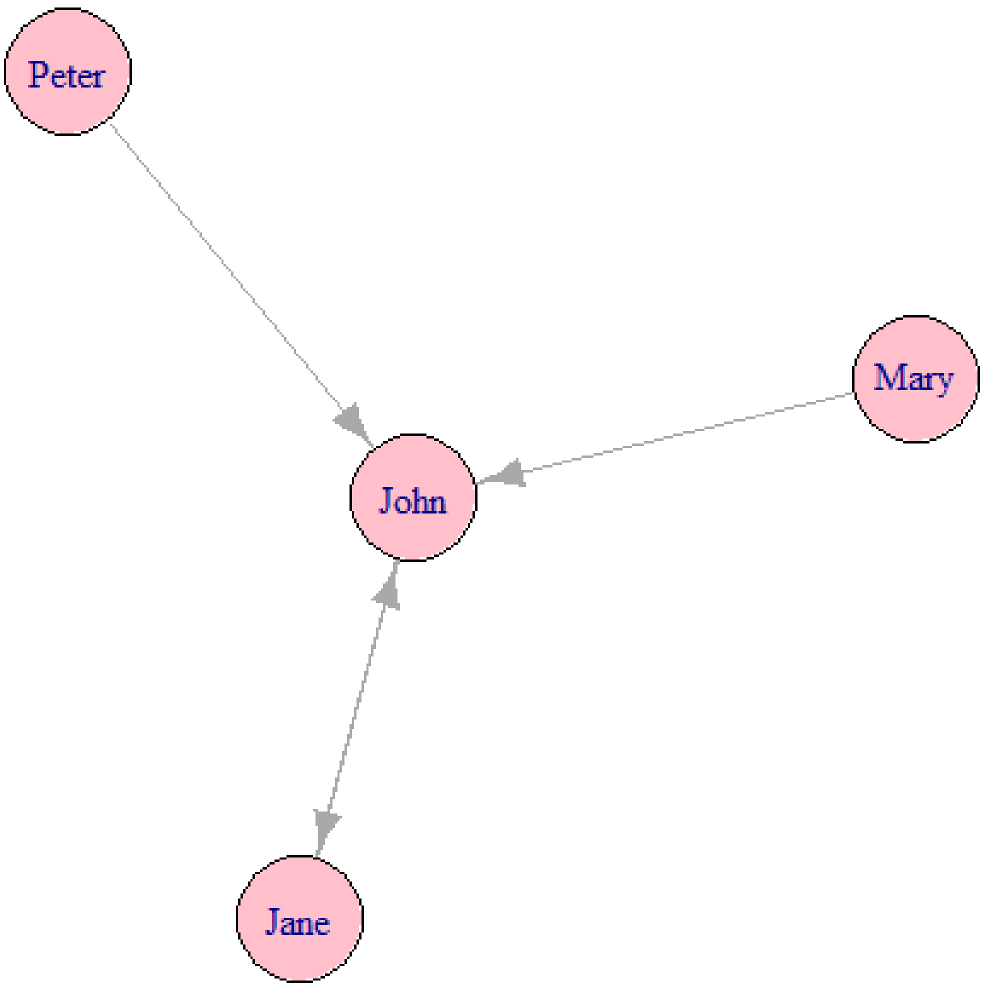
Wrong:
the connectivity principle is not observed!

A fix

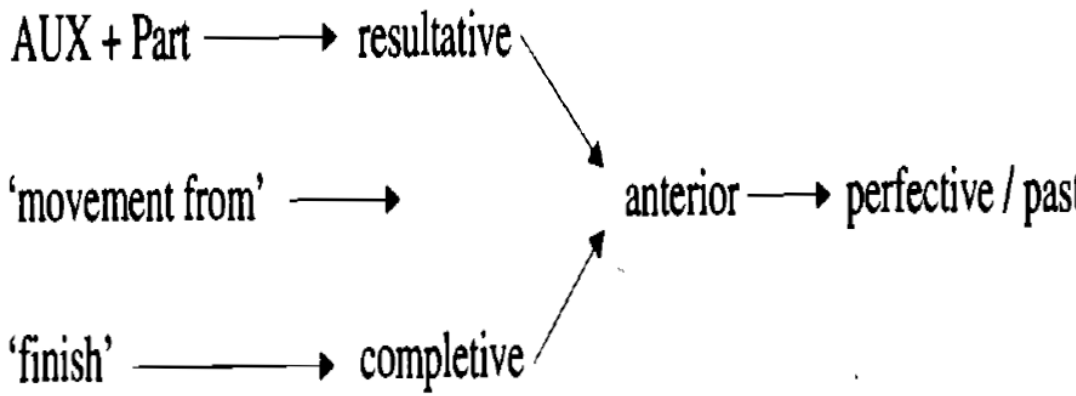


Directed graphs: feelings

- Mary loves John.
- John loves Jane.
- Jane loves John.
- And Peter fancies John, too.

