Which Model Does Not Belong?

A Dialogue (Monologue?)

Michael R. Gryk & Bertram T. Ludäscher

Center for Informatics Research in Science & Scholarship (CIRSS)

School of Information Sciences

University of Illinois, Urbana-Champaign

ILLINOIS
School of Information Sciences

https://github.com/sig-cm/JCDL-2020

*JCDL **SIG**CM Workshop on Conceptual Modeling*
August 1, 2020

1 Introduction

- A monk was summoned to the Buddha's chamber.
- Upon a table were **four** pots: three gold and one silver.
- The **first gold pot** was large with ornate handles.
- The **second** could have been its younger sibling, sharing the same shape and handles yet of a **diminutive** size.
- The **third** was as big as the first but had **no handles**.
- The **silver pot** was large with handles.

- *B: Tell me my student, which of these pots is unlike the others?*
- *M: The **silver** one of course. The **others** are made of **gold**.*
- *B: Please **hoist** each one above your head.*
- *M: Ah, master. I beg forgiveness. The one **without handles** is truly unique.*
- *B: Would you please use them to **milk the cow**.*
- *M: Once again master, I have changed my mind. The **smaller** one is inferior.*
- *B: Several times I have given you a task and each time you have made a different choice. **What task could I assign which would convince you to choose the large, handled pot of gold?***
- *M: There is no such task, master. **That pot is not unique in any way.***
- *B: … Isn't it the only pot of that kind … grasshopper??*

2. Cutting to the Chase …

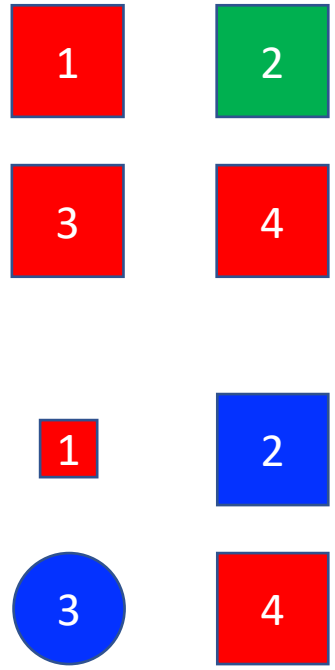
- **WODB** (which one doesn't belong) is a great educational tool for CM!
  - … Datalog and Answer Set Programming are power tools for WODB (and CM in general)

- Step 1: **Define what is (and isn't!) part of your model:**
  - **Observables** (things/concepts you can ask about)
  - **Ignorables** (… you cannot ask about!)

- Step 2: **Fitness-for-Purpose** (FFP) and **Conceptual Resolution** (CR) via **Queries**:
  - **FFP/Q:** Can you answer the questions you care about?
  - **CR/Q**: What questions can you answer (what concepts can you distinguish) in principle?

… cutting to the chase …

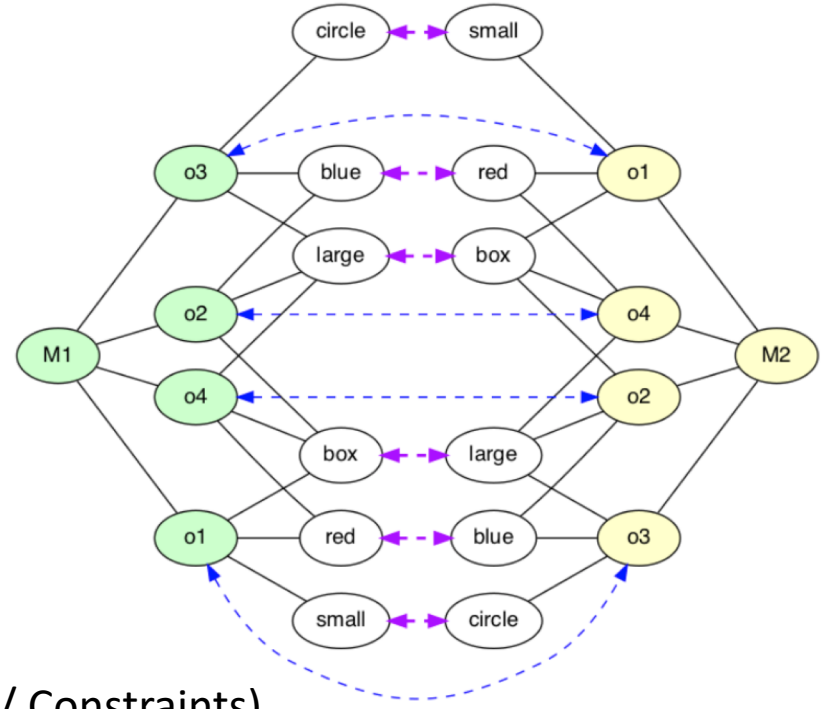
- Step 3 (The WODB Point): **Explainable Answers:**
  - How do you **justify** Which-One-Doesn't-Belong (**WODB**)?
  - i) Agree on **Observables** (color; shape; size; …)
    - … and **Ignorables** (object-ID; object location: NE, SE, SW, N; …)
  - ii) **Formalize** what you ***mean*** using a **Query**:
    - In **UX: What you see is what you get! (WYSIWYG!)**
    - In **CMX: What you get, is it what you see? (WYGiiWYS?)**
    - … b/c if you can't pick the thing which isn't like the other via a query
    - … maybe your query and your ideas/concepts don't agree!
    - … or your model doesn't mean what you think it means!!
  - iii) Even if WYGiWYS(!), **compare justifications** with those for alt-answers by others:
    - Beware: minimal change of your justification = my justification!
    - … or rather: different quality of justifications (cf. Koan .. WODB Ex. 5 below) – being **"meta-special"**



