The Sigma Knowledge Engineering Environment:

An environment for developing large theories in first- and higher-order logic

Adam Pease, Articulate Software

apease@articulatesoftware.com

http://www.ontologyportal.org/

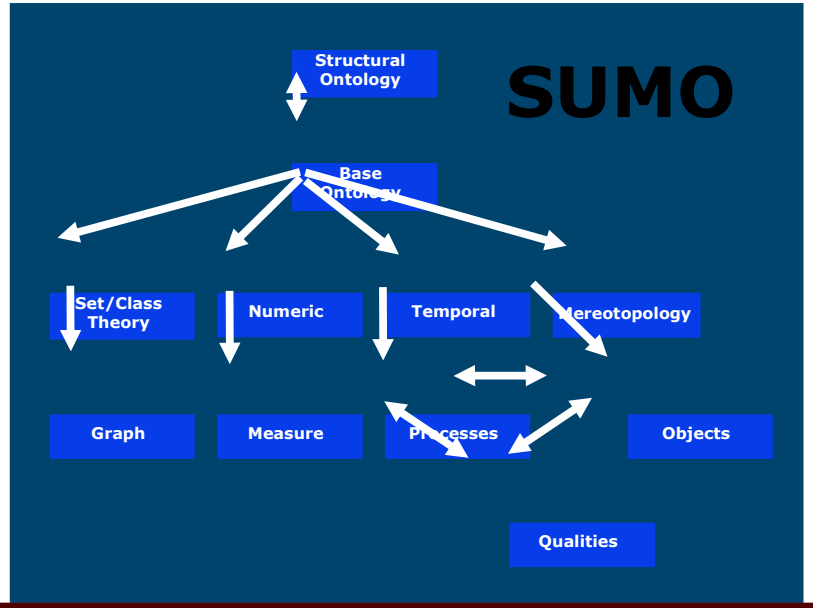
Sigma

- An IDE for SUMO
- Browsing, inference, debugging
- Some information extraction

Suggested Upper Merged Ontology

- 1000 terms, 4000 axioms, 750 rules
- Mapped by hand to all of WordNet 1.6
  - then ported to 3.0
- Associated domain ontologies totalling 20,000 terms and 80,000 axioms
- Mapped to all of YAGO – millions of facts
- Free
  - SUMO is owned by IEEE but basically public domain
  - Domain ontologies are released under GNU
  - www.ontologyportal.org