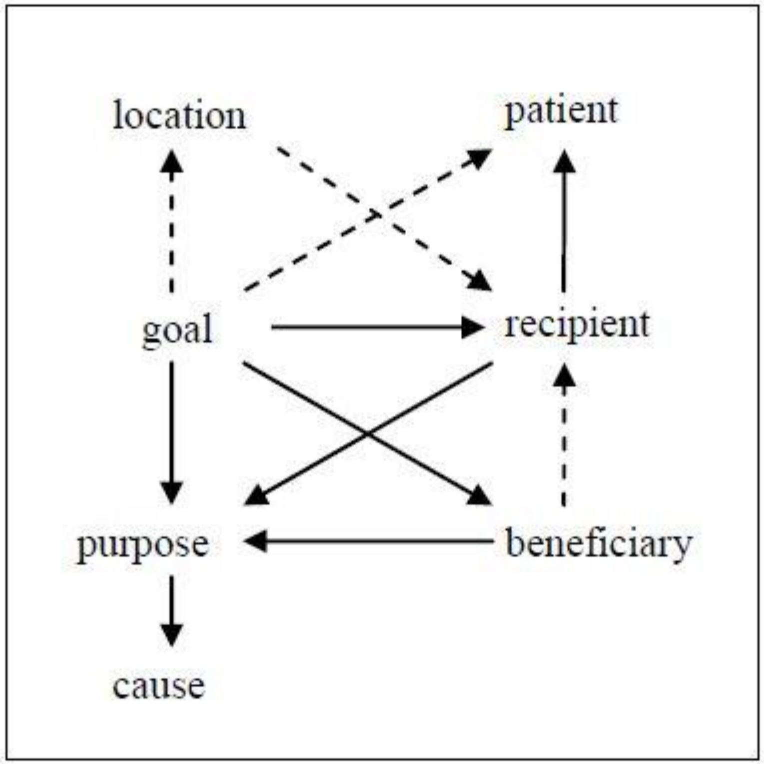


Tense and aspect grams



Bybee et al. (1994)

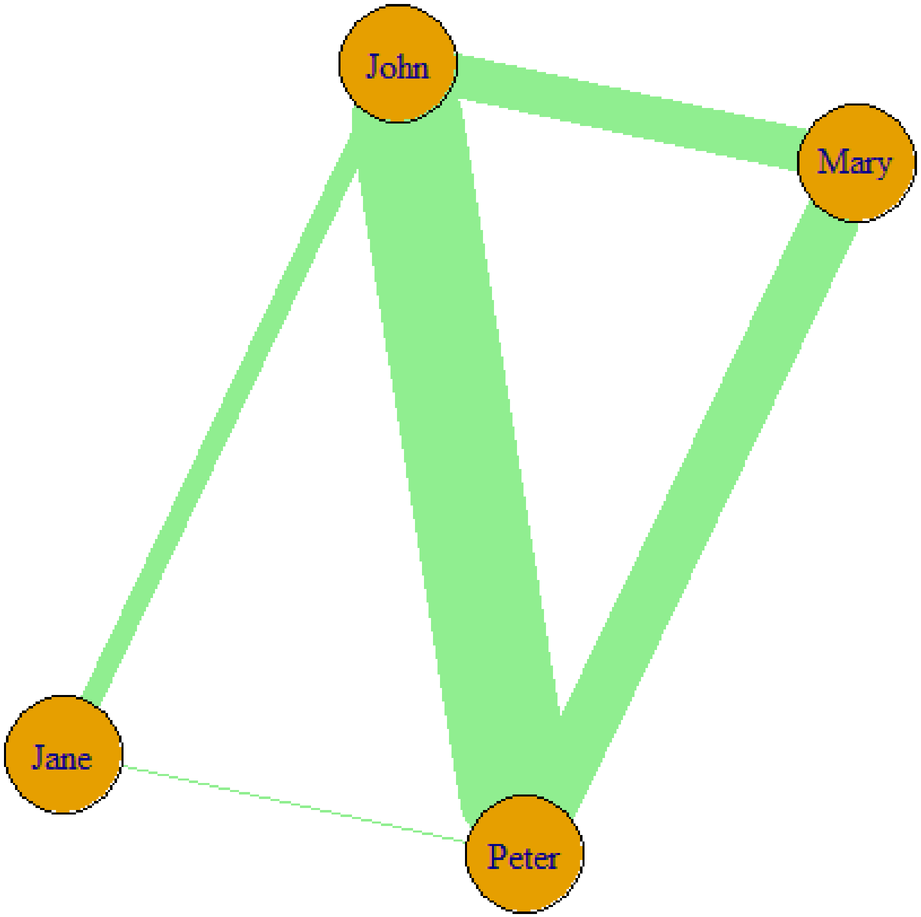
Goal-Recipient Domain



Narrog (2010)

Weighted graphs: number of pizzas eaten together

- Mary and John have eaten 20 pizzas.
- John and Jane have eaten 10 pizzas.
- Peter and John have eaten 50 pizzas.
- Mary and Peter have eaten 25 pizzas.
- Jane and Mary have eaten 0 pizzas.
- Jane and Peter have eaten 1 pizza.