… cutting to the chase …

- Step 4: **Quantifying over all Queries: Semantics from Structure!**

  - **How semantics permeates (conceptual) models:**
    - … via **Vocabulary** (Controlled Vocabs; Ontologies (eg in DLs); etc.)
    - ➔ ***Terminological Perspective*** (another day!)
    - … via **Structure & Connections**
    - ➔ ***Database Theory / Mathematical Logic Perspective*** (today!)

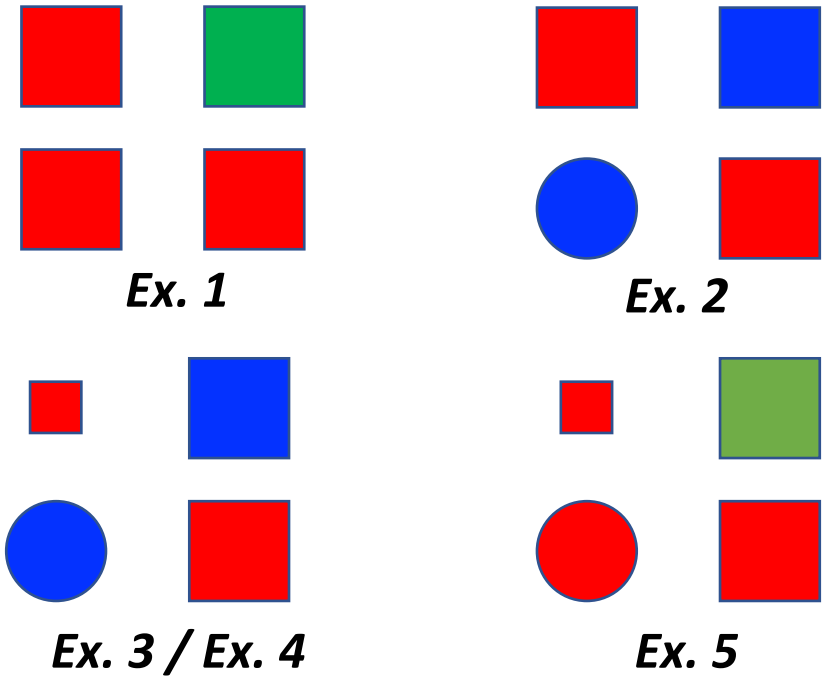


  - **Tools to obtain semantics from structure:**
    - Queries**, Queries, QUERIES**! (cf. Steps 2 & 3)
    - **Automorphisms**
      - Compute in a few easy steps with ASP (= Datalog + Generate & Test / Constraints)
      - … and visualize with *Possible-Worlds-Explorer* (**PWE**); tool created by **Sahil Gupta!**

… Some PWE Snippets …

```
% For any pair of vertices X,Y, there is 0 or 1 map atom m(X,Y):
0 { m(X,Y) } 1 :- v(X), v(Y).

% X1--e-->X2 is mapped to Y1--e-->Y2 whenever X1--m-->Y1 and X2-->m-->Y2
:- e(X1,X2), m(X1,Y1), m(X2,Y2), not e(Y1,Y2).
% ... and vice versa:
:- e(Y1,Y2), m(X1,Y1), m(X2,Y2), not e(X1,X2).

% Each vertex X is source and sink of exactly one m-arc:
:- v(X), #count {Y: m(X,Y)} != 1.
:- v(X), #count {Y: m(Y,X)} != 1.
```

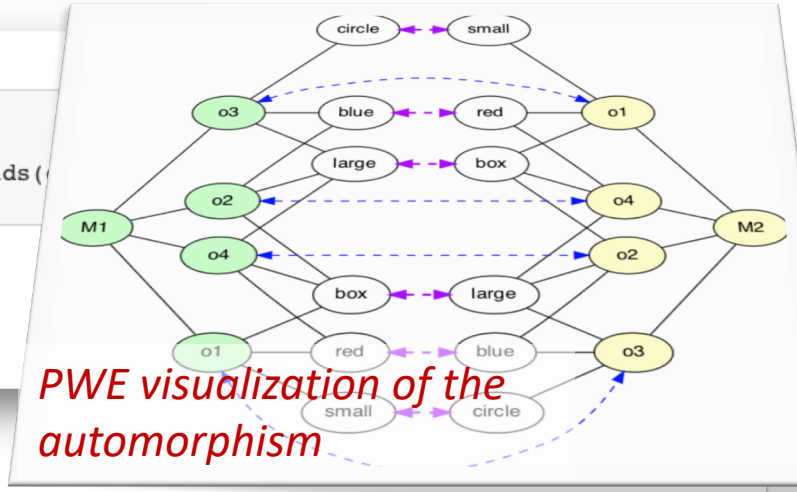
5 logic rules (ASP) to find automorphisms



Visualizing automorphisms:

```
[13]: exps = [ex1, ex2, ex3, ex4, ex5]
      for ex in exps:
          ex['pw_rel_dfs'], ex['rel_schemas'], ex['pw_objs'] = load_worlds(
      ['meta_data'], internal_facts_as_string=False)

      Number of Models: 12
      Number of Models: 2
      Number of Models: 2
      Number of Models: 2
      Number of Models: 6
```

Batch processing several models.