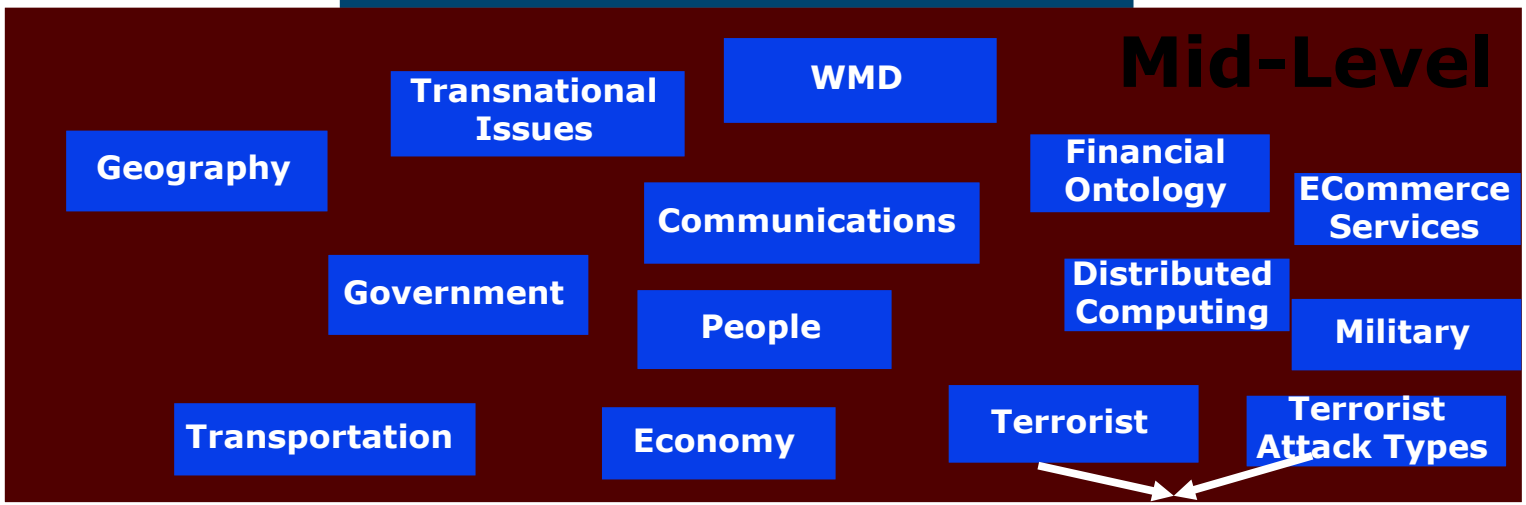
SUMO+Domain Ontology



Total Terms **Total Axioms** **Rules**

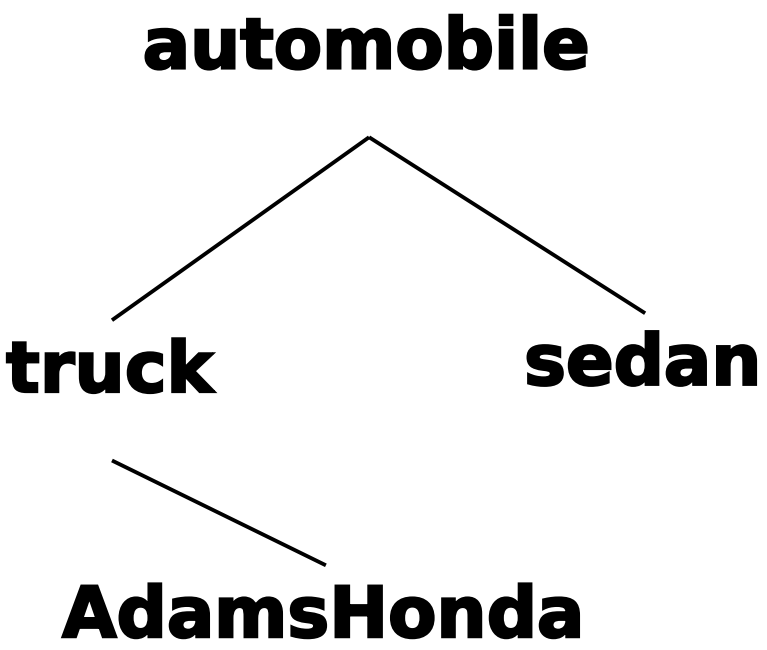
20977 **88257** **4730**

Relations: 1280



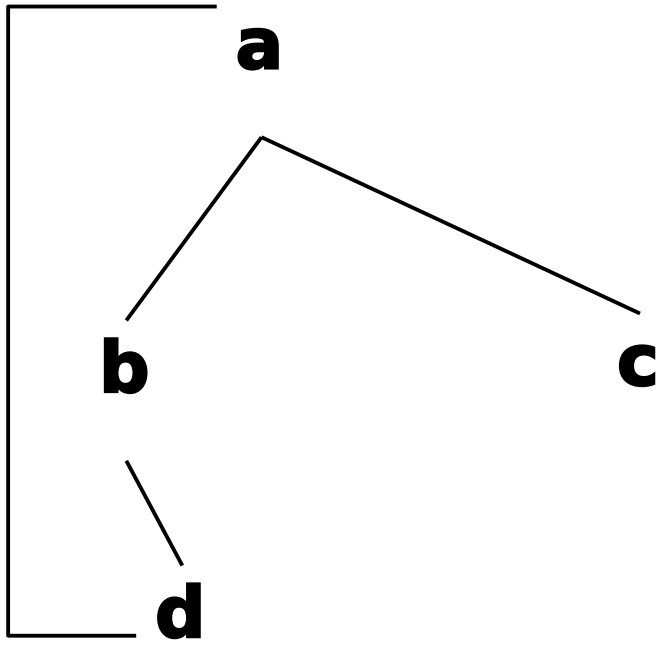
Why Expressive Logic?

Taxonomy



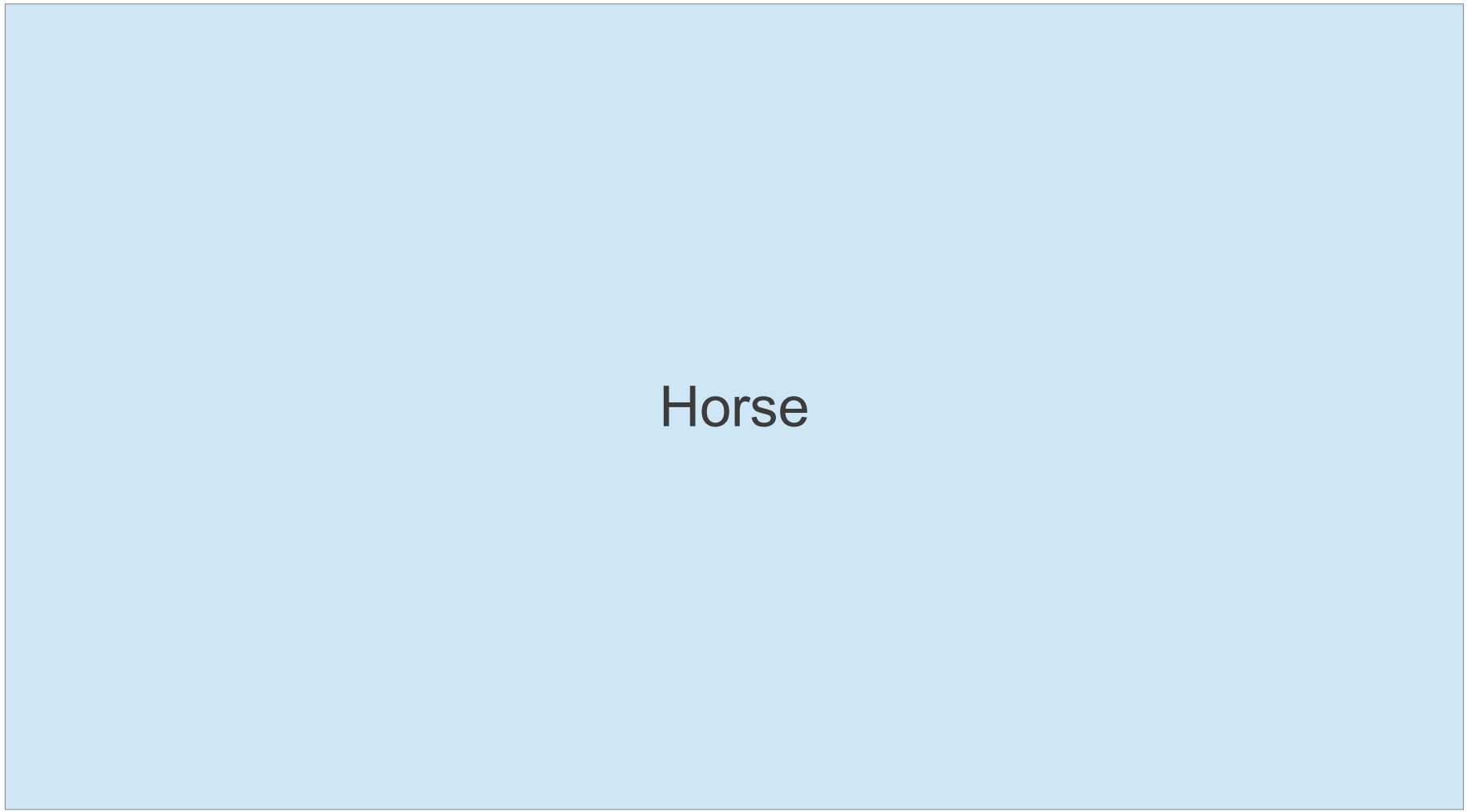
- What's an automobile?
  - truck or sedan
  - Alone it might be taken as not including trucks
  - Does truck include 18-wheelers?

