Towards Relational Theory Formation from Undifferentiated Sensor Data

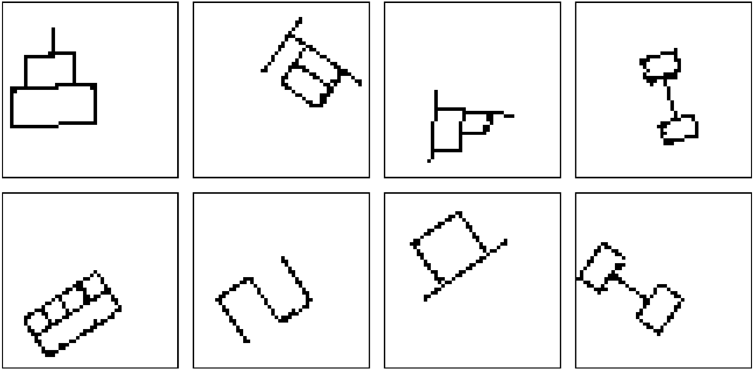
Marc Pickett

Department of Computer Science,
University of Maryland Baltimore County

Advisor:
Dr. Tim Oates

Committee:
Dr. Tim Finin   Dr. Robert Goldstone
Dr. Matt Schmill   Dr. Sergei Nirenburg

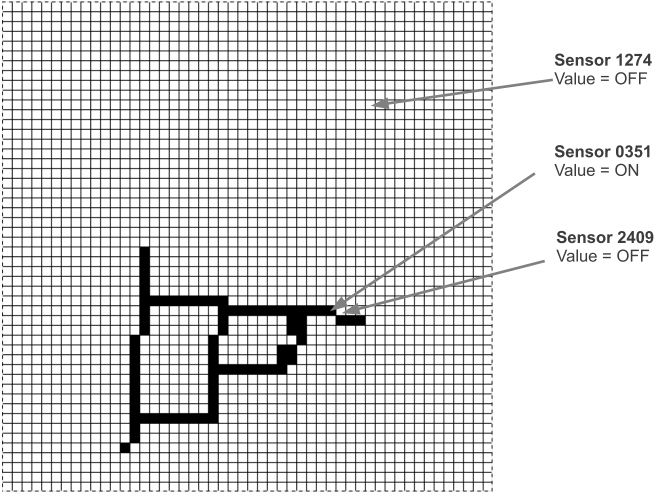
Motivation I



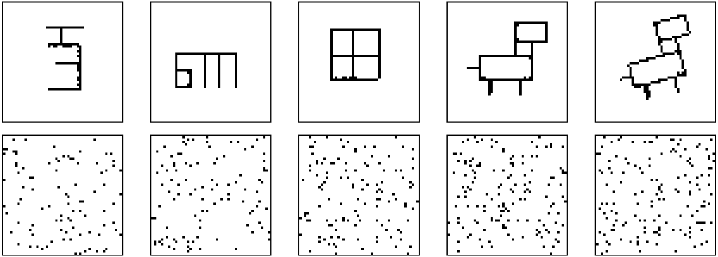
- For vision, people have idea of angle of rotation
- For sound, people have idea of pitch change, speed change
- SPEED-CHANGE in sound *is like* SCALING in vision
  (People can see this **relation**)

Motivation II

Computers start with Raw Sensors



Motivation III



Computer sees this:

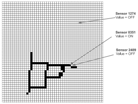
{

S0002=ON,	S0017=ON,	S0048=ON,	S0055=ON,	S0056=ON,	S0117=ON,
S0175=ON,	S0180=ON,	S0197=ON,	S0233=ON,	S0269=ON,	S0284=ON,
S0341=ON,	S0351=ON,	S0404=ON,	S0444=ON,	S0483=ON,	S0490=ON,
S0551=ON,	S0567=ON,	S0573=ON,	S0623=ON,	S0711=ON,	S0729=ON,
S0763=ON,	S0779=ON,	S0798=ON,	S0827=ON,	S0833=ON,	S0859=ON,
S0947=ON,	S0956=ON,	S1027=ON,	S1043=ON,	S1132=ON,	S1137=ON,
S1188=ON,	S1214=ON,	S1239=ON,	S1244=ON,	S1275=ON,	S1305=ON,
S1308=ON,	S1352=ON,	S1395=ON,	S1452=ON,	S1555=ON,	S1572=ON,
S1579=ON,	S1582=ON,	S1631=ON,	S1651=ON,	S1655=ON,	S1771=ON,
S1796=ON,	S1853=ON,	S1891=ON,	S1898=ON,	S1968=ON,	S2059=ON,
S2129=ON,	S2137=ON,	S2161=ON,	S2186=ON,	S2195=ON,	S2206=ON,
S2214=ON,	S2218=ON,	S2227=ON,	S2247=ON,	S2258=ON,	S2308=ON,
S2325=ON,	S2344=ON,	S2355=ON,	S2363=ON,	S2398=ON,	S2406=ON,

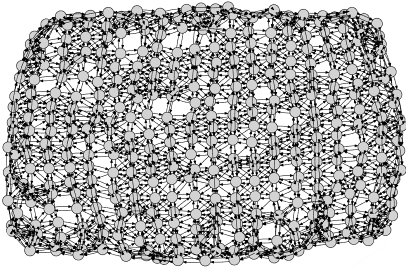
S0000=OFF, S0001=OFF, S0003=OFF, etc.... (all 2,419 other sensors are OFF) }

Motivation IV: Correlations among sensors? (Not Quite)

How to get grid structure?



