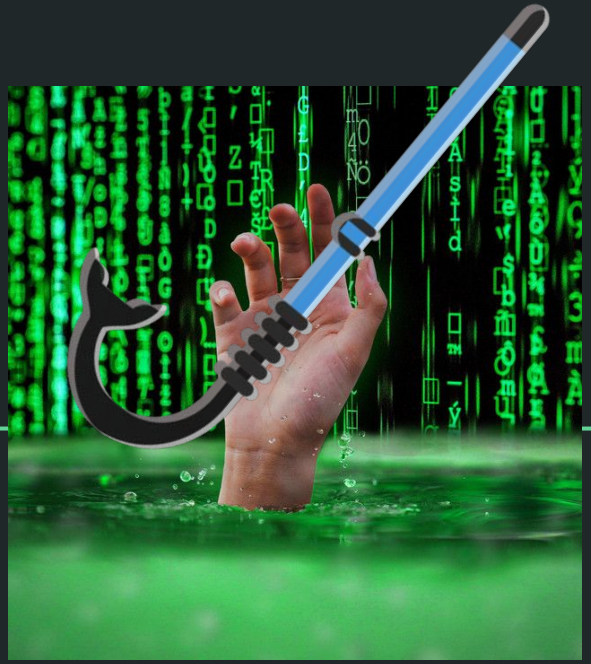
Perceptual reasoning

How not to drown in sensor data



Key questions

“A model of categorization needs to address

- **topics like the representation of concepts**
- **the strength of membership in a category**
- **mechanisms for forming new concepts**
- **the relation between a concept and the outside world**
- **and so on”**

Paper: A logic of categorization - Pei Wang and Douglas Hofstadter

Frequency value of stimuli statements from sensor values

<{A} ➜ [sizeX]>. :|: f1

Stimulus A has attribute P value larger than f1*100 percent of recorded cases

<{B} ➜ [P]>. :|: f2

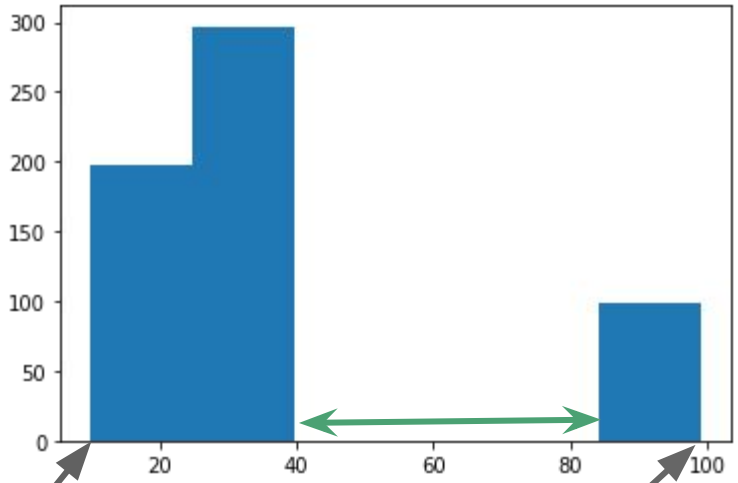
Establish comparative relationships, if f1 > f2:

({A} * {B}) ➜ [P]> f=1, c=c1*c2 if continuous, else f=f1*(1-f2)

<{A} <-> {B}> f=1-|f1-f2|, c=c1*c2 if continuous else f=and(f1,f2)/or(f1,f2)

Example

```
rounds = 99
for i in range(rounds):
 if i == rounds - 1:
   print("Final truth values after", rounds, "rounds:")
 Print = i == 0 or i == rounds-1
 reportValue(10.0, Print=Print)
 reportValue(20.0, Print=Print)
 reportValue(30.0, Print=Print)
 reportValue(33.0, Print=Print)
 reportValue(32.0, Print=Print)
 reportValue(99.0, Print=Print)
```



Final truth values after 99 rounds:
<{S} --> [P]>. :|: {**0.0**, 0.8999440054750203} from value **10.0**
<{S} --> [P]>. :|: {0.08140108534780464, 0.8999260050974266} from value 20.0
<{S} --> [P]>. :|: {0.2905497945607513, 0.8999119632201897} from value 30.0
<{S} --> [P]>. :|: {0.3796873501486525, 0.8999136939010357} from value 33.0
<{S} --> [P]>. :|: {0.34842115553817576, 0.8999120234604105} from value 32.0
<{S} --> [P]>. :|: {**1.0**, 0.899975430646174} from value **99.0**

Concept formation

- Keep up to K instances and concept nodes

New instance **{inst2}** arrives:

- Find closest known in terms of similarity **<{inst2} <-> {?1}>?** by considering overlap in all attributes

  <{inst2} <-> {inst1}>. (Comparison, value similarity)

- Increase useCount for best matched (forgetting based on useCount)
- Find biggest attribute differences to best matched to build