Automation

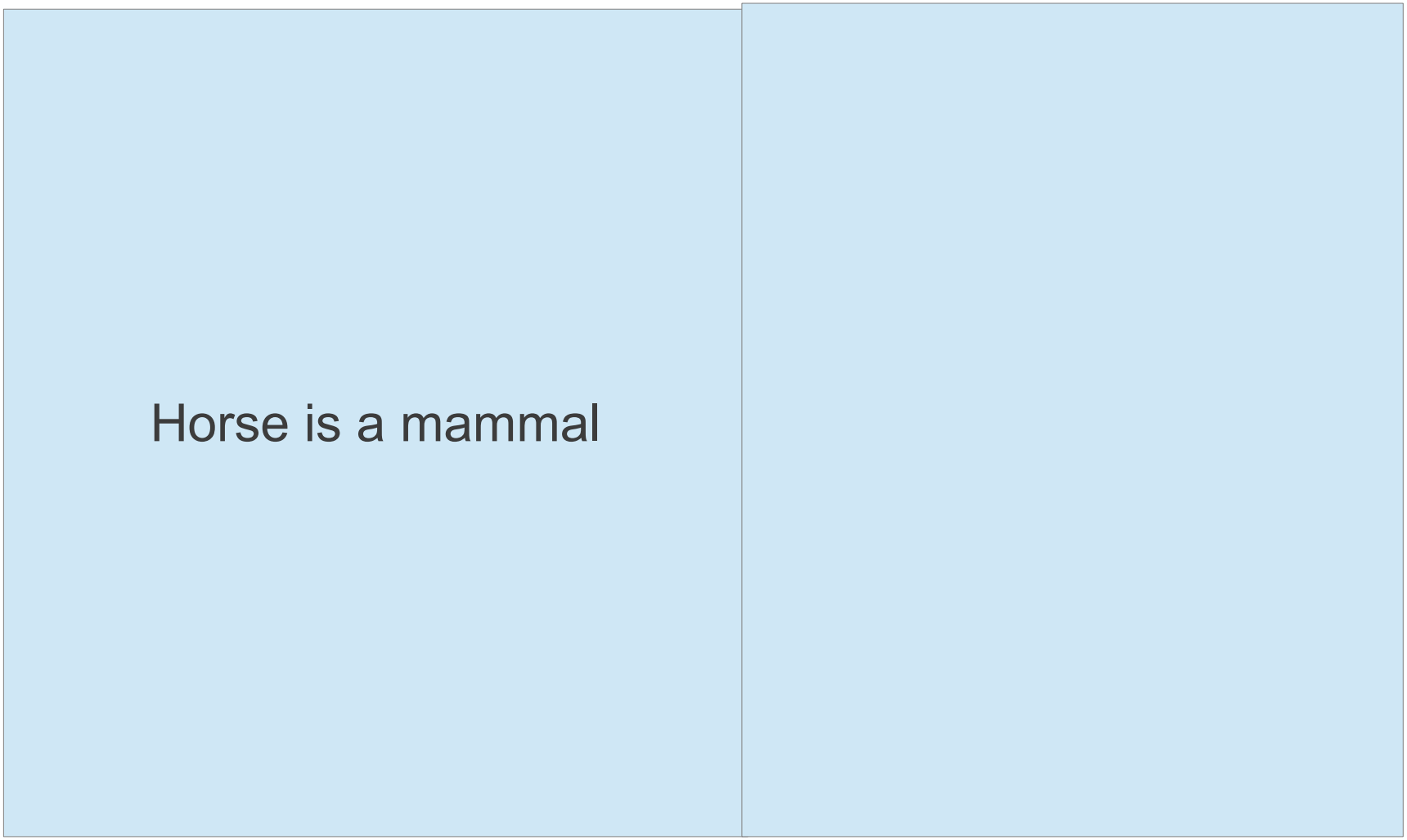


- if d is an a, a can't be a d (usually)