"Connect each sensor to top 8 most correlated others."
Pierce & Kuipers (1997)

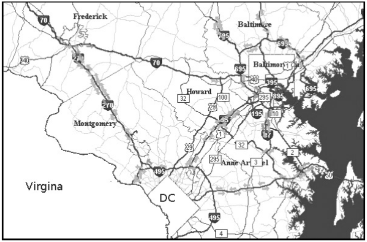


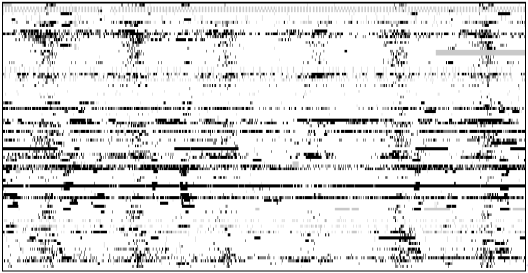
Implicitly gives computer domain knowledge:

- That each node has 8 neighbors
- That domain is 2D
- That domain is spatial

Can't rely on knowledge-engineering

E.g., Highway Traffic speed sensors





Problem, Claims, & Evaluation Criteria

Problem Statement:

How can a computer develop rich relational theories from raw sensor data?

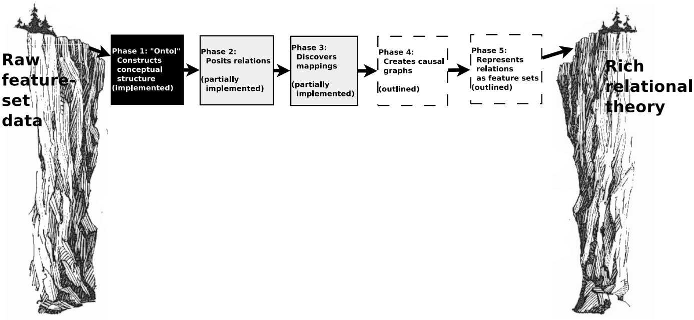
Claims:

1. Partially implemented *design* for bridge from sensors to relational theory
2. 1st link of bridge builds and uses conceptual structures

Evaluation:

1. Bridge story should be **elegant**. We rely on a few principles:
   - **Minimum Description Length** (MDL)
   - "**Signatures**" for recognizing patterns and binding (HMax idea)
   - "**Crunching**" by finding big/frequent overlap to "explain" data
2. System should be **independent of modality** (vision, audio, etc.)
   - Minimal innate knowledge
   - Should work on wide range of domains
   - Might even be in 5 Dimensional world
3. Theory learned by the system should
   - allow for compression of data Wolff (2003)
   - contain concepts *useful* for tasks

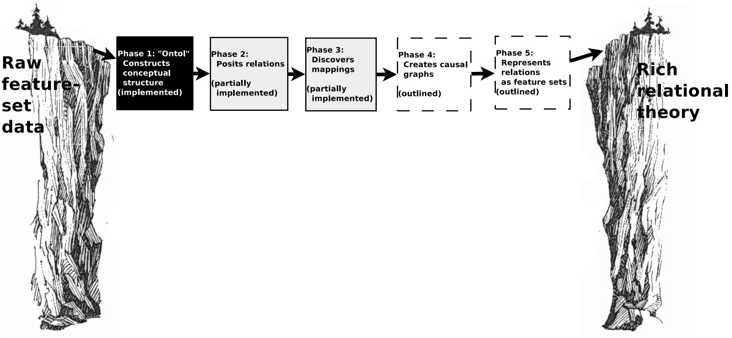
Solution Overview: The Bridge



- All thesis work is building or testing parts of bridge
  - Test on multiple disparate domains
  - Concrete "side applications" along the way
    - Data Compression, Macros in RL, Semi-supervised Learning
  - A few recurring principles: MDL, Signatures, Crunching
- Phase 1 is core of dissertation
- Other phases are bonus

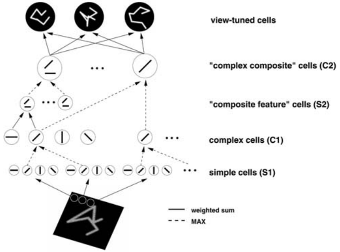
Phase 1:

Creating A Feature-Set Ontology

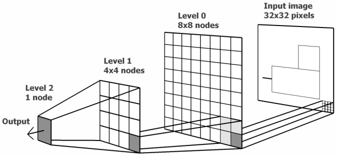


Related Work

HMax
Riesenhuber & Poggio (1999)

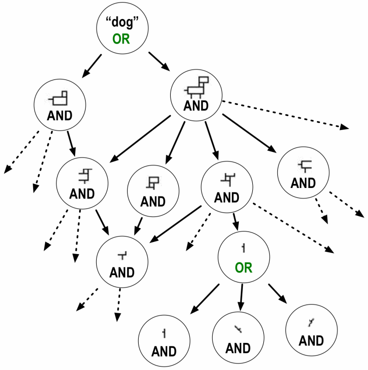


HTMs
Hawkins & Blakeslee (2004)



Neither say how structure is built autonomously

Representation



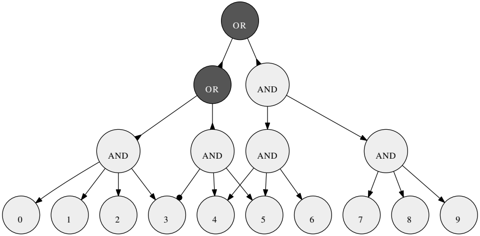
- Like HMax & HTMs
- Can represent invariance
- Uses "signature" idea, like hash or checksum