  <({inst1} * {inst2}) ➜ [blue]> > (Difference, value differences)

- Find best matching concept node in terms of **<{inst1} ➜ ?1>?** to build (again increasing useCount of answer)

  <{inst1} ➜ concept> %f% (Abductions, value similarity + revisions)

- Use common closest attribute to form new concept nodes

{inst1}

t=1

[pink] (most significant difference)

{inst1}

{inst2}

concept

t=2

(from common properties, e.g. 4 legs, trunk, and horns)

Pink elephant example:

(describe/notice new instances in relation to their closest match and relative to their best concept belonging)

<{inst2} <-> {inst1}>. f1,c1

<{inst2} ➔ elephant>. f2, c2

<({inst2} * {inst1}) ➔ (+ pink)>. 1,c3

For vision, for now

- Use YOLOv4 output if it has to work with natural images

- Use label with softmax probability as frequency, width, height, color summary within BB for (R, G, B) values describing the detected pattern (what)

- Treat positionX, position Y extra to build spatial relations among objects (where)

- Feed results to NARS in continuous form, using previous mechanisms to generate compressed description (what is it most similar to, what category does it belong to, how does it differ the most to the most similar instance and category it belongs to)

('inst2',
 {'label': ['person'],
  'width': [492],
  'height': [1216],
  'color': [63, 54, 52]})

('inst1',
  {'label': ['person'],
   'width': [327],
   'height': [1157],
   'color': [105, 110, 119]})

Reasoning result

Nalifier:

<{inst1} --> [label_person]>. %1.0%
<{inst1} --> [width0]>. %0.5%
<{inst1} --> [height0]>. %0.5%
<{inst1} --> [color0]>. %0.32948490230905864%
<{inst1} --> [color1]>. %0.365787896944278%
<{inst1} --> [color2]>. %0.43388429752066116%
Derived: <{inst1} --> [see]>. :|:
<{inst2} --> [label_person]>. %1.0%
<{inst2} --> [width0]>. %1.0%
<{inst2} --> [height0]>. %1.0%
<{inst2} --> [color0]>. %0.0979306324686802%
<{inst2} --> [color1]>. %0.0679882794615877%
<{inst2} --> [color2]>. %0.0622093810663175%
Derived: (<{inst2} <-> {inst1}> &| <({inst2} * {inst1}) --> (+ width0)>). :|: {0.63 0.95}

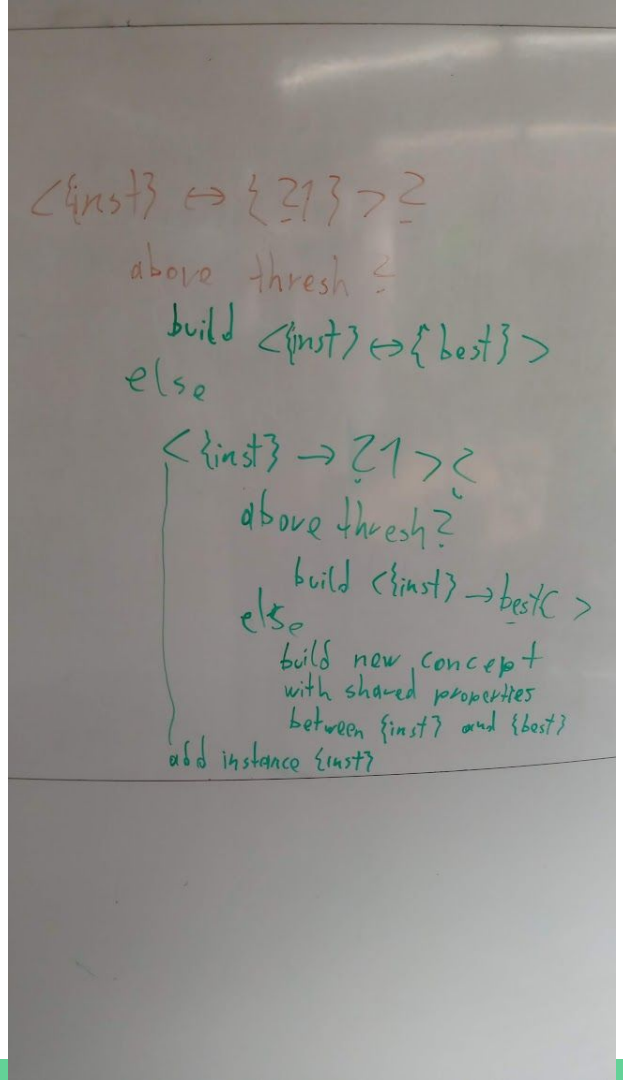
ONA:

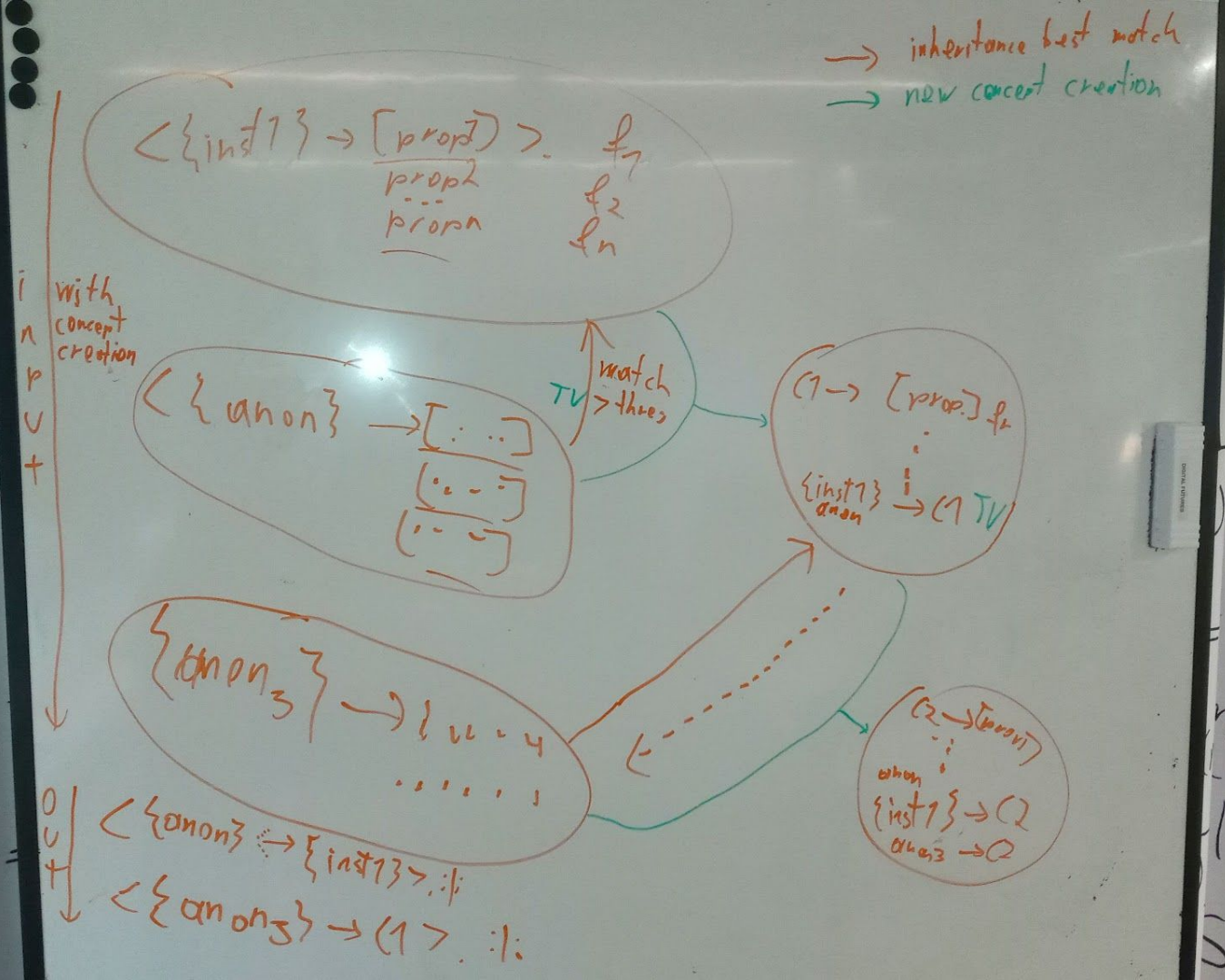
Derived: <{inst1} --> [see]>. :|: {0.63 0.54}
Derived: <({inst2} * {inst1}) --> (+ width0)>. :|: {0.63 0.54}
Input: <({?1} * {inst1}) --> (+ width0)>? :|:
Answer: <({inst2} * {inst1}) --> has_less>. :|: {0.63, 0.54}

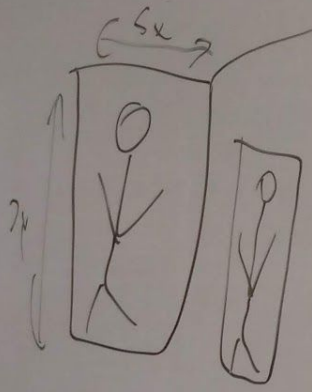
Thank you!

Creating new instances (when there is no good instance match) and concepts (when there is no good concept belonging)

Can later be compared with control approaches which makes less radical control decisions




→ inheritance best match
→ new concept creation
input
with concept creation
match > thres
TV
output



label
size x
size y
ref
green
blue

[big]&human

{human2}

{human}

[small]&human

{human1} → case

({human1} × case) → [sizey]+