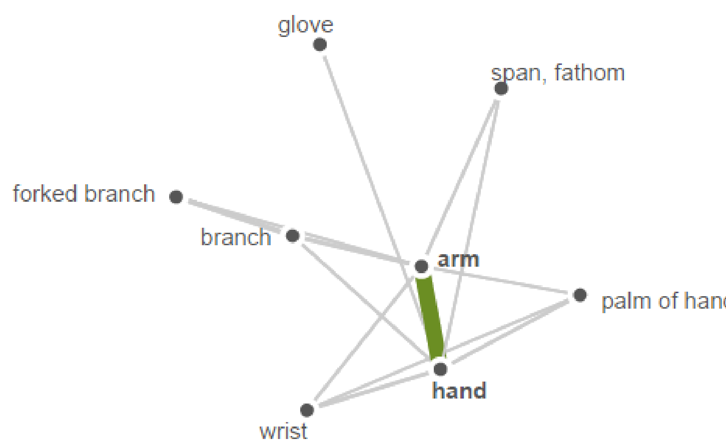


CLiCS: colexification patterns of HAND

Line opacity: **Line weights:** **Coloring:** Family

85 links for "arm" and "hand":

Language	Family	Form
1. Gawwada	Afro-Asiatic	harɗo
2. Hausa	Afro-Asiatic	hannu
3. Iraqw	Afro-Asiatic	dawa
4. Polci	Afro-Asiatic	aam
5. Tarifit	Afro-Asiatic	fus
6. Azerbaijani, South	Altaic	әл
7. Kumyk	Altaic	къол
8. Oroqen	Altaic	ŋaːla
9. Gurinji	Australian	wartarn
10. Vietnamese	Austro-Asiatic	tay



List et al. (2014) http://clics.lingpy.org/

Case study

Colexification patterns of FOOT

Different methods

- From an adjacency matrix with colexification frequencies
- From a data frame with the list of edges and their weights (colexification frequencies)

Different methods

- From an adjacency matrix with colexification frequencies
- From a data frame with the list of edges and their weights (colexification frequencies)

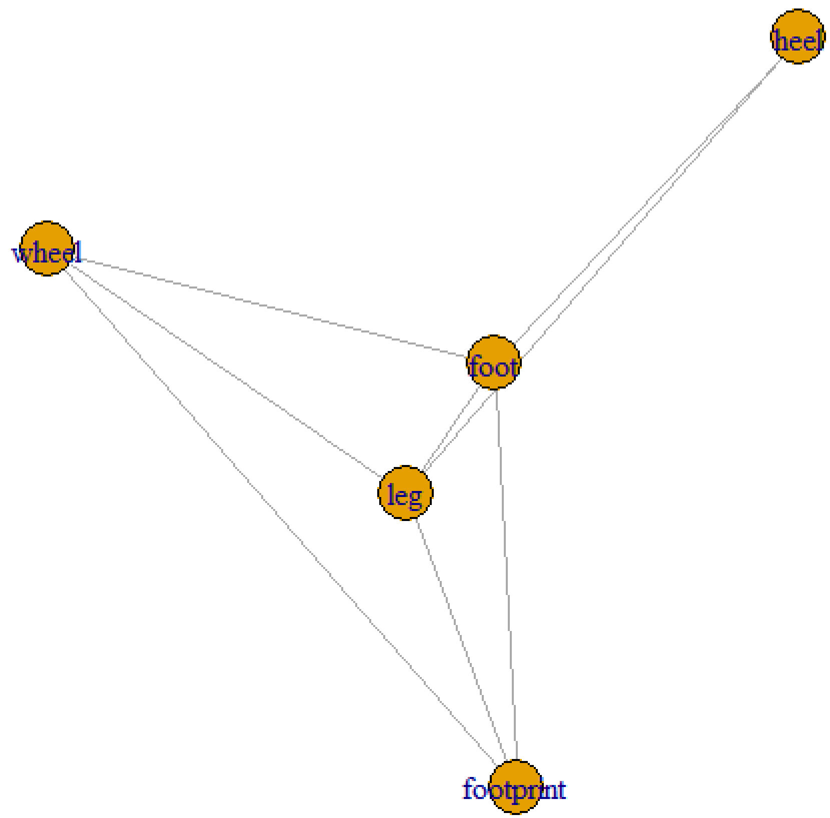
Adjacency matrix based on CLiCS

```
> foot_am
          foot leg footprint wheel heel
foot         0  71        13     6    7
leg         71   0         7     1    2
footprint   13   7         0     1    0
wheel        6   1         1     0    0
heel         7   2         0     0    0
```

- This matrix can be regarded as an adjacency matrix if we assume that colexification frequencies represent conceptual proximity.