Parsing and Inference with An Ontology

- "Inference" does bottom-up "abduction" & "top-down" unfolding like HTMs:
  - AND nodes want all their children to be ON
  - OR nodes want at least 1 child ON
  - All nodes want to be "explained" from above
  - (Can have inhibitory connections too)
- "Parse": Minimal* set of ON/OFF node settings to re-infer inputs
- Best parse minimizes "Probabilistic MDL" function

$$E_R(R_i) = -\log_2 \left(P\left(D_i|R_i,\Omega\right)\right) - k\sum_{r\in R_i}\log_2 P\left(r|\Omega\right)$$

- Parse algorithm searches for this
- Optimal Parsing is NP-Hard (proof in thesis)



Building An Ontology: Chunking

Terminology note:

Ontol system that builds and *uses* ontologies
The Cruncher part of Ontol that builds ANDs

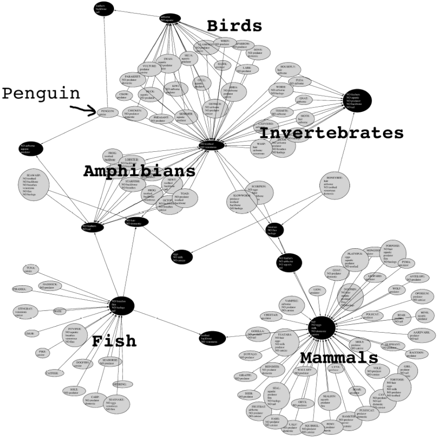
How Cruncher Works:

- Search to minimize $E(\Omega) = k|\Omega| - \sum_i \max_{R_i}\left(\log_2\left(P\left(D_i|R_i,\Omega\right)\right) + k\sum_{r\in R_i}\log_2 P\left(r|\Omega\right)\right)$
- "The Cruncher" does this by recursively squeezing out feature-set overlap. E.g., if
  - S1 = {A, B, C, D, E}
  - S2 = {A, B, C, D, F, G}
  - S3 = {A, B, C, not D, F, G}
  - DL = 17
- Then, new set N1 = S1 ∩ S2 = {A, B, C, D}. Then
  - S1 = {N1, E}, S2 = {N1, F, G}, S3 = {N1, not D, not D, F, G}
  - DL = 14
- Then, new set N2 = S2 ∩ S3 = {N1, F, G}. Then
  - S1 = {N1, E}, S2 = {N2}, S3 = {N2, not D, not D}
  - DL = 13

Cruncher Algorithm:

// Returns an ontology that compactly expresses *S*
Cruncher(*S*) (where *S* is a set of attribute-value sets)
  let *B* be a set of ConceptNodes such that *S*
    foreach attribute-value *A* in *S* there is a corresponding ConceptNode *c* in *B*
      such that $A \in c.hasA$ and $c.isA = \emptyset$.
  while we are still decreasing the description length of *B*
    candidates = findAllIntersections (*B*)
    // score is the potential decrease in description length
    compute score(*B*, *candidate*) **foreach** element in *candidates*
      let *best* be the highest scoring candidate
      if score(*B*, *best*) > 0 **then let** *B* = replaceBest(*B*, *best*) + *best*
  return *B*

Ontology Creation with The Cruncher



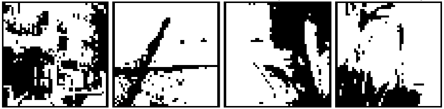
- Cruncher forms taxonomic categories naturally
- Penguin in multiple classes
- Octopus erroneously grouped with Amphibians:

	eggs	aquatic	predator	haslegs	hair	domestic	breathes	toothed	backbone
Octopus	yes	yes	yes	yes	no	no	no	no	no
Amphibians	yes	yes	yes	yes	no	no	yes	yes	yes
Invertebrates	?	?	?	?	?	?	?	no	no

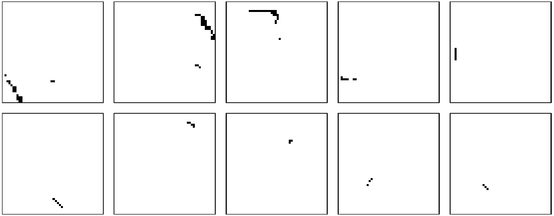
Crunching Patches from Natural Images

- In Zoo ontology, `node102` = {`toothed`, `hair`, `milk`, `4-legs`} (i.e., "Mammal")
- What do concepts for other Crunched feature-set sets look like?

Input: 50x50 Image Patches (undifferentiated)



Concepts Learned



- These concepts are *useful* for describing dataset
- Contiguous chunks
- Cruncher begins with no knowledge of which pixels are next to which

Evaluating Ontol/Cruncher

How to evaluate?

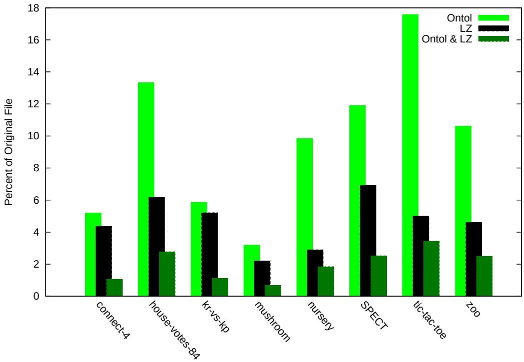
- Can eyeball zoo dataset, but many others too complex
- Primary goal: help span gap between raw sensors and rich theory (difficult to gauge progress)
- Applications: Ontol/Cruncher developed for main goal, but works on "side applications" too:
  - Compression
  - Macro-actions in Reinforcement Learning
  - Semi-supervised Learning

- Test on range of well-known UCI datasets

Dataset	Description	Prediction
`connect-4`	States of Connect 4 boards	**win**, **loss**, **tie**
`house-votes-84`	Congressional voting records	**democrat**, **republican**
`kr-vs-kp`	Chess endgame features	white **win** or **nowin**
`mushroom`	Mushroom features	**poisonous** or **edible**
`nursery`	Nursery schools features	recommendation: **very**, **not**, **priority**, etc.
`SPECT`	Features from cardiac images	**normal** or **abnormal**
`tic-tac-toe`	tic-tac-toe game states	x **win** or **nowin**
`zoo`	Features of animals	**mammal**, **amphibian**, **fish**, etc.