PWE visualization of the automorphism

```
In [14]: ex1['pw_rel_dfs']['m_2']
Out[14]:
```

	pw	x1	x2
0	1	[t, o1, red]	[t, o1, red]
1	1	[t, o1, large]	[t, o1, box]
2	1	[t, o1, box]	[t, o1, large]
3	1	[t, o2, green]	[t, o2, green]
4	1	[t, o2, large]	[t, o2, box]
...	...	...	...
235	12	o3	o4
236	12	red	red
237	12	box	box
238	12	large	large
239	12	green	green

240 rows × 3 columns

PWE listing of a model

Ex. 1: three red boxes are **indistinguishable.**

Ex. 3: by **swapping** constants, non-trivial **symmetries** (automorphisms) are found! => What can and cannot be picked w/ queries!

https://github.com/idaks/WODB/blob/master/automorphisms/Automorphism%20Exps%20ID-V.ipynb

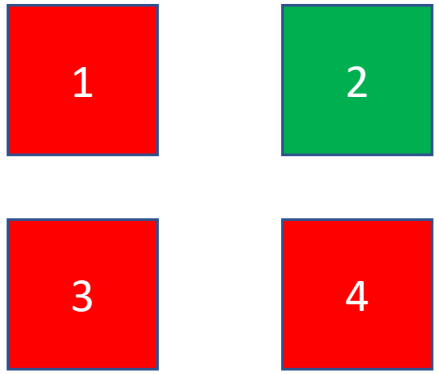
Example 1: It's so easy …

property(fig1, **color**, **red**).
property(fig1, shape, box).
property(fig1, size, large).

property(fig**2, color, green**).
property(fig2, shape, box).
property(fig2, size, large).

property(fig3, **color**, **red**).
property(fig3, shape, box).
property(fig3, size, large).

property(fig4, **color**, **red**).
property(fig4, shape, box).
property(fig4, size, large).

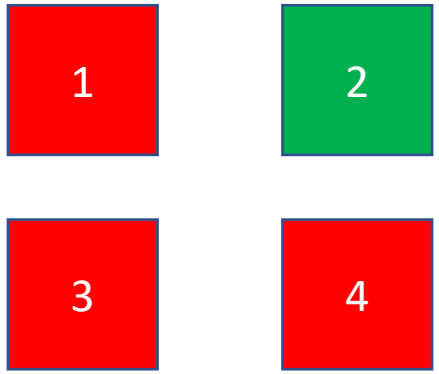



```
% Fig. X is unique wrt property P and value V, if there is no ..
unique(X,P,V) :- property(X,P,V), not another(X,P,V).

% .. other figure Y that has the same property/value pair:
another(X,P,V) :- property(X,P,V), property(Y,P,V), X != Y.

% clingo -n0 example1.lp4 unique.lp4
unique(fig2,color,green)
```

Example 1: It's even easier …

(drop property column; values suffice..)

```
property(fig1, red).
property(fig1, box).
property(fig1, large).

property(fig2, green).
property(fig2, box).
property(fig2, large).

property(fig3, red).
property(fig3, box).
property(fig3, large).

property(fig4, red).
property(fig4, box).
property(fig4, large).
```

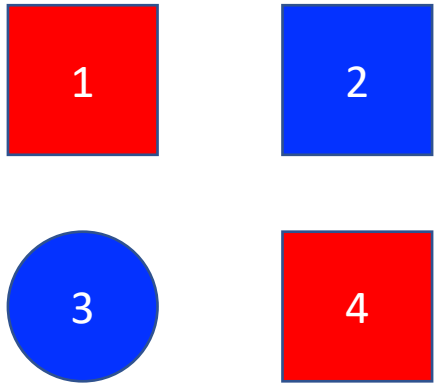



```
% We could add this
% but don't really need it:

type(red, color).
type(blue, color).
type(small, size).
type(large, size).
type(box, shape).
type(circle, shape).
```

```
% Fig. X is unique wrt property-value V, if there is no ..
unique(X,V) :- property(X,V), not another(X,V).

% .. other figure Y that has the same property/value pair:
another(X,V) :- property(X,V), property(Y,V), X != Y.

% clingo -n0 example1.lp4 unique.lp4
unique(fig2,green)
```

