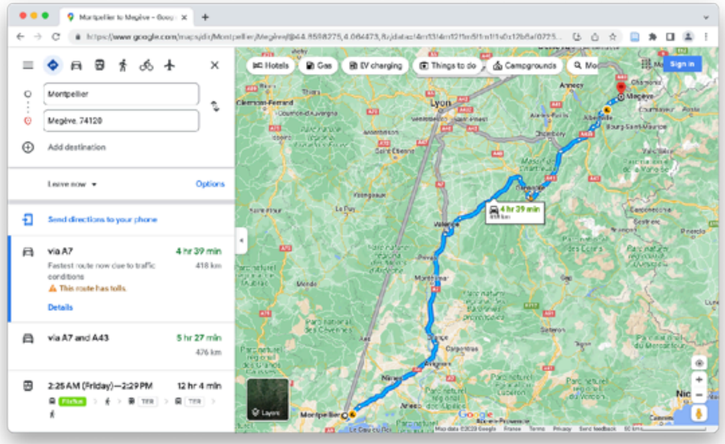
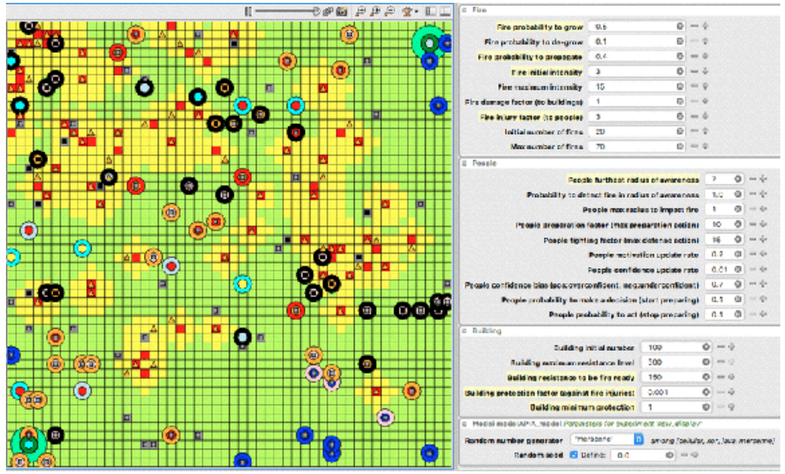
Interacting with numbers