Compression Performance of Ontol/Cruncher

Compression (Lower is Better)

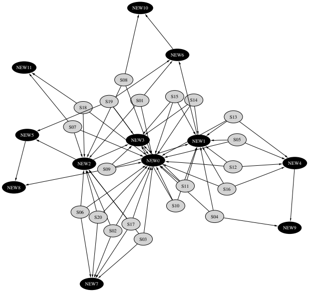


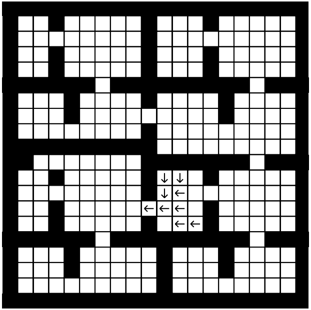
- Ontol not for all text files (just feature-set descriptions)
- Lossless because files are sorted (otherwise add $\log_2(|\mathit{items}|!)$ to specify ordering)
- Cruncher doesn't compress gensyms (so use LZ to do this)

Application: Creating Macro-actions in RL

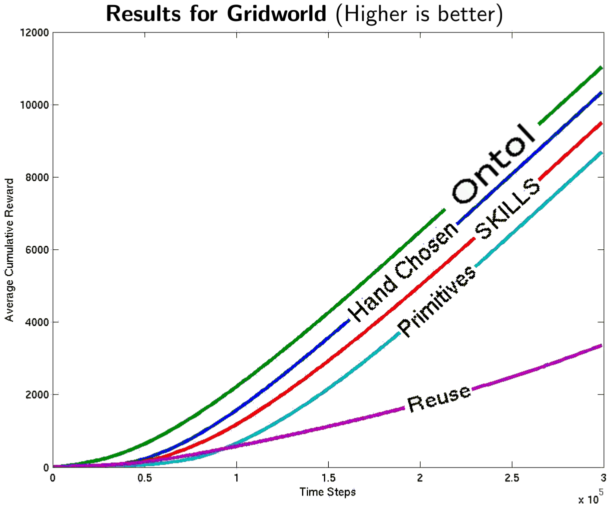
(Pickett & Barto (2002))

- Crunch policies from many MDPs with same structure but different reward
- Use "crunched" subpolicies as policy building blocks or "macro-actions"





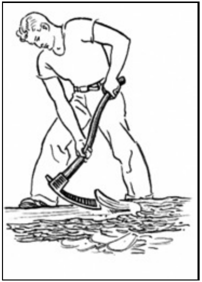
Application: Creating Useful Macro-actions

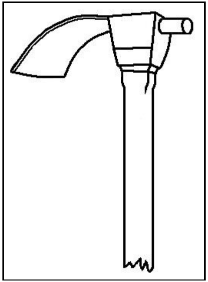


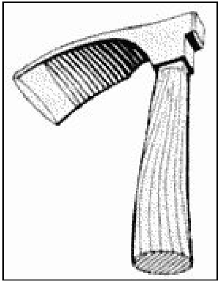
Averaged over 100 runs. Does well on other domains too (see Thesis)

Application: Semi-supervised Learning

An Adze

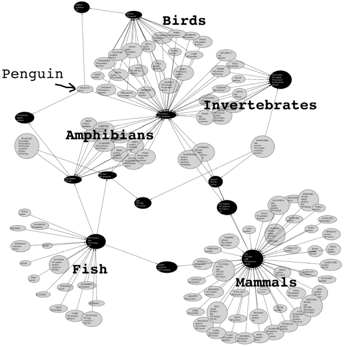






- Related Work: ILP Muggleton (1996)
- Learns from handful of *positive* training instances

How To Learn from a Few Positive Instances



Bayesian Energy Function: $E_{ss}(M) = |D| \log_2 N(M) + |M| \log_2 |U| - \sum_{M_i \in M} \log_2 N(M_i)$

1. Build ontology from unlabeled training set
2. Search for Boolean expression M (which may use nodes in ontology), s.t.
   - M is *True* for all positives
   - M is *False* for negatives (if any)
   - M minimizes $E_{ss}(M)$

(Negatives unnecessary, but *can* be used)

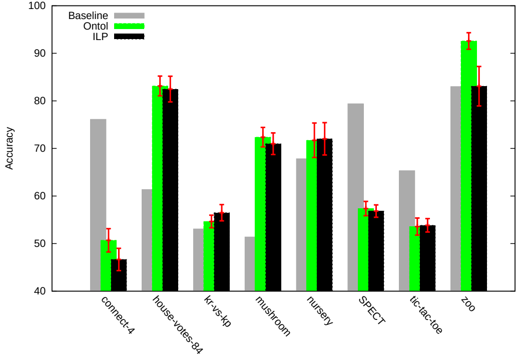
Semi-supervised Experiment Setup

Experiment: given P & N (# of *P*ositive & *N*egative instances to use):
& *unlabeled* training set
& *labeled* training set
& *labeled* testing set

1. Build ontology from *unlabeled* training set
2. Average over 100 trials:
    1. randomly pick class to learn from *labeled* training set
    2. randomly pick P positive instances from class (& N negatives)
    3. search for Boolean expression M to minimize $E_{ss}(M)$
    4. use M to classify testing set