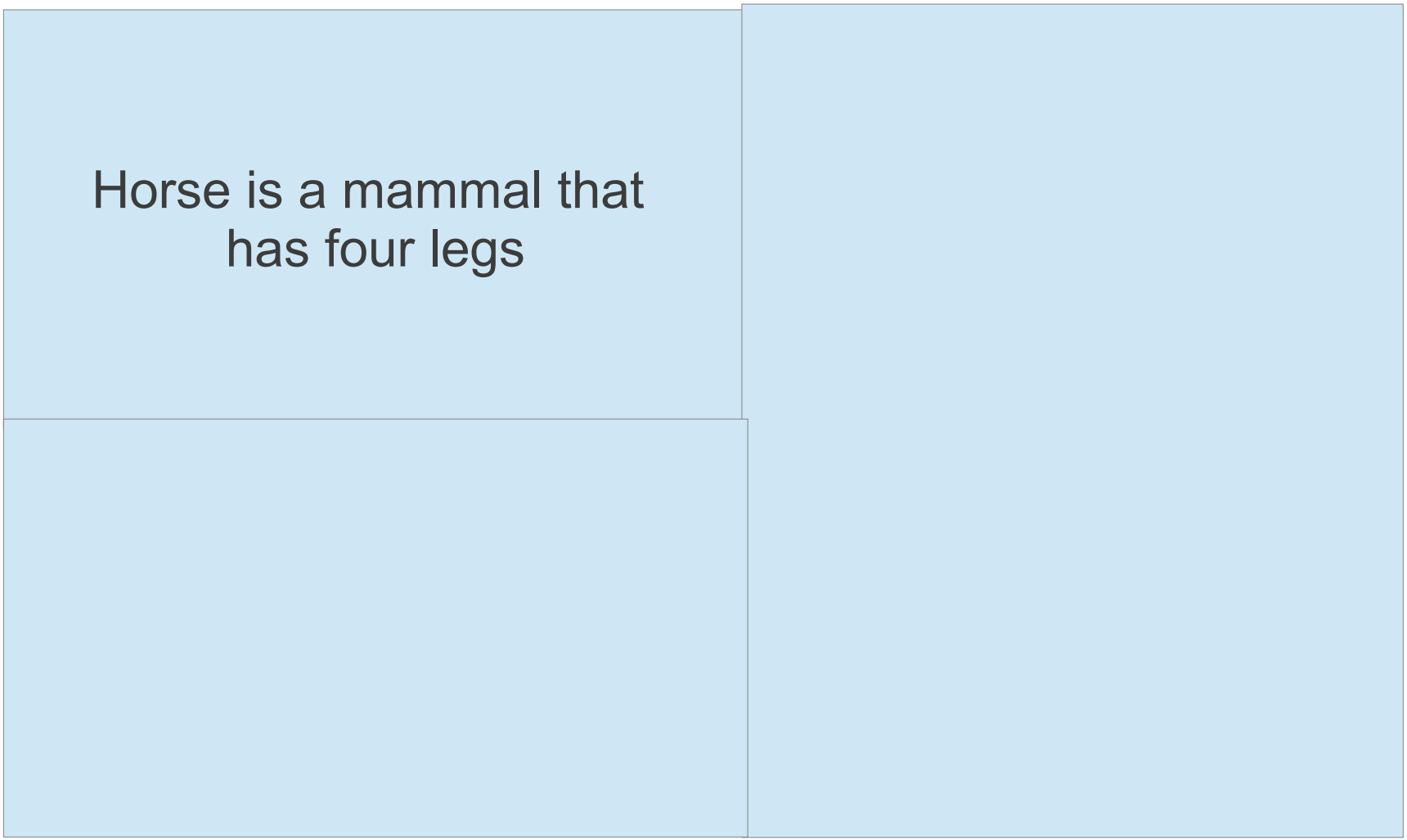
Fixing Meaning



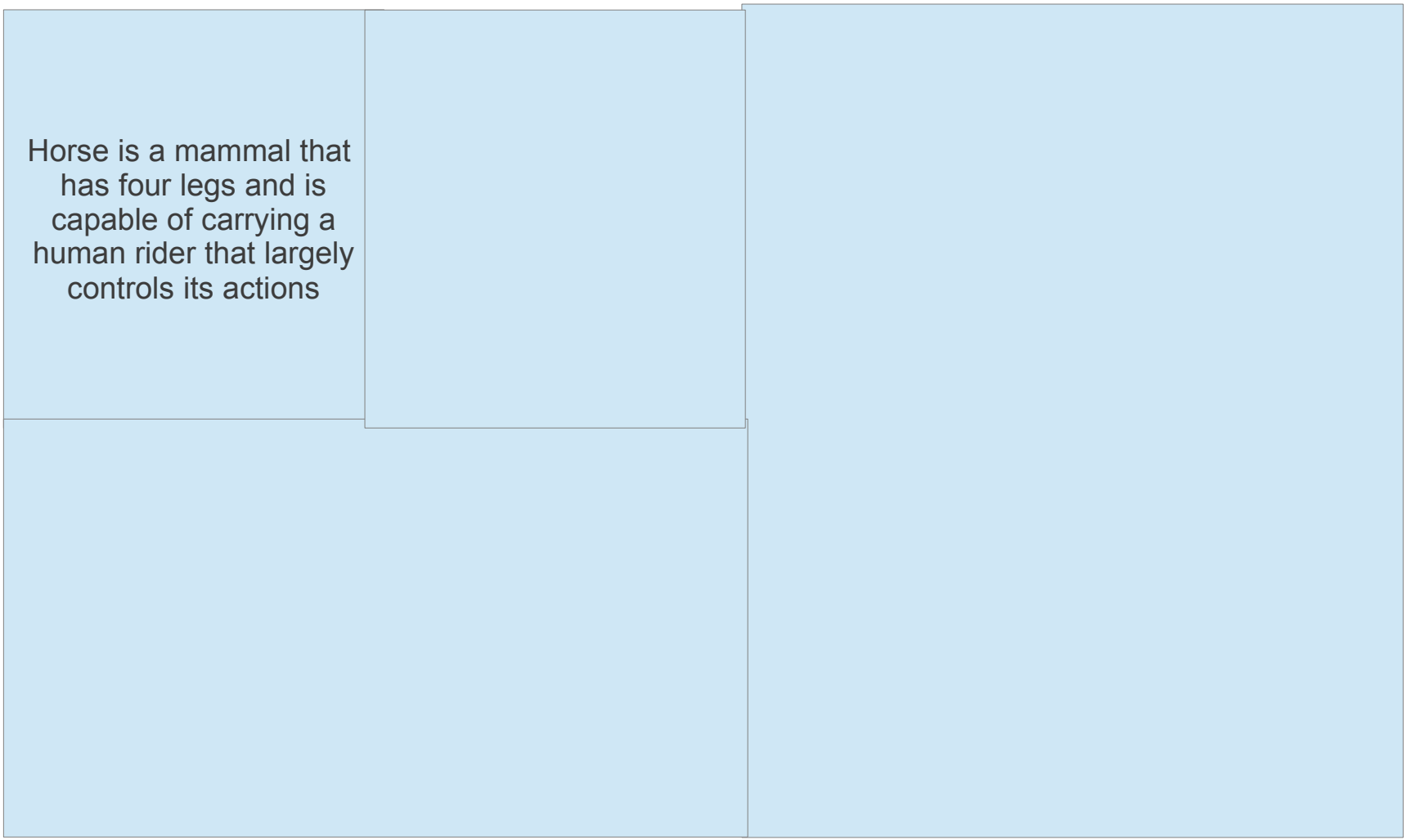
Fixing Meaning



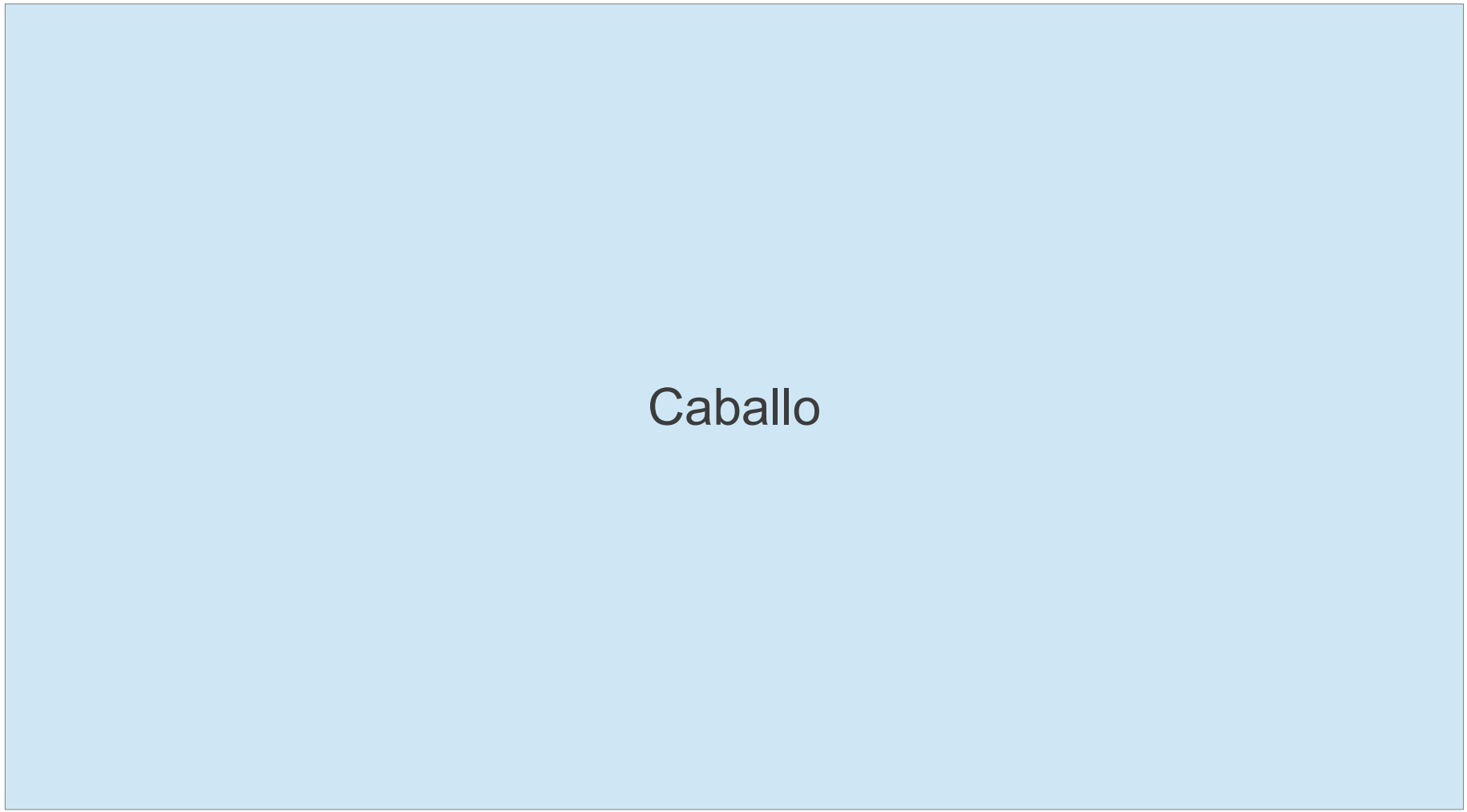
Fixing Meaning



Fixing Meaning



Fixing Meaning



Call it by another name

- But is it the same?
- One might assert the term is the same
  - is it?
- If definitions are shared but shallow, what might be missing?