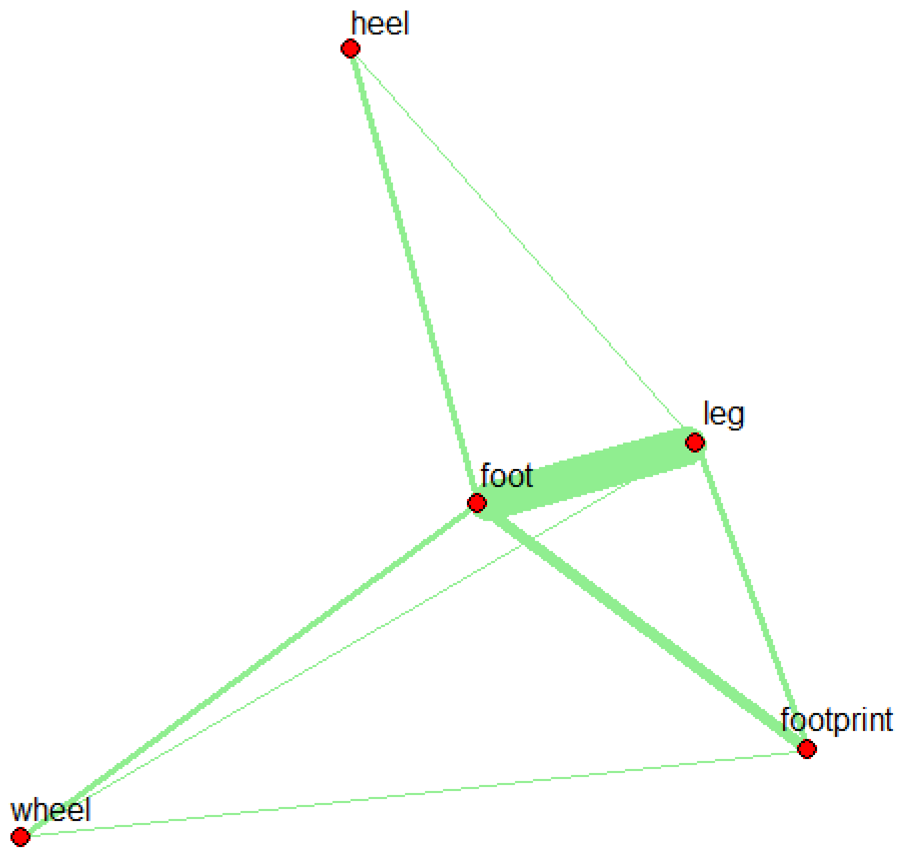
Make a graph from an adjacency matrix

```
> library(igraph)

> foot_graph <-
graph_from_adjacency_matrix(foot_am, mode =
"undirected", weighted = TRUE)

> plot(foot_graph)
```


heel
wheel
foot
leg
footprint

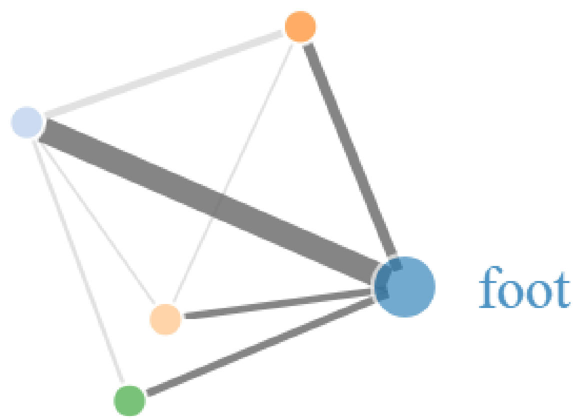
Add weights and make prettier

```
> plot(foot_graph, edge.width =
E(foot_graph)$weight^0.7, edge.color =
"lightgreen", vertex.label.cex = 1, vertex.size =
5, vertex.label.color = "black",
vertex.label.family = "sans", vertex.color =
"red", vertex.label.dist = 1.5)
```


heel
leg
foot
footprint
wheel

D3 graphs from igraph graphs

```
> library(networkD3)

> foot_D3 <- igraph_to_networkD3(foot_graph)

> forceNetwork(Links = foot_D3$links, Nodes =
foot_D3$nodes, NodeID = 'name', Group = 'name',
opacity = 0.8, Value = "value", linkDistance =
100, fontSize = 20)
```


foot

Different methods

- From an adjacency matrix with colexification frequencies
- From a data frame with the list of edges and their weights (colexification frequencies)

Data frame with edges and weights

```
> foot_df
       from         to weight
1      foot        leg     71
2      foot  footprint     13
3      foot      wheel      6
4      foot       heel      7
5       leg  footprint      7
6       leg      wheel      1
7       leg       heel      2
8 footprint      wheel      1
```