Example 2: There's still only one …

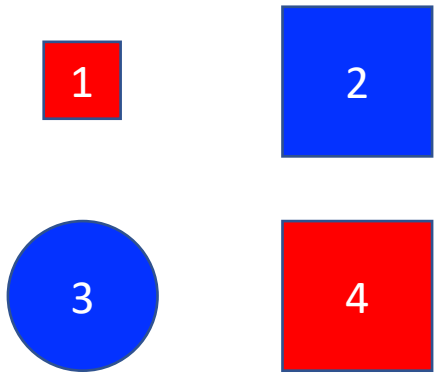
```
property(fig1, red).
property(fig1, box).
property(fig1, large).

property(fig2, blue).
property(fig2, box).
property(fig2, large).

property(fig3, blue).
property(fig3, circle).
property(fig3, large).

property(fig4, red).
property(fig4, box).
property(fig4, large).
```




```
% Fig. X is unique wrt property-value V, if there is no ..
unique(X,V) :- property(X,V), not another(X,V).

% .. other figure Y that has the same property/value pair:
another(X,V) :- property(X,V), property(Y,V), X != Y.

% clingo -n0 example2.lp4 unique.lp4
unique(fig3,circle)
```

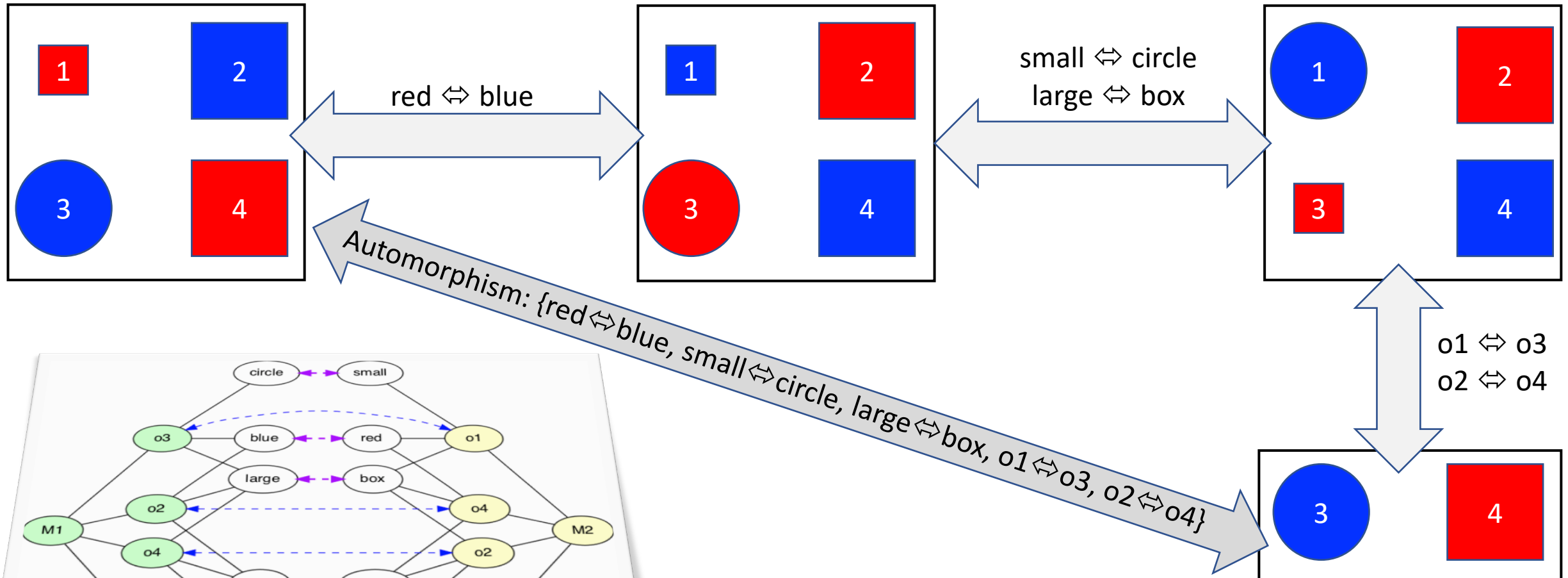
Example 3: who is unique here … ?

```
property(fig1, red).
property(fig1, box).
property(fig1, small).

property(fig2, blue).
property(fig2, box).
property(fig2, large).

property(fig3, blue).
property(fig3, circle).
property(fig3, large).

property(fig4, red).
property(fig4, box).
property(fig4, large).
```




```
% Fig. X is unique wrt property-value V, if there is no ..
unique(X,V) :- property(X,V), not another(X,V).

% .. other figure Y that has the same property/value pair:
another(X,V) :- property(X,V), property(Y,V), X != Y.

% clingo -n0 example3.lp4 unique.lp4
unique(fig1,small)
unique(fig3,circle)
```