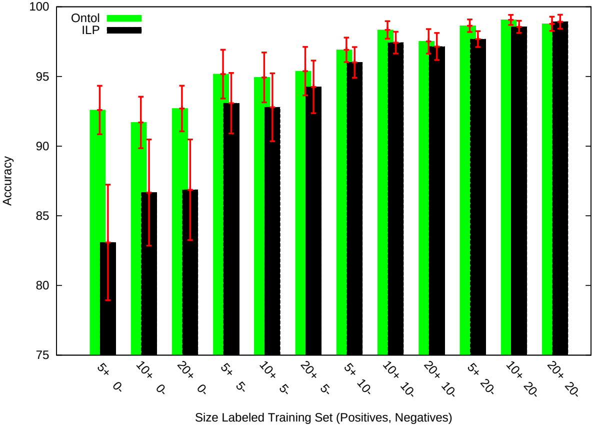
Results from Semi-supervised Learning

Results for $P = 5$ **and** $N = 0$ (Higher is better, Red = 95% conf.)



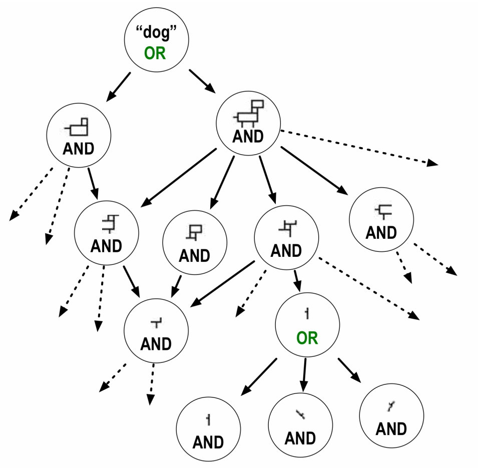
- Ontol significantly outperforms ILP on zoo and connect-4
- Ontol is never significantly worse than ILP
- Underperform Baseline ("everything is most common class")? How?
  Overspecialization:
  E.g., "Mammal": `gorilla`, `monkey`, `chimpanzee`, `orangutan`, `baboon`

Results from Semi-supervised Learning: Increasing Training Size



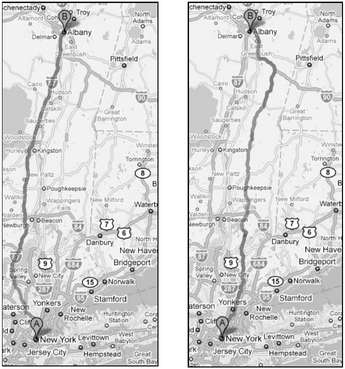
Back to Ontology Building

- Crunching gives **AND**s
- What about **OR**s?



Building ORs: Merging

- Merging finds "OR"s or Equivalence Classes
- Interchangeable concepts form OR
- Find ORs by Crunching context



Building ORs: Merging

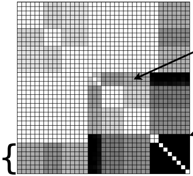
Introduction
A Feature-Set Ontology
Application to Supervised Learning
Parameterized Concepts
Finding Useful Mappings
Apples to Angles
Conclusion
References

Artificial "grammar" for creating bags of features.

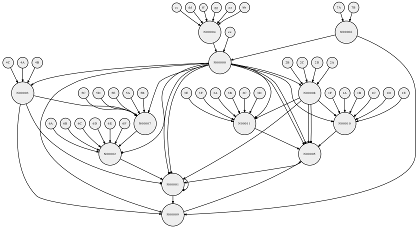
$< S >$	::=	$\{< doubles > < M >\}$
$< M >$	::=	$< A >$ *(with probability .4)* $\mid$ $< B >$ *(with probability .6)*
$< A >$	::=	$< 4 >, < 5 >, < 6 >, < 7 >$
$< B >$	::=	$< 1 >, < 2 >, < 3 >$
$< doubles >$	::=	$< AA > < BB > < CC > < DD > < EE > < FF > < GG >$
$< 1 >$	::=	1A $\mid$ 1B $\mid$ 1C $\mid$ 1D $\mid$ 1E $\mid$ 1F
$< 2 >$	::=	2A $\mid$ 2B $\mid$ 2C $\mid$ 2D
$< 3 >$	::=	3A $\mid$ 3B $\mid$ 3C $\mid$ 3D $\mid$ 3E $\mid$ 3F
$< 4 >$	::=	4A $\mid$ 4B $\mid$ 4C
$< 5 >$	::=	5A $\mid$ 5B $\mid$ 5C $\mid$ 5D $\mid$ 5E
$< 6 >$	::=	6A $\mid$ 6B $\mid$ 6C $\mid$ 6D $\mid$ 6E $\mid$ 6F
$< 7 >$	::=	7A $\mid$ 7B
$< AA >$	::=	aa, $\mid \emptyset$
$< BB >$	::=	bb, $\mid \emptyset$
$< CC >$	::=	cc, $\mid \emptyset$
$< DD >$	::=	dd, $\mid \emptyset$
$< EE >$	::=	ee, $\mid \emptyset$
$< FF >$	::=	ff, $\mid \emptyset$
$< GG >$	::=	gg, $\mid \emptyset$

E.g., {cc, dd, gg, 4C, 5A, 6B, 7B}
{aa, bb, dd, 1F, 2A, 3A}
{cc, dd, ee, 4C, 5A, 6C, 7B}

"Context" as Cooccurrence Matrix

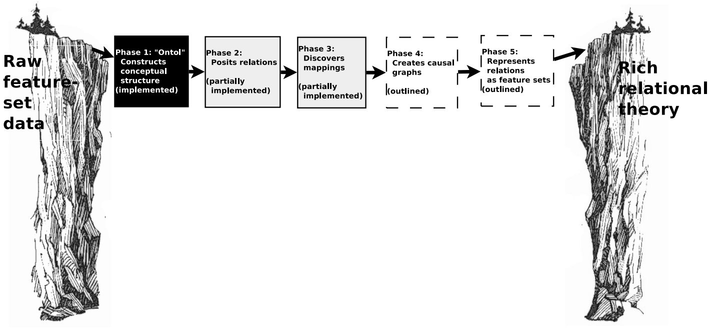


Each row really just bags of features
So, Crunch context to get candidate Equivalence Classes



Will integrate with crunching in future work.