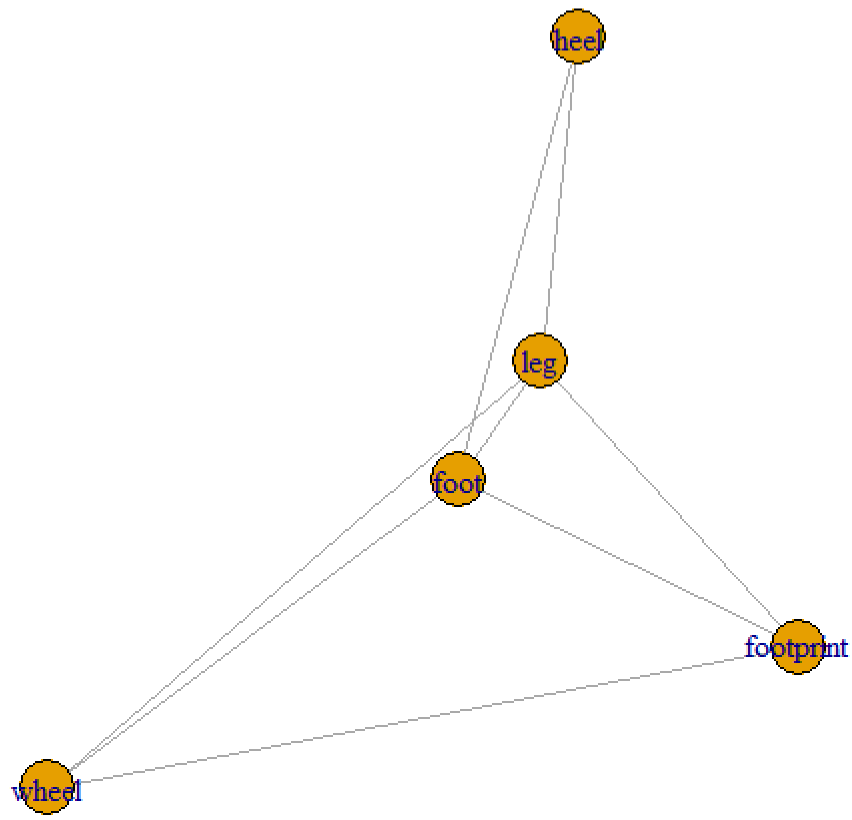
Important:

1) The column with weights should be named so explicitly.

2) No zero weights.

Make a graph from a data frame

```
> foot_graph1 <- graph_from_data_frame(foot_df,
directed = FALSE)
> plot(foot_graph1)
```


heel
leg
foot
footprint
wheel

Exercise

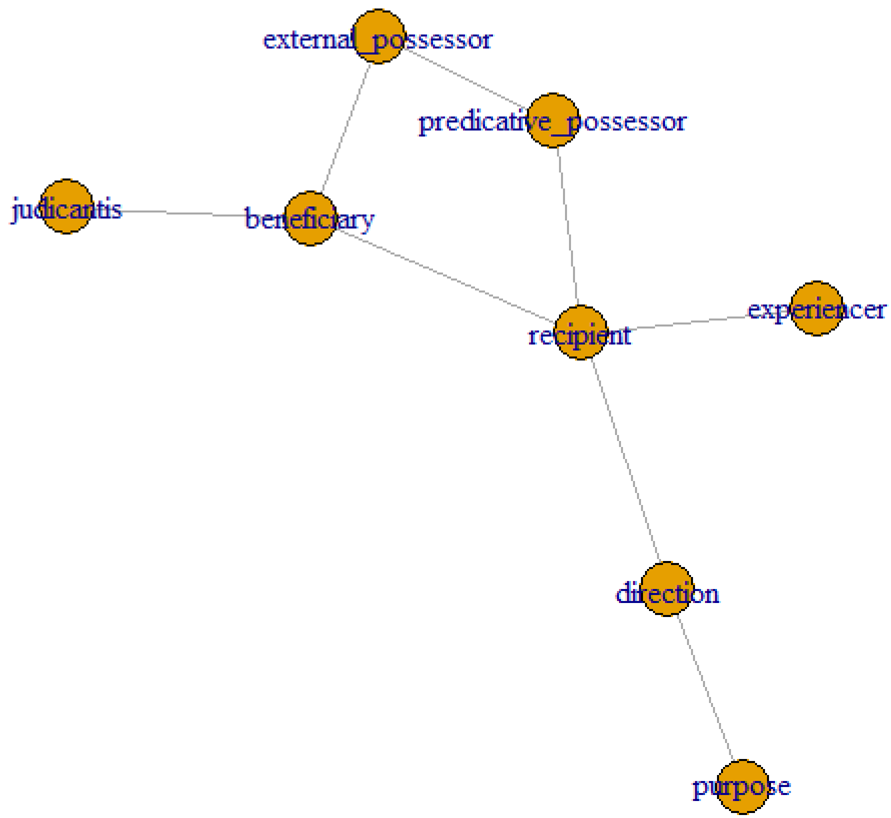
- Choose any concept from CLiCS. Find its 5 most frequent colexification neighbours or more (Query > All Links).
- Find the colexification frequencies between all these concepts (Query > Direct Links).
- Create a matrix with those frequencies in R or a data frame with edges and their weights.
- Represent the relationships in a static weighted graph.
- Represent the network in a dynamic weighted D3 graph.