Example 3&4: circle/small, blue/red, large/box



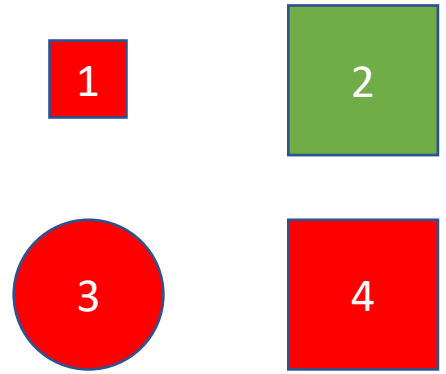
PWE visualization of the automorphism

Raw ASP output of automorphism map:

m(t(o3,blue),t(o1,red)) m(t(o3,large),t(o1,box)) m(t(o3,circle),t(o1,small))
m(t(o4,red),t(o2,blue)) m(t(o4,large),t(o2,box)) m(t(o4,box),t(o2,large)) m(t(o1,red),t(o3,blue))
m(t(o1,small),t(o3,circle)) m(t(o1,box),t(o3,large)) m(t(o2,blue),t(o4,red))
m(t(o2,large),t(o4,box)) m(t(o2,box),t(o4,large)) m(blue,red) m(large,box) m(circle,small)
m(red,blue) m(box,large) m(small,circle) m(o3,o1) m(o4,o2) m(o1,o3) m(o2,o4)

Example 5: You're oh so special (not!?) …

```
property(fig1, red).
property(fig1, box).
property(fig1, small).

property(fig2, green).
property(fig2, box).
property(fig2, large).

property(fig3, red).
property(fig3, circle).
property(fig3, large).

property(fig4, red).
property(fig4, box).
property(fig4, large).
```

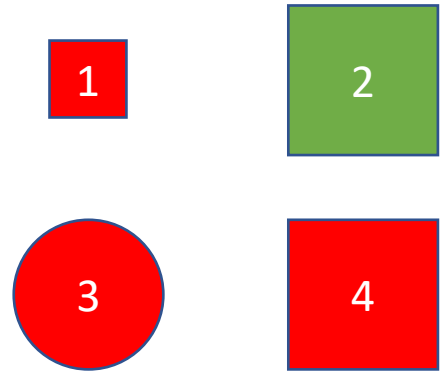


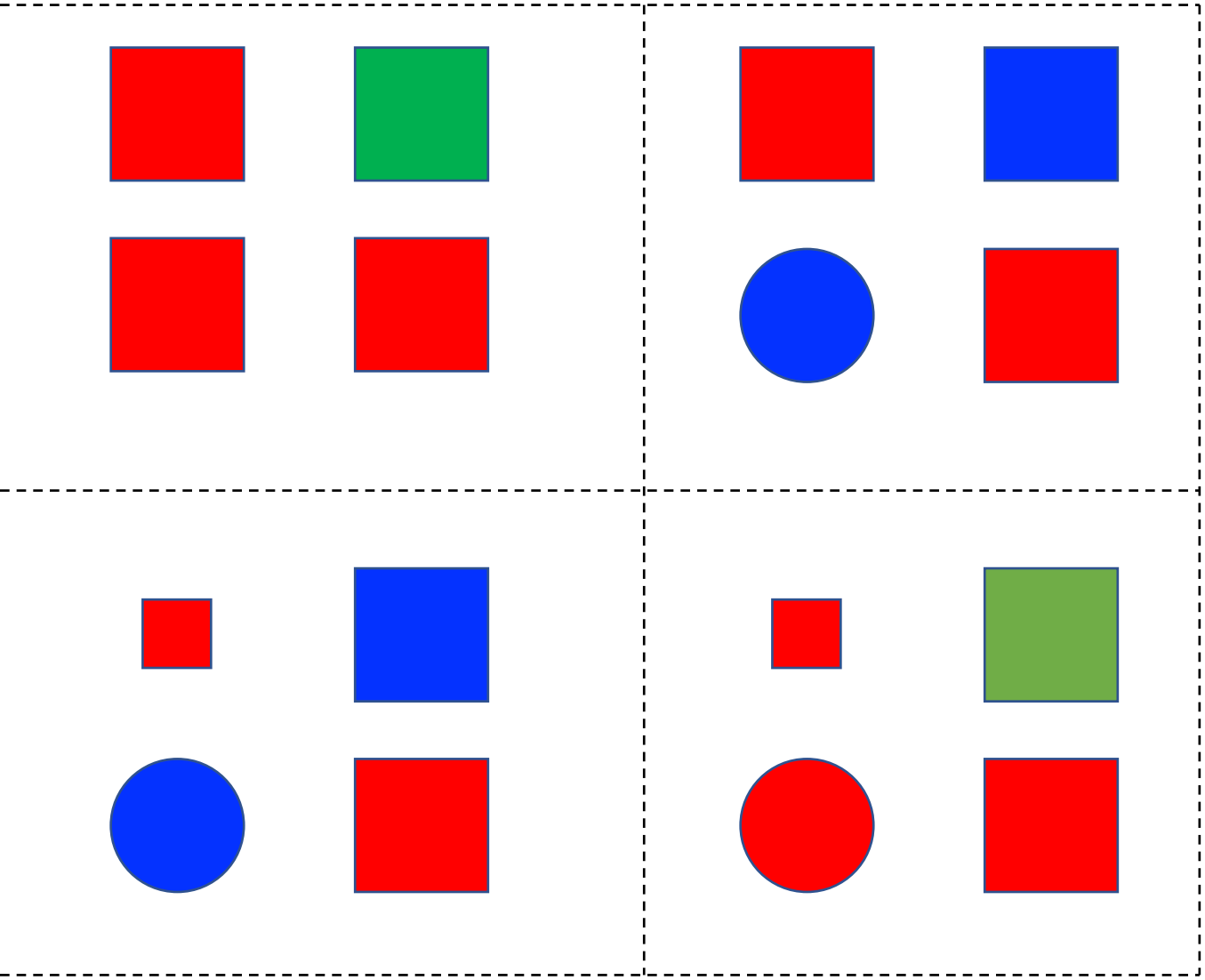

```
% Fig. X is unique wrt property-value V, if there is no ..
unique(X,V) :- property(X,V), not another(X,V).

% .. other figure Y that has the same property/value pair:
another(X,V) :- property(X,V), property(Y,V), X != Y.

% clingo -n0 example4.lp4 unique.lp4
unique(fig1,small)
unique(fig2,green)
unique(fig3,circle)
```