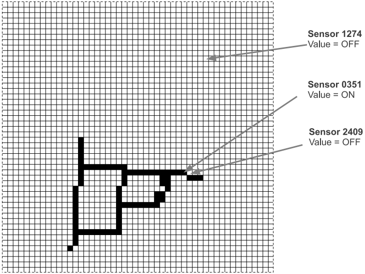
Phase 2: Parameterized Concepts

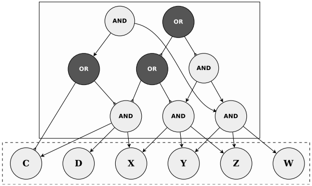


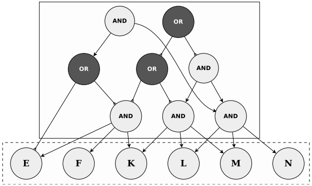
Parameterized Concepts

Merging alone won't give us translation

Need isomorphism or "analogy"

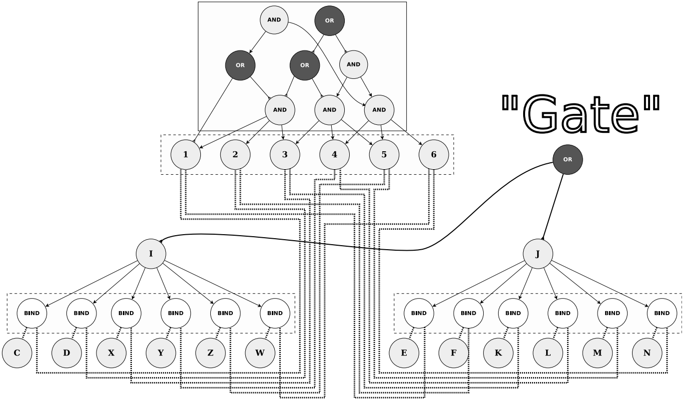






- How to represent parameterized calls?
- How to efficiently find *behaviorally* similar areas?

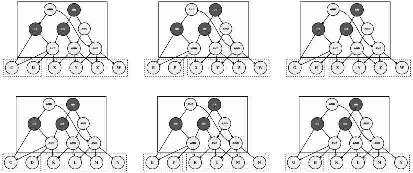
Mechanics of Parameterized Calls

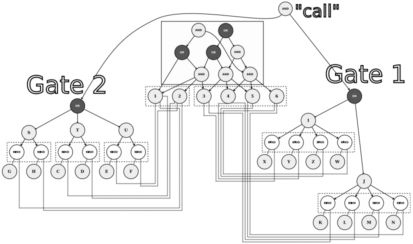


- Extract out overlap
- *Parameterize* differences
- Gate uses **BIND** nodes to control who calls region

Mechanics of Parameterized Calls

Can also represent combinatorics

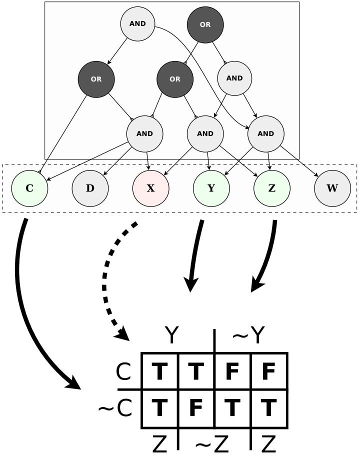




Behavioral Signatures

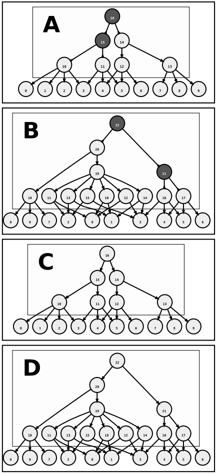
- To find similar regions, create signatures
- Same core idea as representing "dog" shape (HMax)

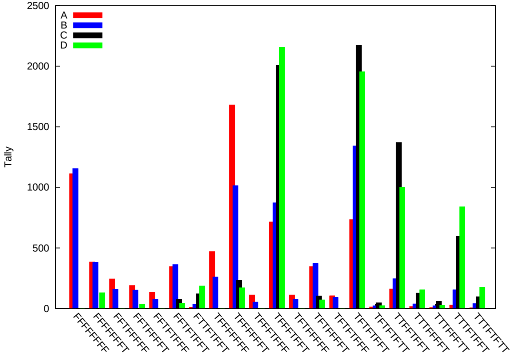
To "sample" from inputs of graph (using sample size = 3):



1. Grab 3 nodes randomly (e.g. C, Y, & Z)
2. Make Karnaugh map for X over all 8 combos of ON/OFF for C, Y, & Z
3. Tally up 1 more for **TTFFTFTT**
4. Canonicalize: Try all 3! orderings of C, Y, Z to find 1st alphabetically
5. (Do steps 2-4 for K-maps for D & W too)
6. GOTO 1 and Repeat