Appendix

R code for making graphs from Haspelmath and Bybee

Making undirected graphs with R (datives)

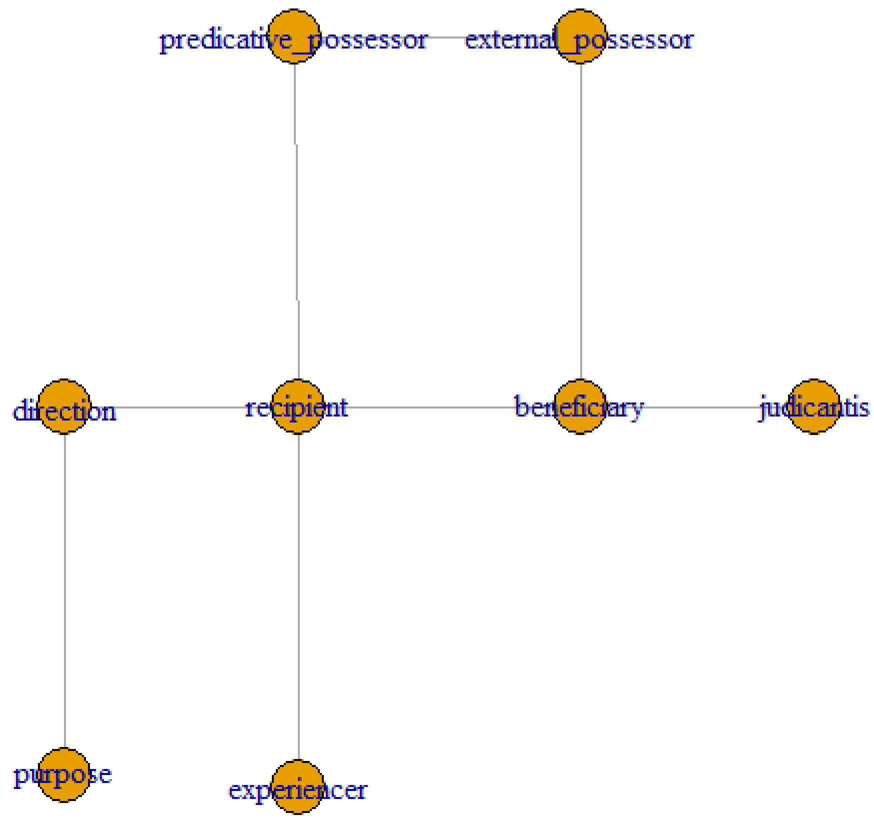
```
> dat_graph <- make_graph(~ purpose - direction - recipient - beneficiary -  judicantis, experiencer - recipient - predicative_possessor - external_possessor - beneficiary) #one way of providing the edges

> plot(dat_graph)
```



external_possessor
predicative_possessor
judicantis
beneficiary
recipient
experiencer
direction
purpose

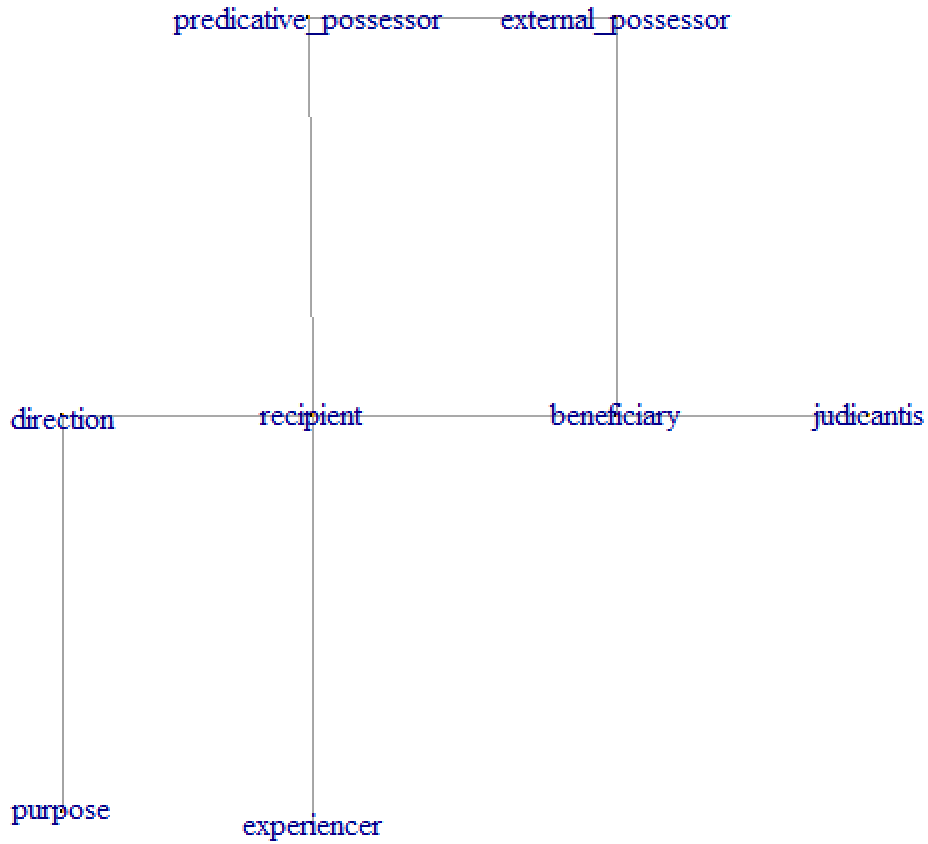
Edit your graph interactively

```
> tkplot(dat_graph) #call the interactive plot
[1] 1 #ID of the plotting device
> xy <- tk_coords(1) #save coordinates; change the
ID, if necessary
```


predicative_possessor
external_possessor
direction
recipient
beneficiary
judicantis
purpose
experiencer