Example 5: … if (almost) everyone is special, who really does stand out? The only “unspecial” or “most normal” figure!

```
property(fig1, red).
property(fig1, box).
property(fig1, small).

property(fig2, green).
property(fig2, box).
property(fig2, large).

property(fig3, red).
property(fig3, circle).
property(fig3, large).

property(fig4, red).
property(fig4, box).
property(fig4, large).
```

```
% Fig. X is unique wrt property-value V, if there is no ..
unique(X,V) :- property(X,V), not another(X,V).

% .. other figure Y that has the same property/value pair:
another(X,V) :- property(X,V), property(Y,V), X != Y.

% Which figure X is special (wrt some property / value)
special(X) :- unique(X,_).
normal(X) :- property(X,_), not special(X).

% clingo -n0 example5.lp4 unique.lp4
unique(fig1,small)
unique(fig2,green)
unique(fig3,circle)
special(fig1)
special(fig2)
special(fig3)
normal(fig4)
```





```
$ source run-wodb.sh

*** Processing example1.lp4 ***
normal(fig1) normal(fig3) normal(fig4)
special(fig2)


*** Processing example2.lp4 ***
normal(fig1) normal(fig2) normal(fig4)
special(fig3)


*** Processing example3.lp4 ***
normal(fig2) normal(fig4)
special(fig1) special(fig3)


*** Processing example4.lp4 ***
normal(fig4)
special(fig1) special(fig2) special(fig3)
```

*Couldn't resist: Voila! Behold the **Meta-WODB** …*

Observations/ Preliminary Conclusions

- A WODB puzzle can have **several** "right" answers.
- To be comprehensible, answers need **justifications**.
- Questions and answers become comprehensible via **queries:**
  - queries **formalize** concepts: what is *meant* by a unique/special/… figure (and what *isn't* meant ..)
  - answers to queries come with **derivations** (a form of *provenance*) which provide the **justification** needed to understand the answer
- Each query generates a possible **solution space.**
- For a given query (i.e., within a solution space), answers can be **clustered** into **equivalence classes**:
  - we disagree which one doesn't belong, because **we care about different properties**
  - … but if a **minimal change** will turn your argument into mine and vice versa ➔ same equivalence class!
- Non-equivalent answers correspond to different **types** of **argument** (not just different preferences)
- **WODB** is a great **educational tool** for CM!
- **ASP** (= Datalog Queries + Generate&Test / Constraints) is a **power tool** for CM!
- **Possible World Explorer (PWE)** combines ASP + Python/Jupyter Notebooks: **CM super powers**!

3. M's Story *(cont'd)*

Table 1. Alternate model for the Which One Doesn't Belong Problem.

POT	IS GOLD	HAS HANDLES	IS LARGE
1	X	X	X
2	X	X	
3	X		X
4		X	X

The unique pot is very easy to spot with this conceptual model; it is the one in which all attributes are True.

Table 2. Example of a Character-taxon matrix. Adapted from Thomer, *et al.*, 2018.

TAXON	HAS 5 FINGERS	HAS FUR	LAYS EGGS
LION	X	X	
LIZARD	X		X
PLATYPUS	X	X	X
ZEBRA		X	

The "binary trick" !?

Is it really *needed* to find the meta-special?
(B has questions or even doubts …)

FYI: There are other solution methods e.g. **FCA** (Formal Concept Analysis)

4 Conclusions

"All models are wrong, some are useful." Wrong is a characterization regarding some truth or gold standard of correctness and Box's adage warns us not to believe any model can express truth. Yet if some models are useful (and alternatively others are not), how can usefulness be defined, measured and validated?

The authors present no answer to this question but hope to begin a discussion, a dialogue to which others are invited. Two models have been presented in the context of a toy problem. Of course both models are wrong; are either of them useful? Can their usefulness be defined? In the case of B's model the usefulness appears to be linked to the query language which provides a justification for the uniqueness of the "normal" item. In the case of M's model the usefulness appears to be limited by the requirement that all properties are binary. Yet, that modeling paradigm is ubiquitous in biological taxonomy classification.