Thibault Raffaillac

Journées de Rochebrune, 18 January 2023

Interacting with computers



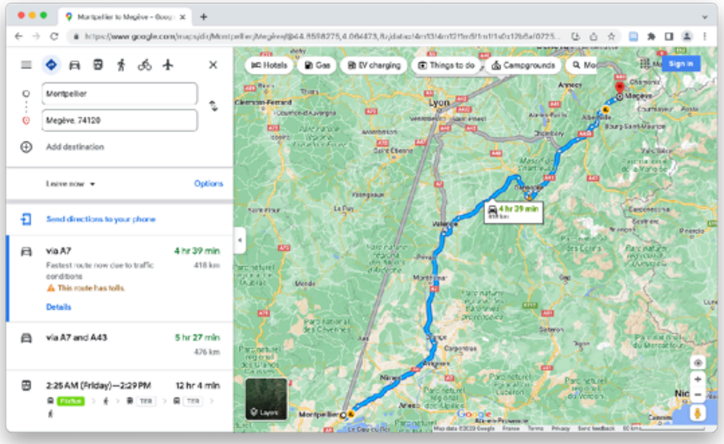


Interacting with computers

back-end

front-end

user

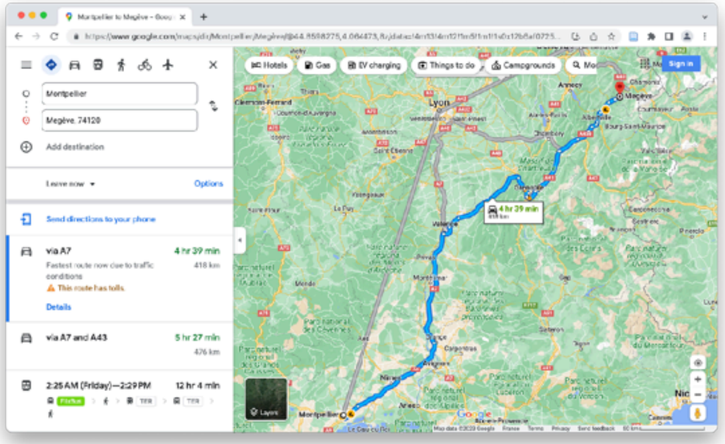


Interacting with computers

back-end

front-end

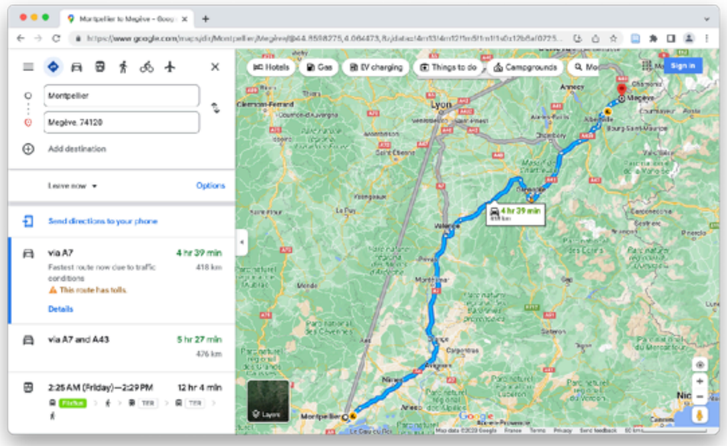
user



(x, y)
(x0, y0, x1, y1)