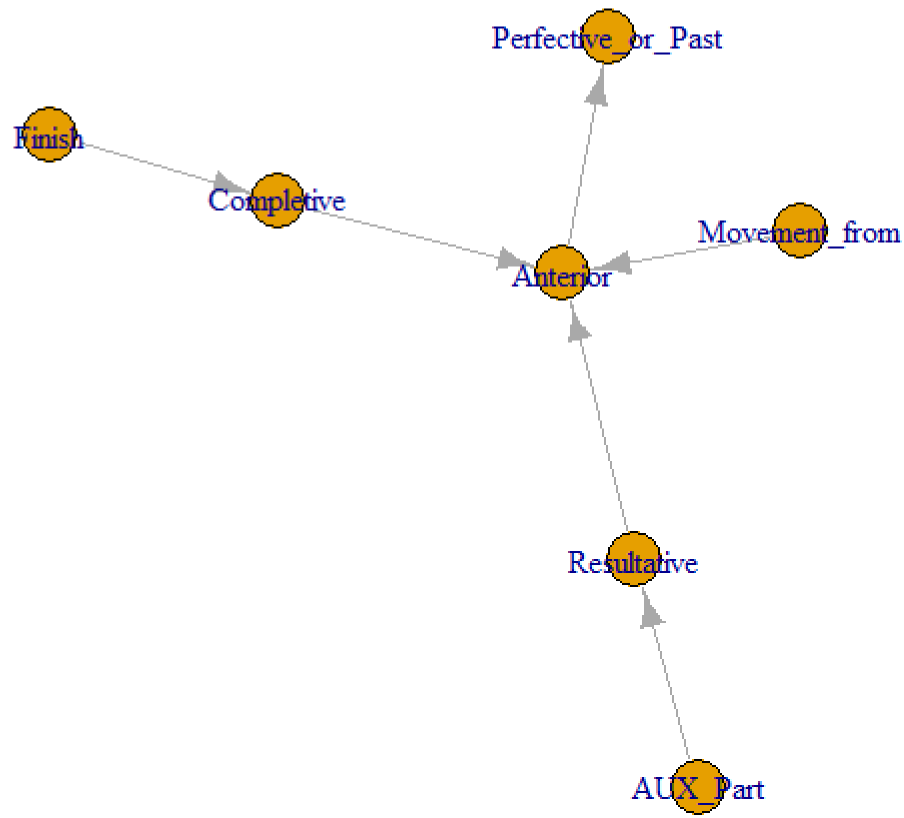
Plot your graph with new coordinates

```
> plot(dat_graph, layout = xy) #plot with the new coordinates
> plot(dat_graph, layout = xy, vertex.size = 0) #no vertex symbols
```


predicative_possessor
external_possessor
direction
recipient
beneficiary
judicantis
purpose
experiencer

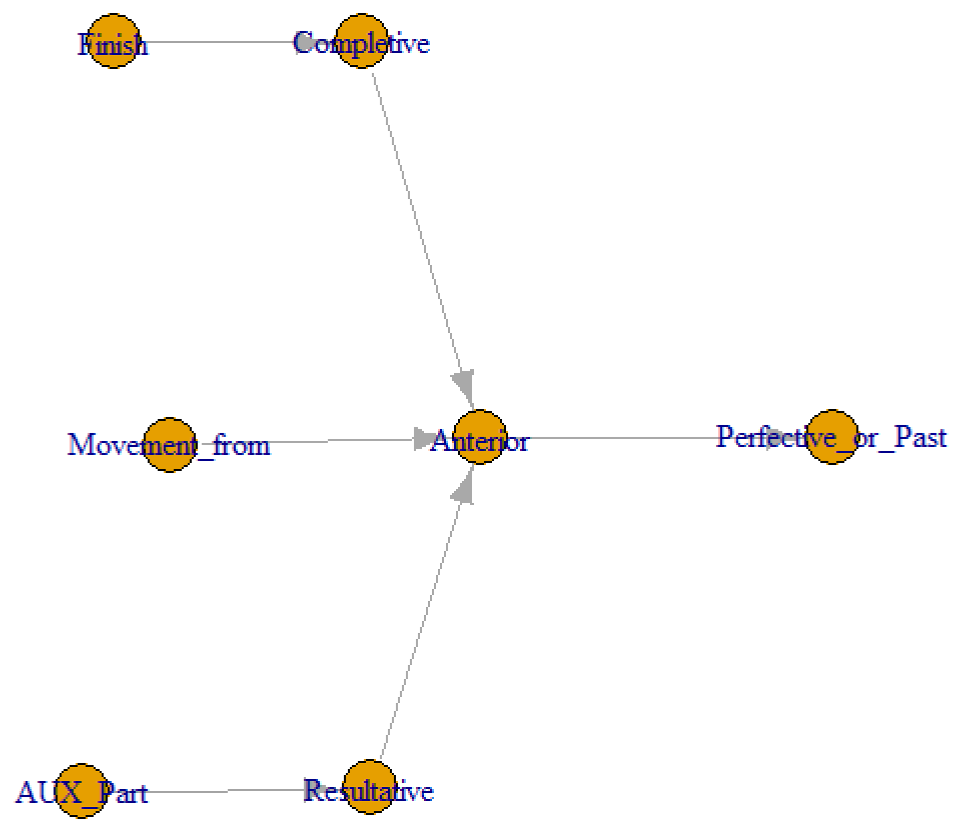
Directed graphs with R (tense and aspect grams)

```
> ta_graph <- make_directed_graph(c("AUX_Part",
"Resultative", "Resultative", "Anterior",
"Anterior",  "Perfective_or_Past", "Finish",
"Completive", "Completive",  "Anterior",
"Movement_from", "Anterior")) #another way of
entering the data

> plot(ta_graph)
```