Behavioral Signatures: Results

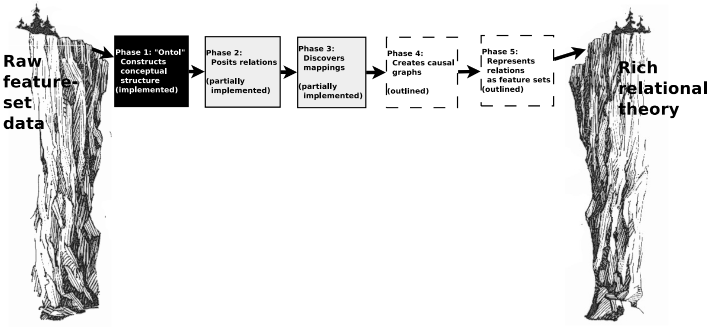




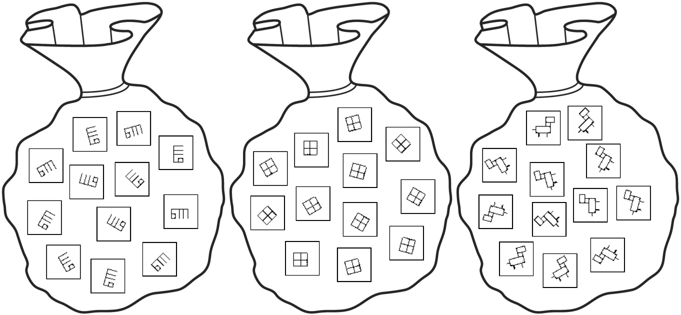
$$
\begin{aligned}
\|A-B\|_2 &= 926,354 & (9.72\%)\\
\|A-D\|_2 &= 9,109,628 & (30.49\%)\\
\|A-C\|_2 &= 9,546,816 & (31.21\%)\\
\|B-D\|_2 &= 5,544,956 & (23.79\%)\\
\|B-C\|_2 &= 5,843,940 & (24.42\%)\\
\|C-D\|_2 &= 301,794 & (5.55\%)
\end{aligned}
$$

Phase 3:

Finding Useful Mappings

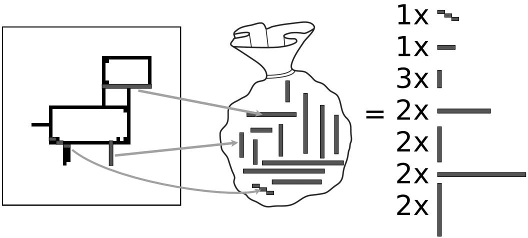


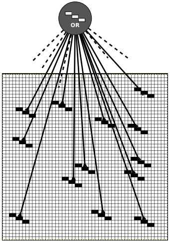
The Problem of finding Mappings



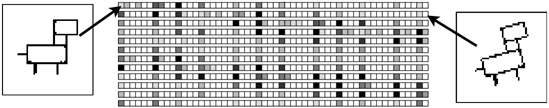
The Problem of finding Mappings II

We assume Phase 2 will give us something like this.





What Makes a Useful Mapping? **MDL!**

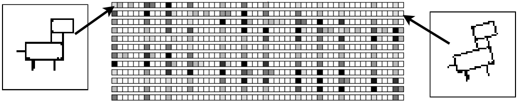


Mapping is set of ordered pairs of features. E.g.:

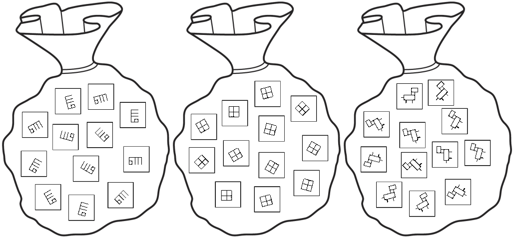
$$
\begin{array}{ccc}
\multicolumn{3}{c}{\textbf{Mapping37}\ (\text{Rotate}15^\circ)} \\
000.05 & \rightarrow & 015.05 \\
000.10 & \rightarrow & 015.10 \\
000.15 & \rightarrow & 015.15 \\
000.25 & \rightarrow & 015.25 \\
005.10 & \rightarrow & 020.10 \\
005.15 & \rightarrow & 020.15 \\
 & \vdots & \\
090.05 & \rightarrow & 105.05 \\
090.10 & \rightarrow & 105.10 \\
090.15 & \rightarrow & 105.15
\end{array}
$$

Dog on right = **Mapping37**(Dog on left) + any residual

Finding Useful Mappings: Algorithm

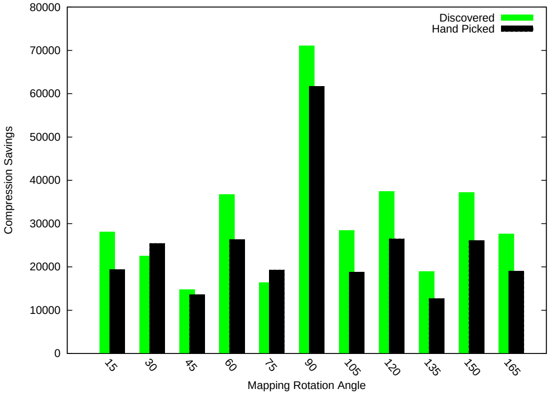


- Find mapping to minimize description length
- Search is 2-pass, like EM:
  - Which features map to which?
  - Which instances map to which?
- Once mapping found, use to reduce DL, then repeat
  - *like Cruncher!*



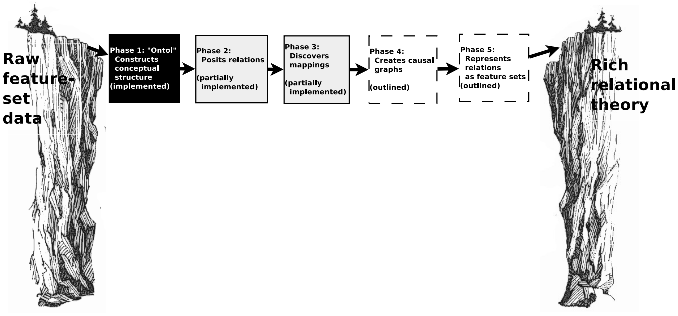
Results: Finding Rotations

Compression Savings Using Discovered Mappings
(Higher is better.)