click
drag&drop
scroll

Interacting with controls



OK

value 0/1

number in an interval

zone de texte

character string

option 1
option 2

index in a list

choix

value 0/1

jj/mm/aaaa

date

condiments
mayo
ketchup

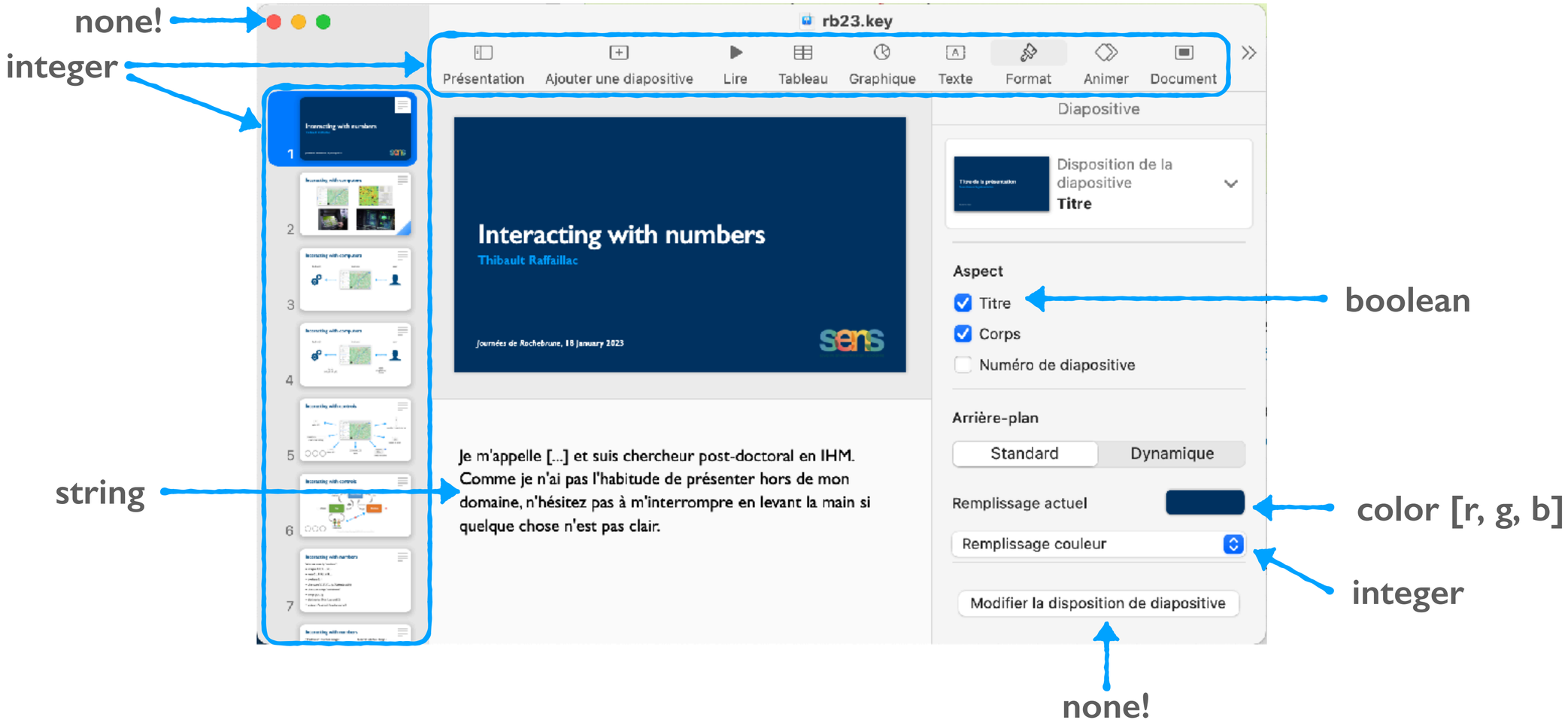
index in a list

Interacting with controls

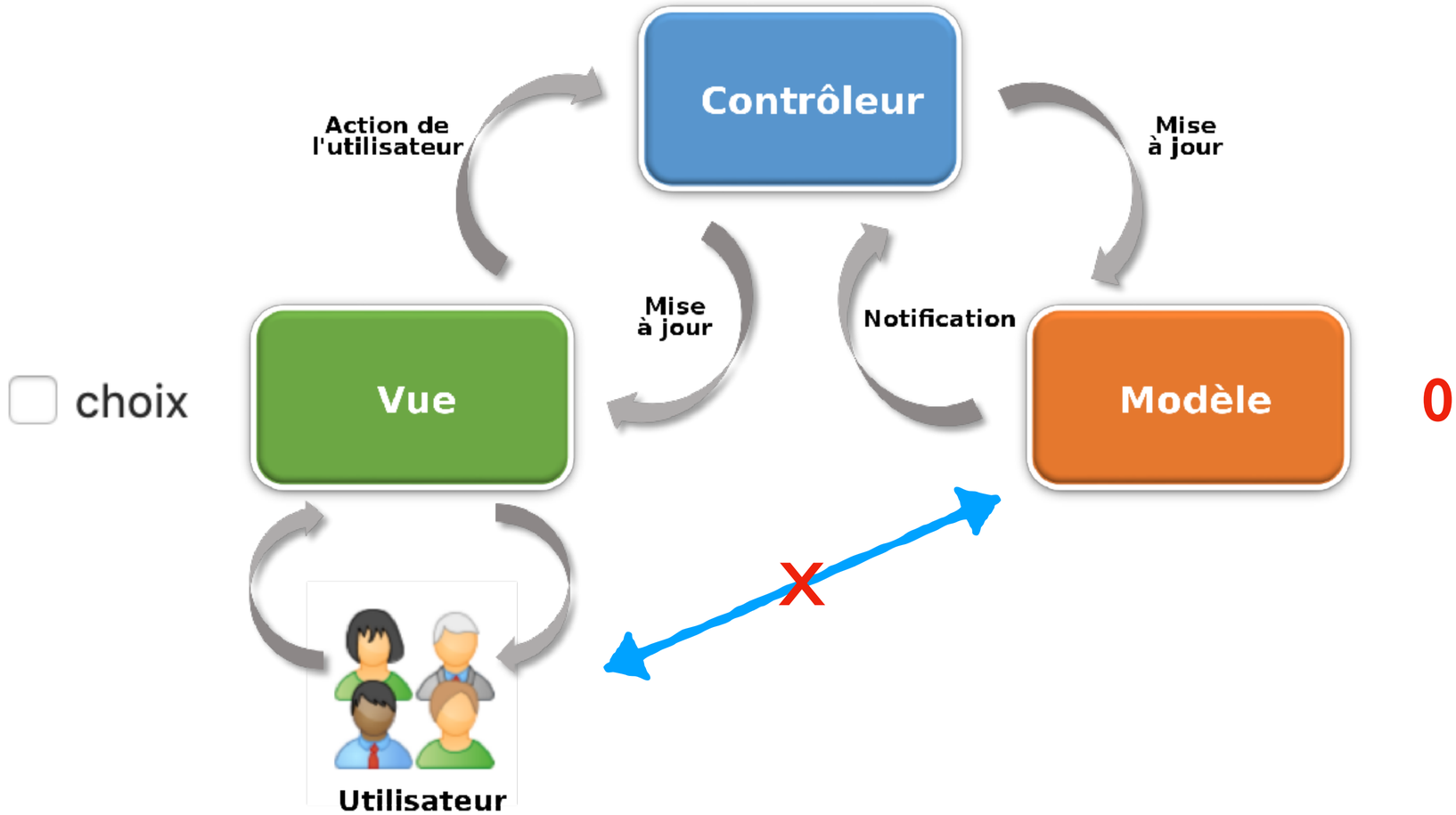
What we mean by “numbers”:

- integers 0, 1, 2, ..., 57, ...
- reals 0.1, 3.14, 1.618, ...
- booleans 0, 1
- characters 'a', 'b', 'c', ... (c.f. Unicode table)
- character strings "rochebrune"
- arrays [0, 1, 2]
- dictionaries {first: 1, second: 2}
- pointers ["a string", "another string"]

Interacting with controls



Interacting with controls



© https://fr.wikipedia.org/wiki/Modèle-vue-contrôleur

Interacting with numbers

"Traditional" interface design :

1. Back-end programming
2. Defining a visual interface by assembling controls (*encapsulating numbers*)
3. Linking the interface with the back-end (e.g. *when users click on this button, execute this function*)

Desired interface design :

1. Back-end programming
2. Designation of the numbers to be controlled, and automatic generation of the interface around them
3. Voilà !

Interacting with numbers

- Who interacts with numbers? ~~Users~~ Programmers

What allows us to move from numbers to interactive controls?

Related works

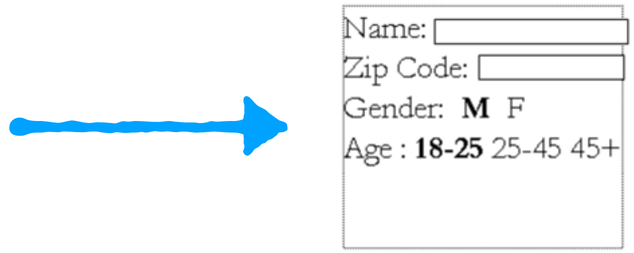
Model-Based User Interface Development

Research field active for 40 years (Meixner et al., 2011)

Brief description of the interface and automatic generation of the rest, 3 types:

- task model
- dialog model
- presentation model

```
<abstractContainer id="idao2" name="Register Data">
  <abstractIndividualComponent id="idao9" name="Input Zip Code">
    <input id="idao15" name="input zip code" actionType="interaction" dataType="String"
      attributeDomainCharacterization="zipCode" />
  </abstractIndividualComponent>
  <abstractIndividualComponent id="idao10" name="Input Name">
    <input id="idao14" name="input Name" actionType="interaction" dataType="String"
      attributeDomainCharacterization="name" />
  </abstractIndividualComponent>
  <abstractIndividualComponent id="idao11" name="input gender">
    <input id="idao16" name="Select gender" actionType="interaction" dataType="String"
      attributeDomainCharacterization="gender" />
  </abstractIndividualComponent>
  <abstractIndividualComponent id="idao12" name="input age cathegory">
    <input id="idao17" name="input ageCategory" actionType="interaction" dataType="String"
      attributeDomainCharacterization="ageCategory" />
  </abstractIndividualComponent>
```



UsiXML, Vanderdonckt et al., 2009

Related works

Model-Based User Interface Development

Limits :

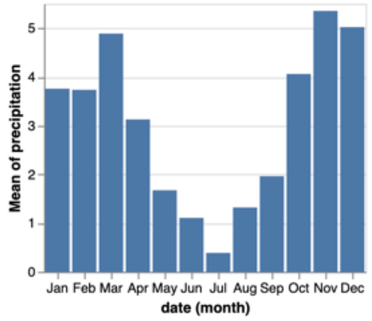
- requires a dedicated learning for the tool or language
- still verbose (vs. generated interface)
- often requires fixing the interface proposed by the tool