The title of the paper is "Which Model Does Not Belong" which poses the last unanswered question. Assuming we can define the usefulness of the two models, is there a method for probing their equivalence? Is one of these two models more useful than the other or is it impossible to identify which model does not belong?

ACKNOWLEDGMENTS

The authors thank Sahil Gupta who has created analysis tools for studying WODB problems, using his Possible Worlds Explorer tool (Gupta, 2020). In particular, his latest tool can identify the automorphisms of WODB instances.[3]

3. M's Story (Cont'd)

.. B responds with more questions …

Character Matrix

	isLarge	isBox	isRed
1:	0	1	1
2:	1	1	0
3:	1	0	1
4:	**1**	**1**	**1**

Here the matrix works great to find WODB: "1 1 1" stands out!

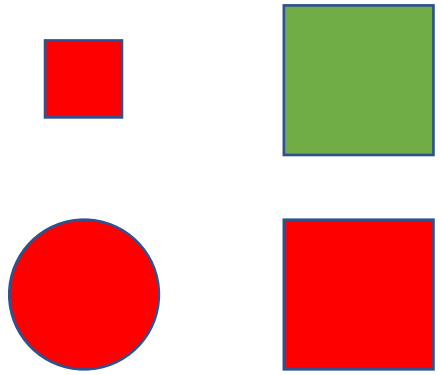
	isSmall	isCirc	isGreen
1:	1	0	0
2:	0	0	1
3:	0	1	0
4:	**0**	**0**	**0**

… still OK: the "0 0 0" stands out …

	isLarge	isCirc	isGreen
1:	0	0	0
2:	1	0	1
3:	1	1	0
4:	1	0	0

*… but what about here? Now #4 does **not** stand out ..*

… 5 more combinations
(can generate all via ASP …)



property(fig1, color, red).
property(fig1, shape, box).
property(fig1, size, small).

property(fig2, color, green).
property(fig2, shape, box).
property(fig2, size, large).

property(fig3, color, red).
property(fig3, shape, circle).
property(fig3, size, large).

property(fig4, color, red).
property(fig4, shape, box).
property(fig4, size, large).

Some References

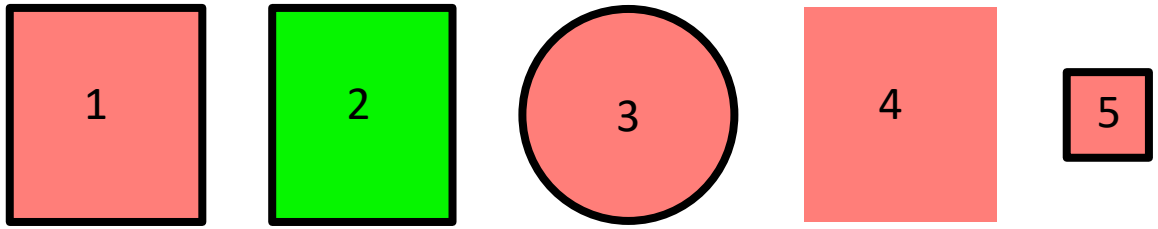
- **WODB/ASP/PWE Source code: https://github.com/idaks/WODB**

- WODB web site: http://wodb.ca/index.html
- A book: https://www.stenhouse.com/content/which-one-doesnt-belong
- Sesame Street: One of these things is not like the others ..
  - Easy: https://www.youtube.com/watch?v=rsRjQDrDnY8
  - A bit more tricky: https://www.youtube.com/watch?v=lqegob0v9W8
- Quite tricky:
  - https://twitter.com/mcnally_gerry/status/1243582829507813376?s=20
  - https://twitter.com/search?q=%23wodb&src=typd

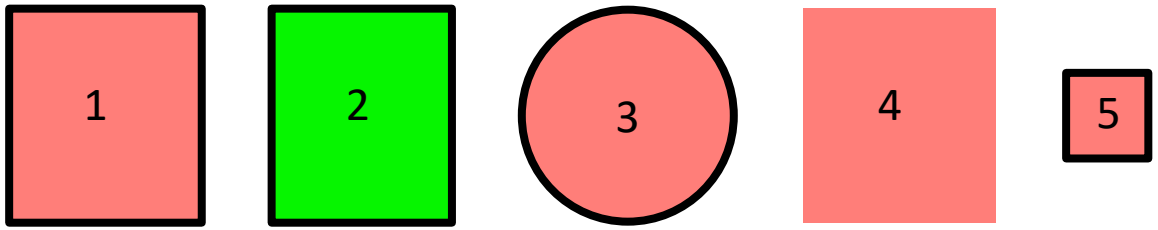
THE END

(additional fun stuff ahead ..)

Enjoy WODB! Challenge Example …



Which one doesn’t belong?