- If definitions are different are they consistent?
  - How do you determine consistency?

Inferential Closure

- (subclass Horse Mammal)

  (instance Horse MrEd) ->

  (instance MrEd Mammal)
- (=>

  (instance ?X Mammal)

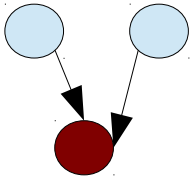
  (exists (?H)

  (and

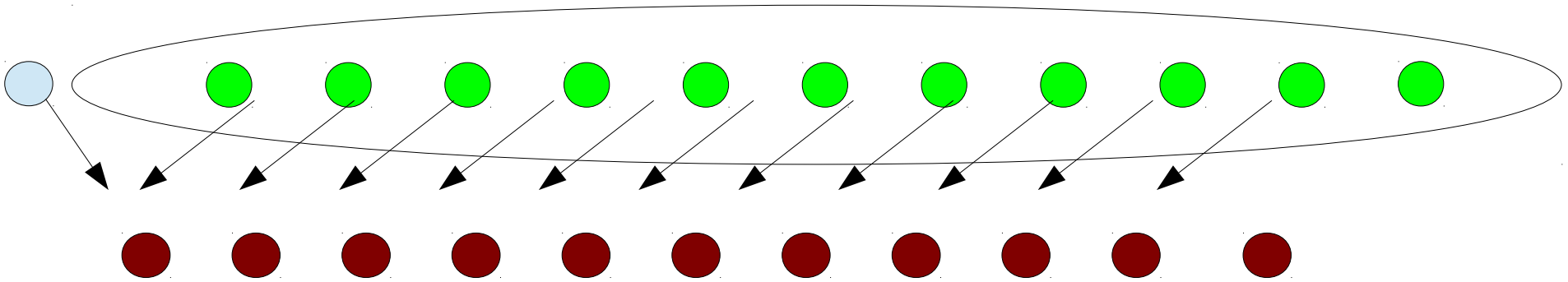
  (instance ?H Head)

  (part ?H ?X))))