Related works

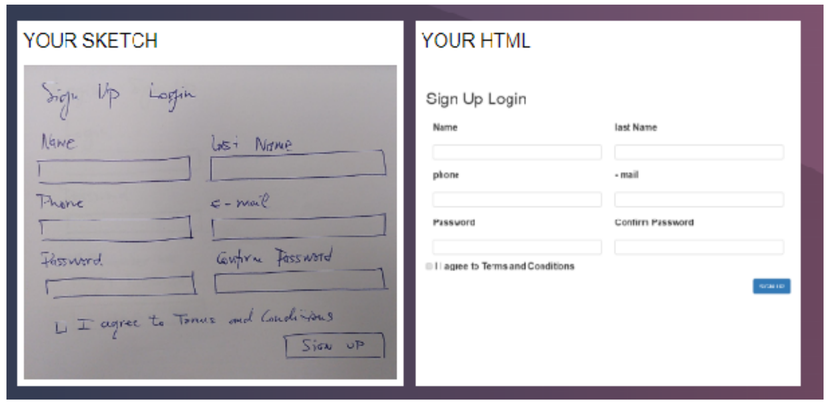
Ad-hoc tools

Data → visualization
(ex. *Vega Lite*, Excel)

```
{
  "data": {"url": "data/seattle-weather.csv"},
  "mark": "bar",
  "encoding": {
    "x": {"timeUnit": "month", "field": "date", "type": "ordinal"},
    "y": {"aggregate": "mean", "field": "precipitation"}
  }
}
```



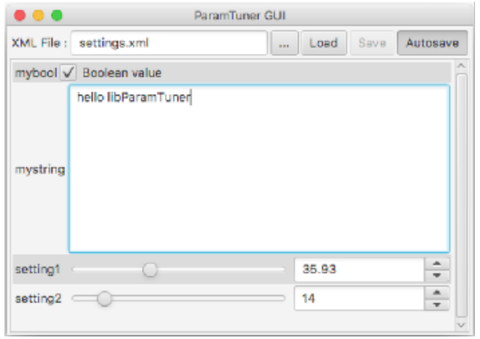
Sketch → interface
(ex. *Sketch2Code*, Uizard)



https://jee138doshi.medium.com/sketch2code-a01922fa14c9

Parameters → interface
(ex. *Baloup et al., 2017*)

```
<?xml version="1.0" encoding="UTF-8"?>
<ParamList>
    <varD type="double" value="2.3" min="0" max="100"/>
    <varI type="int"    value="12"  min="0" max="100"/>
    <varS type="string" value="lorem ipsum"/>
</ParamList>
```



Moving from numbers to interactive controls

No universal solution, but gathering hints from multiple sources:

numbers	structure	environment	user
type (ex. integer → choice in menu, character → keyboard control, boolean → checkbox)	cardinality (ex. pair → 2D point, triplet → date or color, sextuplet → date and time)	relative position (ex. number at the root → full page control, number among others → thin control)	a priori indications (ex. this group of numbers is a date, I want to control this with mouse, this must fit in a small square)
distribution of values (e.g. [1900-2100] → calendar)	hierarchy (ex. list of character strings → menu, list of images → carousel)	domain of the tool (ex. parameters → horizontal stacked controls, avionics → push buttons and rotary dials)	a posteriori indications (ex. rotate this control by 90°, make it taller)

Autograph :
https://www.youtube.com/watch?v=xW_t5q7wqxc

Autograph

Number and structure hints

```
G = [
    [0, 1, 1, 0],
    [0, 0, 1, 0],
    [0, 0, 0, 1],
    [0, 0, 0, 0]]
```

Using **introspection**:

```
print(type(G)) # <class 'list'>

print(isinstance(G, list)) # True

print(isinstance(G[0], list)) # True
```

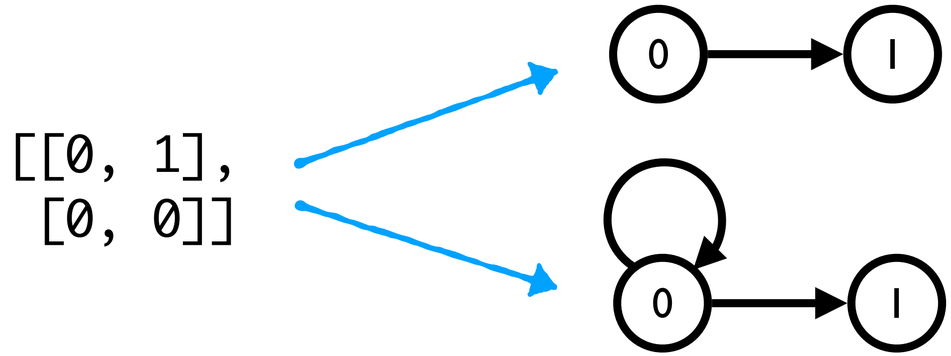
G is the adjacency matrix of an unlabeled directed graph if:

- G is a list
- all elements in G are lists
- G and all its elements have same length
- all elements of elements of G are integers with values 0/1

Autograph

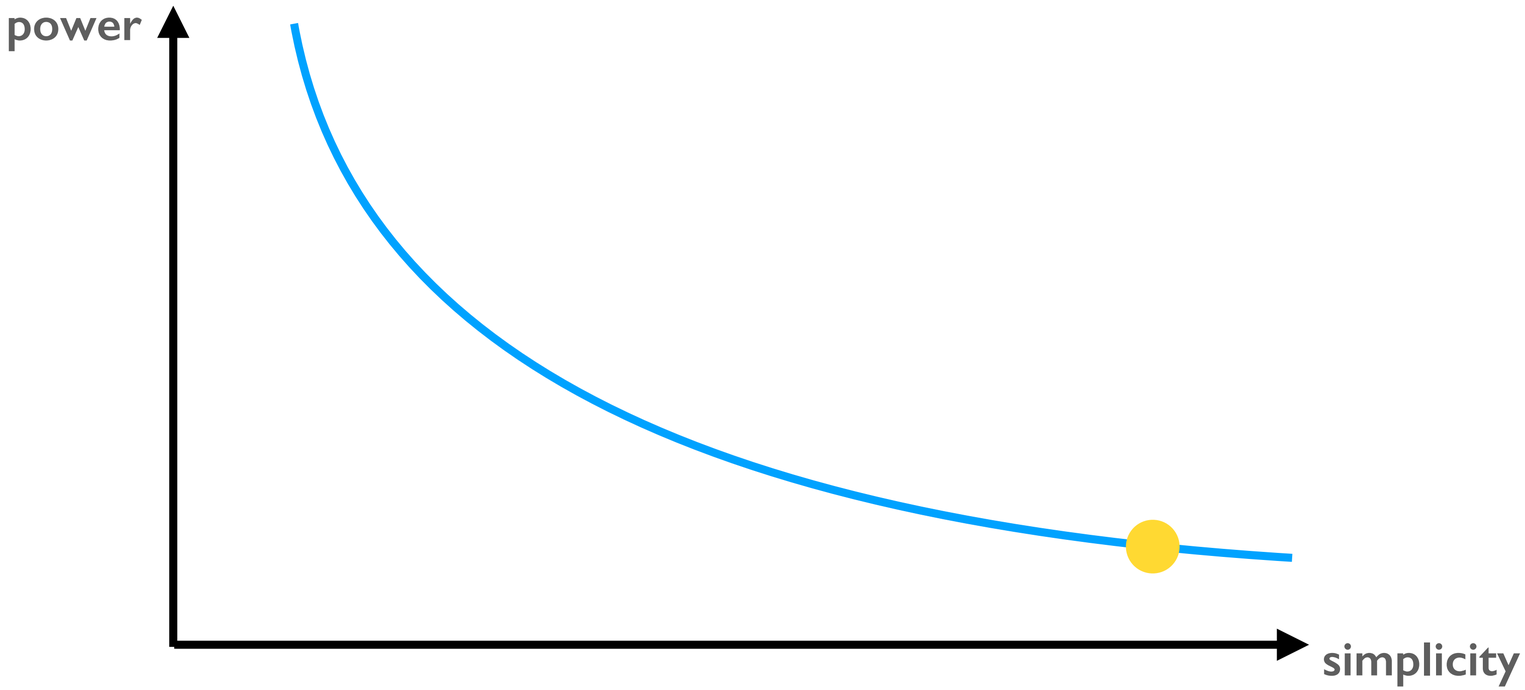
User hints

- Executing `autograph(G)` is an a priori hint that G looks like a graph (hence not `autoviz(G)` or `autoplot(G)`)
- Users can change the selected interpreted structure to fix ambiguous graphs (a posteriori hint)



Limits

- Autograph generates a control, not a full interface
- Can only display generic graphs (no visual distinction between types of nodes, no hierarchy between elements, ...)



Future work

- Moving from proof of concept to complete demonstration
- Disseminating to high schools for the teaching of graphs in NSI speciality
- Applying this method to other complex structures (e.g. Automap)
- *Can we apply this method (providing a structure of numbers, introspection) to the generation of entire user interfaces??*

Thank you for your attention