Typical justification:

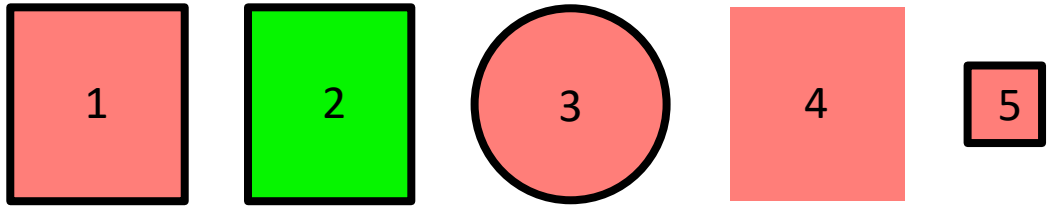
Figure X doesn’t belong because it’s the only figure having property P:

```
special(X)   :- has_prop(X,P), not another(X,P).
another(X,P) :- has_prop(X,P), has_prop(Y,P), X != Y.
```

.. but this doesn’t work here… (why not?)

WODB? FCA to the rescue..

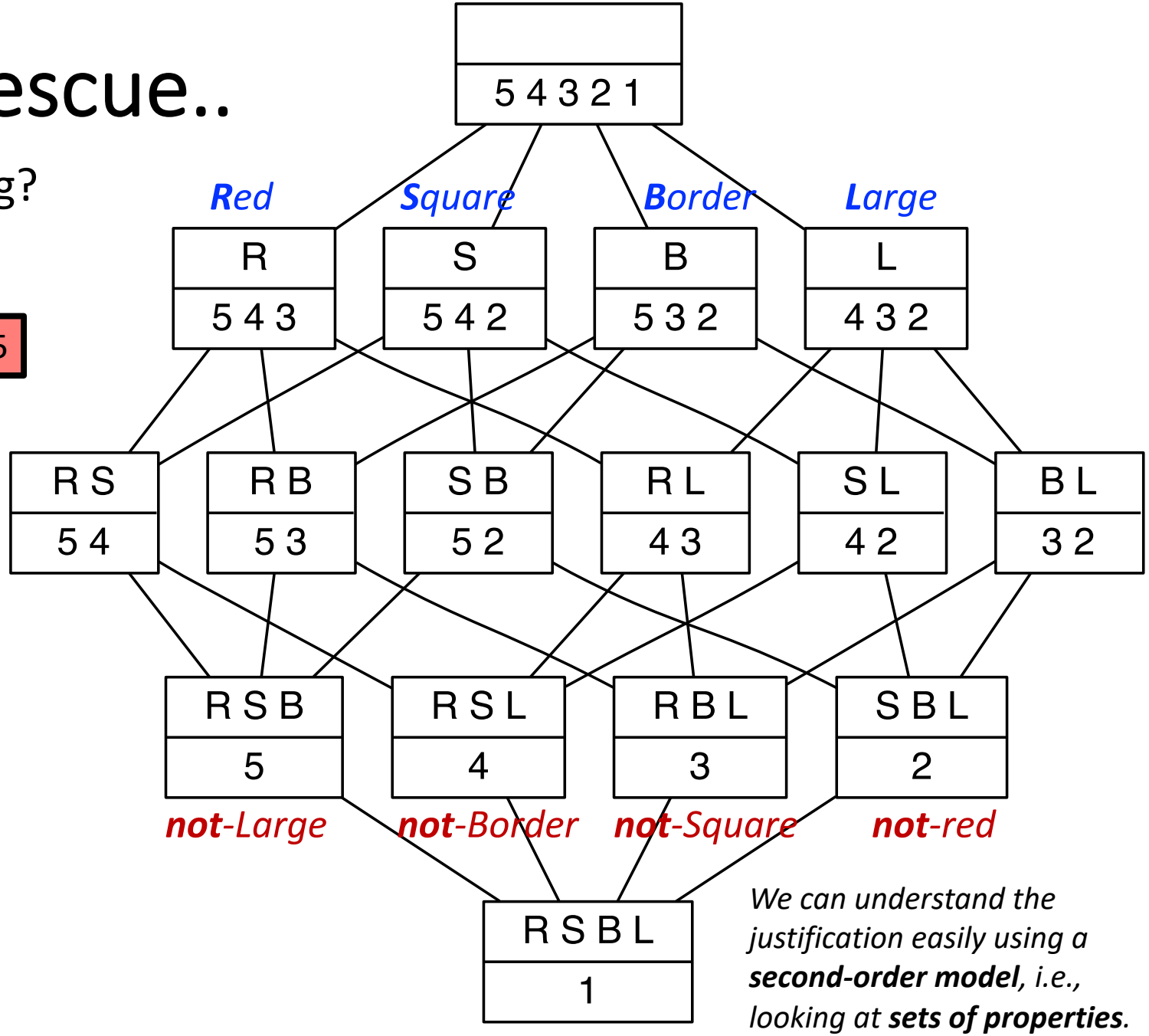
- Which of the 5 figures doesn't belong?



- ***F**ormal **C**oncept **A**nalysis* can tell!

- Figures 2,3,4,5 all have …
  - have 3 out of 4 common properties
  - have 1 out of 4 "special" properties

- Figure 1 is special because it …
  - has 4 out of 4 common properties
  - has 0 out of 4 "special" properties



*We can understand the justification easily using a **second-order model**, i.e., looking at **sets of properties**.*