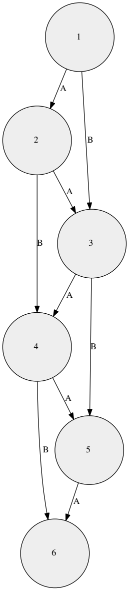


Phase 4:

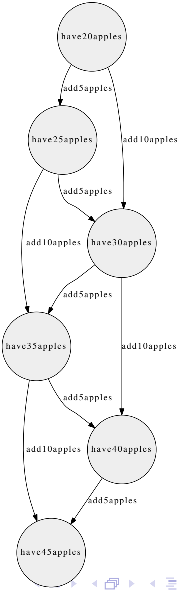
Comparing Apples to Angles



Building & Using Graphs for how Mappings Behave

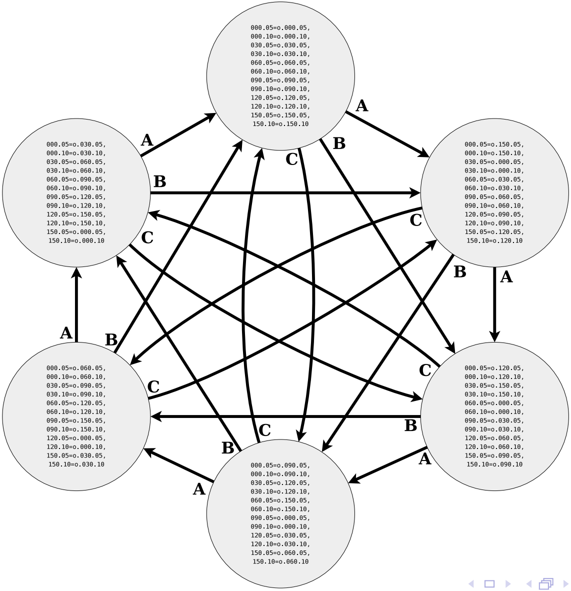


Angles

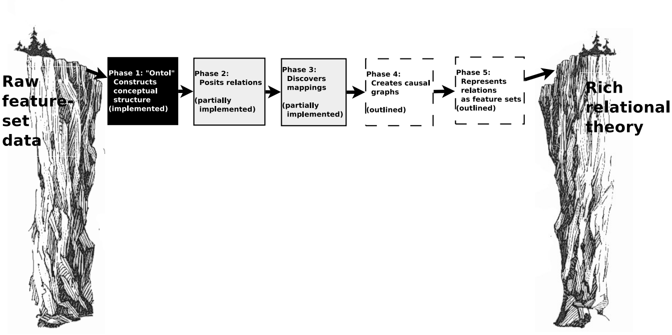


Apples

Building & Using Graphs for how Mappings Behave



Substantiation of Claims



1. Design provides plausible story for getting rich theory from raw sensor data
   - Phases 2 & 3 provide proof-of-concept for how core ideas can bridge gap
   - Phase 2: Representing and Finding Behavioral Isomorphisms
   - Phase 3: Finding and Using Generalized Mappings (e.g., Rotation)
2. Phase 1 creates *useful* structure from feature-sets
   - Does better compression than Lempel-Ziv alone on feature sets
   - Finds useful macro-actions for Reinforcement Learning
   - Learns concepts from a handful of positive training instances

Future Work: Fill In Bridge

- The Speed Prior
- The MacGlashan Transform: Representing Relational Structures as Feature-Sets
- Meta-Cognition: Feeding the Dragon its Tail
- Future work for Phase 1
  - Combined Chunking and Merging
  - Splitting
  - Incremental Learning
  - Wide Signatures and Low Resolution
  - Constraint Satisfaction Search
- Future work for Phase 2
  - Segmentation
  - Munching behavioral signatures
- Future work for Phase 3
  - Finding Primitive Mappings and Minimal Mapping Set
  - Future Application: Using Mappings for Speaker Classification and Identification

References

Hawkins, J., and Blakeslee, S. 2004. *On Intelligence*. Times Books.

Muggleton, S. 1996. Learning from positive data. In Muggleton, S., ed., *Inductive Logic Programming Workshop*, volume 1314 of *Lecture Notes in Computer Science*, 358–376. Springer.

Pickett, M., and Barto, A. 2002. PolicyBlocks: An Algorithm For Creating Useful Macro-Actions in Reinforcement Learning. In *Proceedings of the International Conference on Machine Learning*.

Pierce, D., and Kuipers, B. 1997. Map learning with Uninterpreted Sensors and Effectors. *Artificial Intelligence* 92:169–229.

Riesenhuber, M., and Poggio, T. 1999. Hierarchical Models of Object Recognition in Cortex. *Nature neuroscience* 2(11):1019–1025.

Wolff, J. G. 2003. Information Compression by Multiple Alignment, Unification and Search as a Unifying Principle in Computing and Cognition. *Artif. Intell. Rev.* 19(3):193–230.