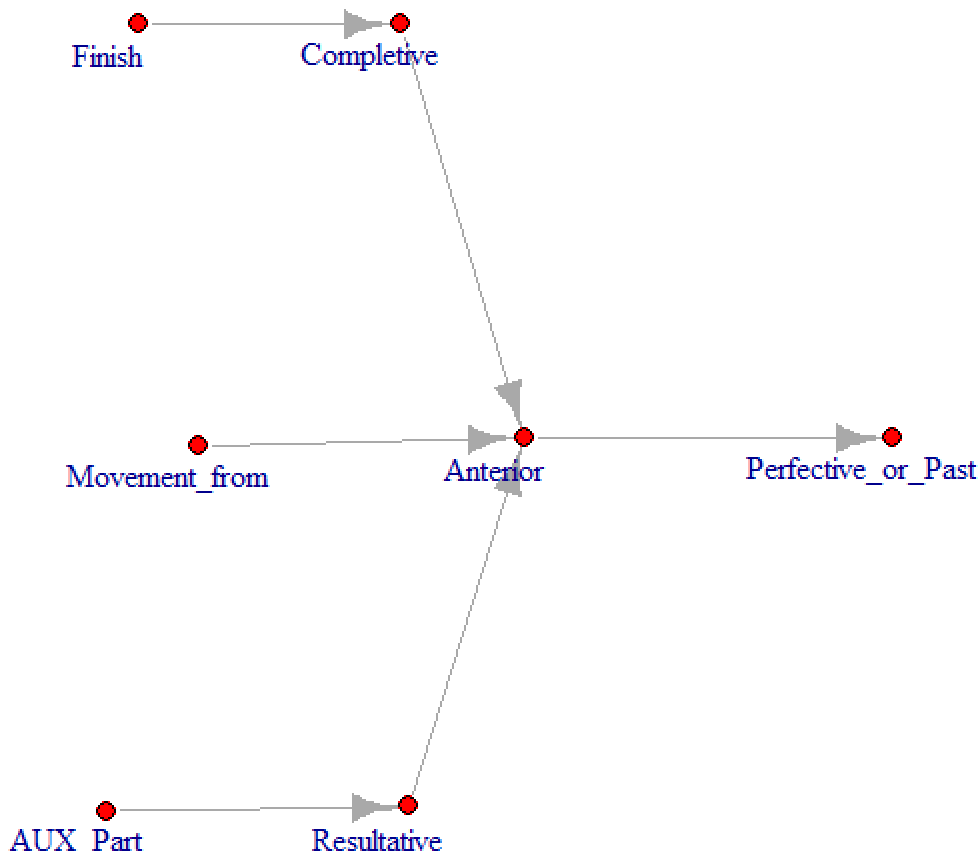

Perfective_or_Past
Finish
Completive
Movement_from
Anterior
Resultative
AUX_Part

Edit the graph interactively and plot it with new coordinates

```
> tkplot(ta_graph)
> tk_coords(2) #or another id of the device
> xy <- tk_coords(2)
> plot(ta_graph, layout = xy)
> plot(ta_graph, layout = xy, vertex.size = 5,
vertex.label.dist = -1.5, edge.arrow.size= 1.2,
vertex.color = "red") #a prettier version
```



Finish
Completive
Movement_from
Anterior
Perfective_or_Past
AUX_Part
Resultative


Finish
Completive
Movement_from
Anterior
Perfective_or_Past
AUX_Part
Resultative

References

- Haspelmath, M. (2003) The geometry of grammatical meaning: Semantic maps and cross-linguistic comparison. In Tomasello, Michael (ed.), *The new psychology of language*, vol. 2. Mahwah, NJ: Lawrence Erlbaum, 211-242.
- Bybee, J. L., R. Perkins & W. Pagliuca. 1994. *The Evolution of Grammar: Tense, Aspect, and Modality in the Languages of the World*. Chicago/London: The University of Chicago Press.
- van der Auwera, J. (2013) Semantic maps, for synchronic and diachronic typology. In A. Giacalone Ramat, C. Mauri & P. Molinelli (eds.), *Synchrony and diachrony: a dynamic interface*. Amsterdam: Benjamins, 153-176.