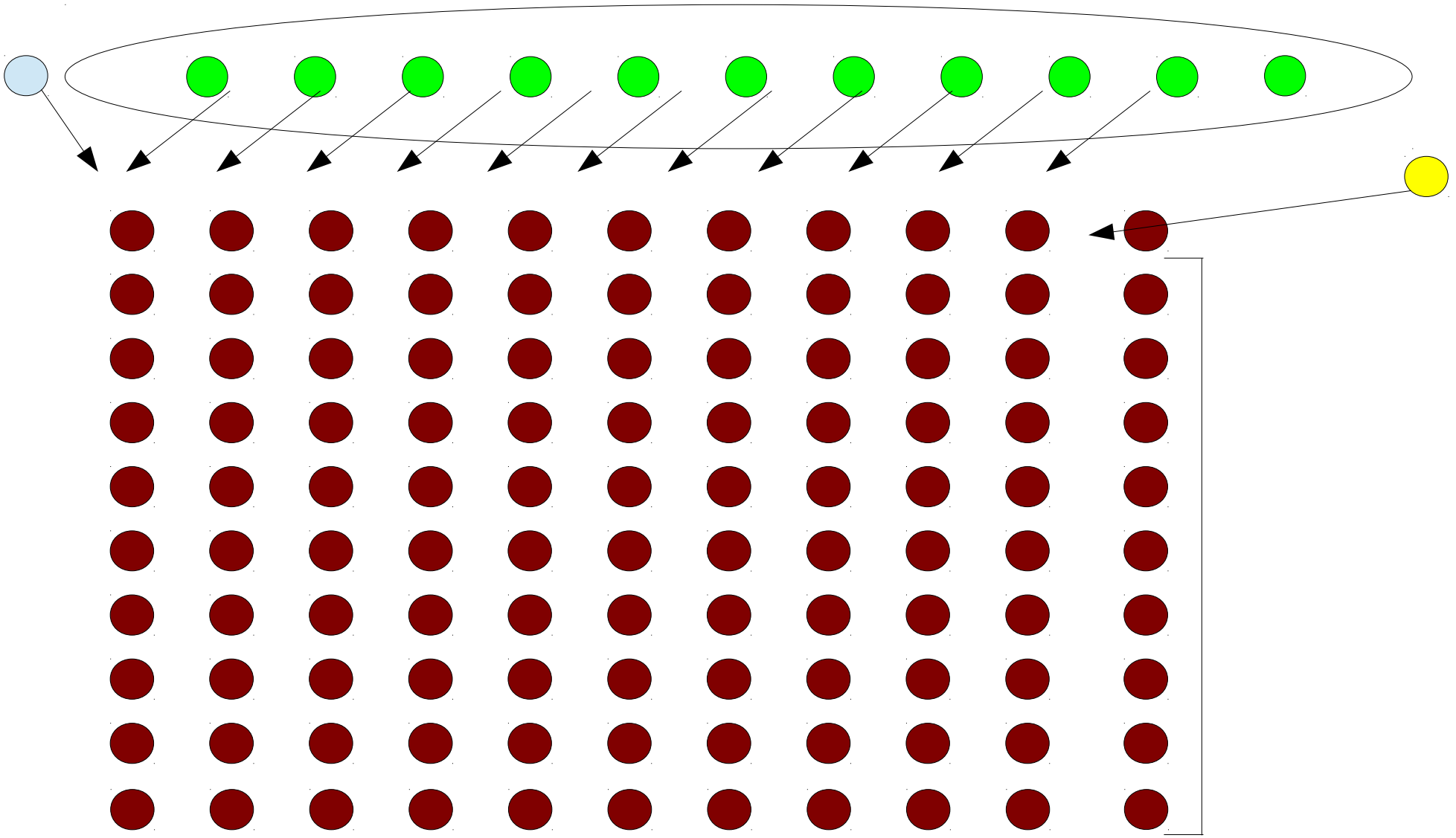
Inferential Closure



Inferential Closure



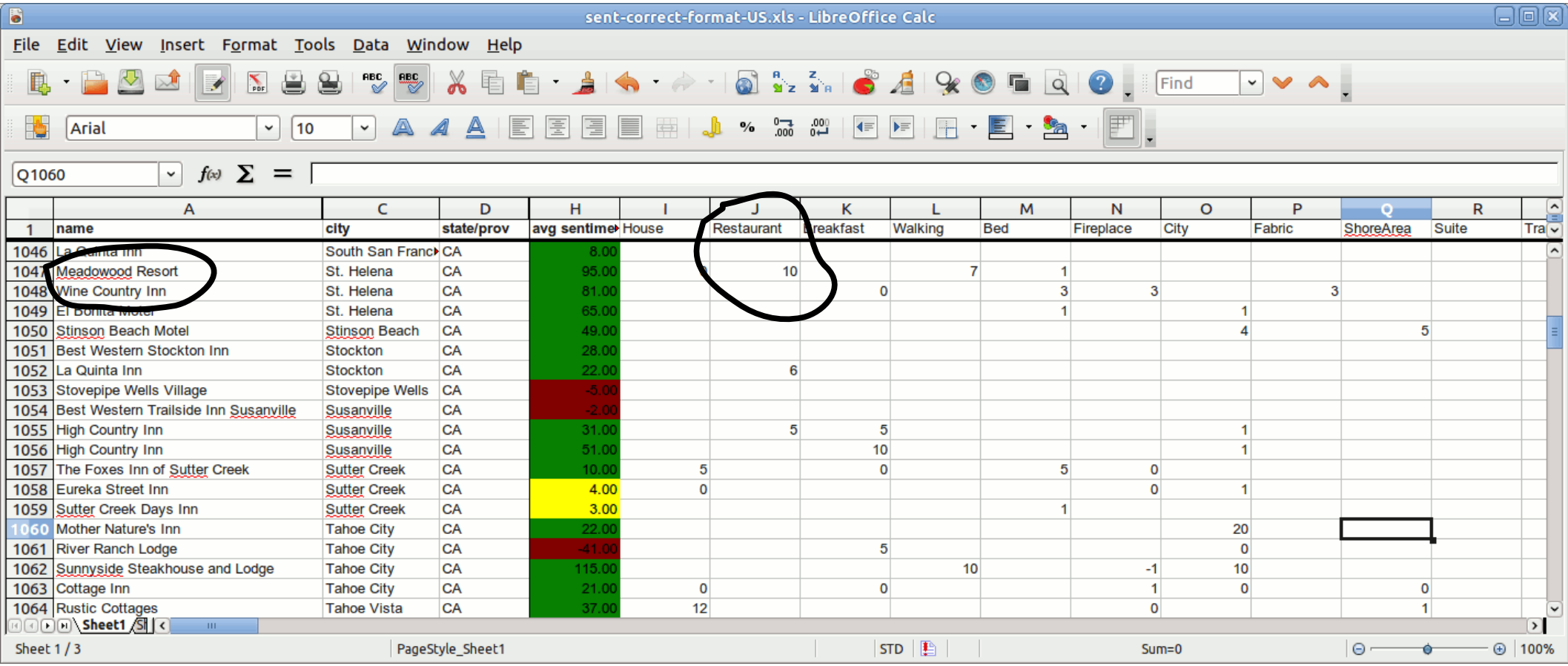
Inferential Closure



Text Processing

Sentiment Analysis

- Emotional content of text
- Pilot project combining
  - Sentiment analysis (computational linguistics)
  - Concept extraction (linguistic semantics/ontology)
- Note this is just a pilot project and the computational linguistic method used is really basic, not state of the art
- Applications:
  - Fine grained search by features
  - Ratings by review, not by stars, and integrated across sources
  - Merge hotel ratings from different services that have different scales by using sentiment



	A	C	D	H	I	J	K	L	M	N	O	P	Q	R
1	name	city	state/prov	avg sentime	House	Restaurant	reakfast	Walking	Bed	Fireplace	City	Fabric	ShoreArea	Suite
1046	La Quinta Inn	South San Franc	CA	8.00										
1047	Meadowood Resort	St. Helena	CA	95.00		10		7	1					
1048	Wine Country Inn	St. Helena	CA	81.00			0		3	3		3		
1049	El Bonita Motel	St. Helena	CA	65.00					1		1			
1050	Stinson Beach Motel	Stinson Beach	CA	49.00							4		5	
1051	Best Western Stockton Inn	Stockton	CA	28.00										
1052	La Quinta Inn	Stockton	CA	22.00		6								
1053	Stovepipe Wells Village	Stovepipe Wells	CA	-5.00										
1054	Best Western Trailside Inn Susanville	Susanville	CA	-2.00										
1055	High Country Inn	Susanville	CA	31.00		5	5				1			
1056	High Country Inn	Susanville	CA	51.00			10				1			
1057	The Foxes Inn of Sutter Creek	Sutter Creek	CA	10.00	5		0		5	0				
1058	Eureka Street Inn	Sutter Creek	CA	4.00	0					0	1			
1059	Sutter Creek Days Inn	Sutter Creek	CA	3.00					1					
1060	Mother Nature's Inn	Tahoe City	CA	22.00							20			
1061	River Ranch Lodge	Tahoe City	CA	-41.00			5				0			
1062	Sunnyside Steakhouse and Lodge	Tahoe City	CA	115.00				10		-1	10			
1063	Cottage Inn	Tahoe City	CA	21.00	0		0			1	0		0	
1064	Rustic Cottages	Tahoe Vista	CA	37.00	12					0			1	

Meadowood, St. Helena: **`Restaurant`**:10

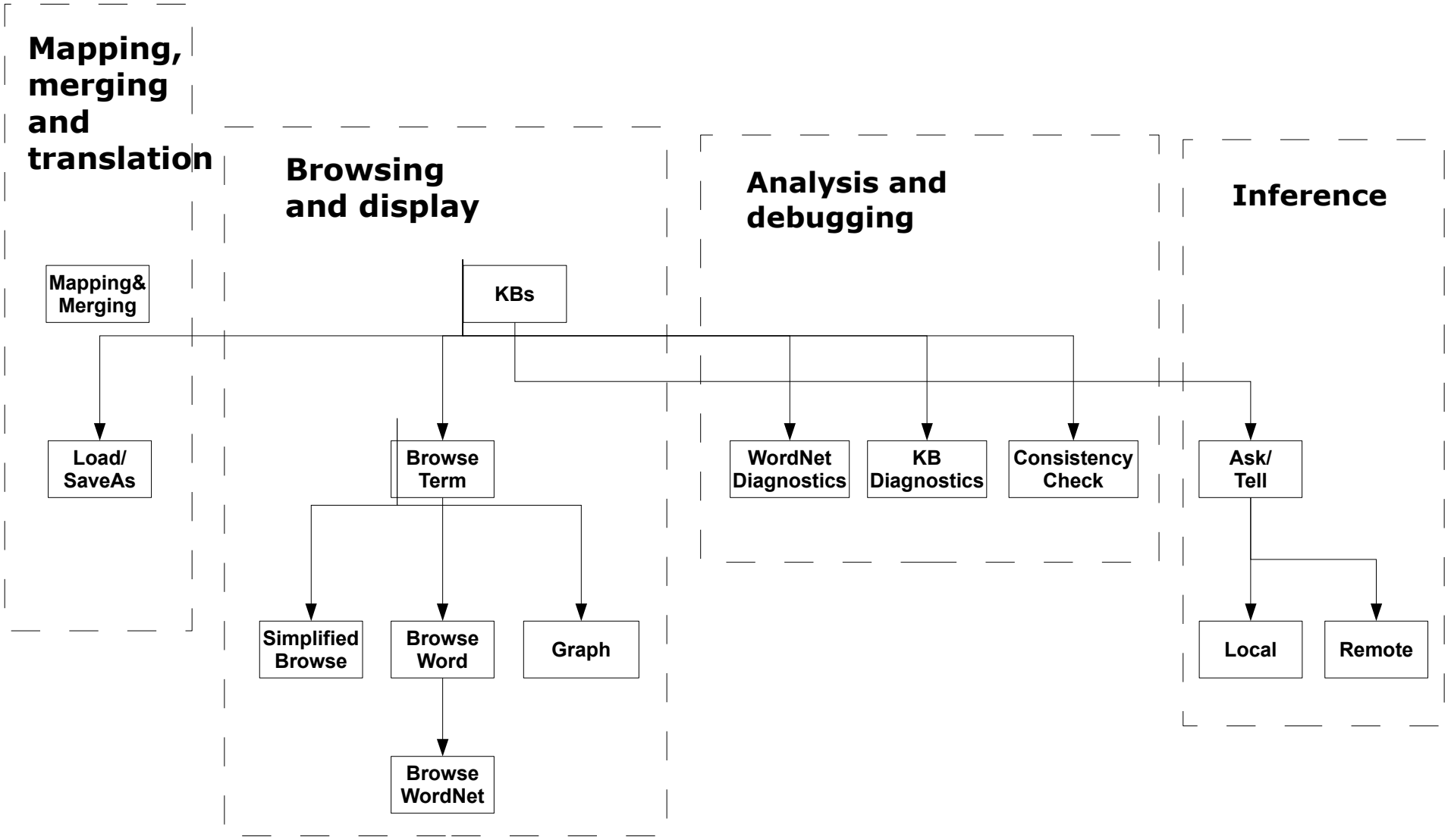
“In recent years the elegant but unstuffy dining room has won rave reviews, becoming a destination restaurant.“

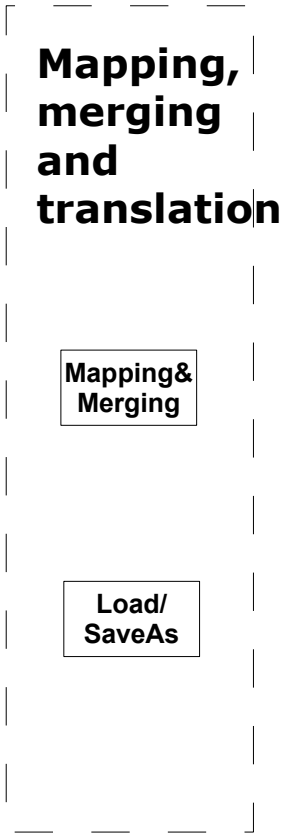
Marys Lake Lodge and Resort, CO: **`Roadway`**: -8

“Not to mention it is very expensive and located in a place that doesn't get much sun so it's icy and cold; and the maintenance of roads is terrible in winter.”

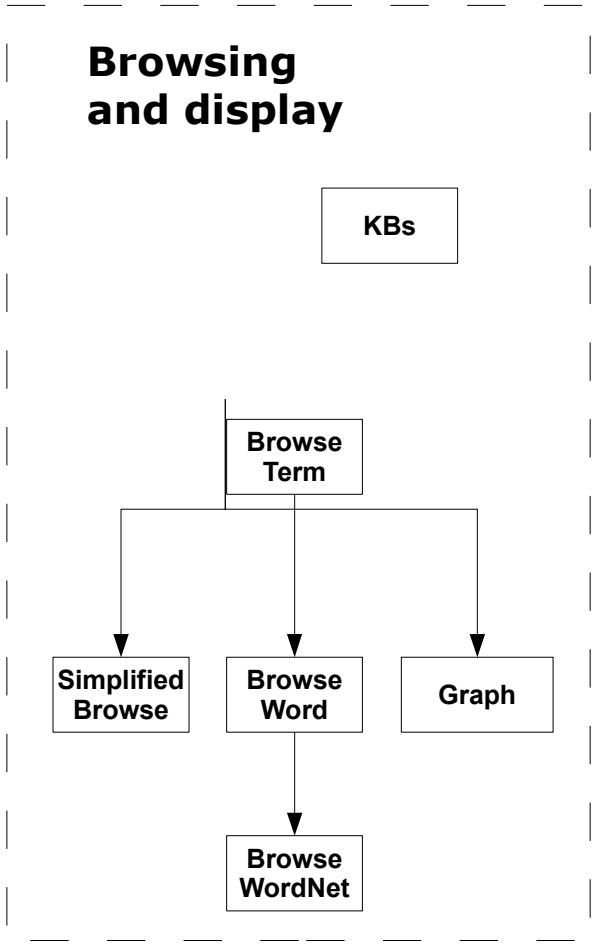
Sigma

Sigma Functions





- Simple string distance-based merging tool
  - More complicated algorithms seemed to have little practical effect
  - Most of the value was in a convenient GUI
  - Most ontologies to be merged have so little to match on
- Supported Languages
  - SUO-KIF
  - OWL
  - Prolog
  - TPTP
  - THF


Browsing
and display
KBs
Browse
Term
Simplified
Browse
Browse
Word
Graph
Browse
WordNet

Sigma knowledge engineering environment
Browsing Interface

[Home | Graph |]
KB: SUMO **Language:** EnglishLanguage

KB Term: Walking Show
English Word: Noun Show

Walking (walking)


Rollerblade, afoot, amble, ambulate, ambulation, angry_walk, backpack, break, bumble, canter, careen, circumambulate, clamber, climb, climb_up, clomp, clump, cock, coggle, constitutional, constitutionalize, countermarch, crab, creep, curvet, dash, debouch, dodder, dogtrot, drag, dressage, drift, err, escalade, exhibit, falter, fast_break, file, file_in, file_out, fire_walking, flounce, flounder, foot, footer, footslog, footstep, forage, gait, gallop...

appearance as argument number 1

(documentation Walking EnglishLanguage "ambulating relatively slowly, i.e. moving in such a way that at least one foot is always in contact with the ground.")	Merge.kif 8825-8826	
(externalImage Walking "http://upload.wikimedia.org/wikipedia/commons/0/ 0f/ Robotpeintre.gif")	pictureList.kif 3030-3030	externalImage walking and "http://upload.wikimedia.org/wikipedia /commons/0/0f/ Robotpeintre.gif"
(externalImage Walking "http://upload.wikimedia.org/wikipedia/commons/6/ 6f/ Walk-Cycle.gif")	pictureList.kif 3276-3276	externalImage walking and "http://upload.wikimedia.org/wikipedia /commons/6/6f/ Walk-Cycle.gif"
(externalImage Walking "http://upload.wikimedia.org/wikipedia/commons/d/ d2/ Marcheur_en_comp%C3%A9tion.jpg")	pictureList.kif 3277-3277	externalImage walking and "http://upload.wikimedia.org/wikipedia /commons/d/d2/ Marcheur_en_comp%C3%A9tion.jpg"
(subclass Walking Ambulating)	Merge.kif 8824-8824	Walking is a subclass of ambulating

appearance as argument number 2

(partition Ambulating Walking Running)	Merge.kif 8819-8819	Ambulating is exhaustively partitioned into walking and running
(subclass Wading Walking)	Mid-level-ontology.kif 236-236	Wading is a subclass of walking
(termFormat EnglishLanguage Walking "walking")	english_format.kif 792-792	term format english language, walking and "walking"

antecedent

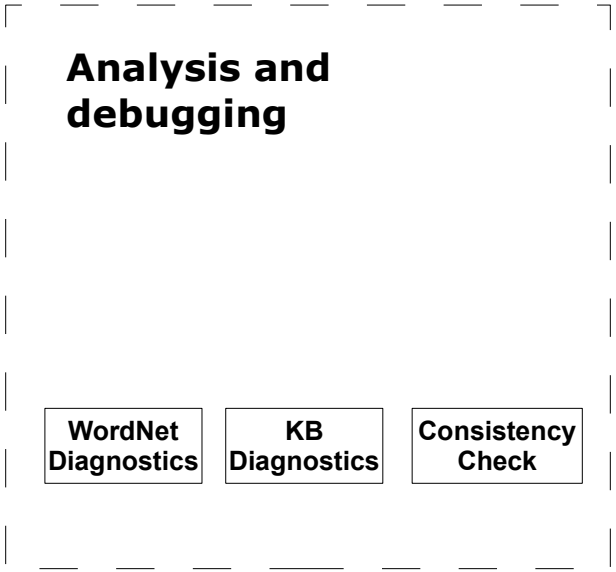
```
(=>
   (and
      (instance ?WALK Walking)
      (instance ?RUN Running)
      (agent ?WALK ?AGENT)
      (agent ?RUN ?AGENT)
      (holdsDuring
         (WhenFn ?WALK)
         (measure ?AGENT
            (SpeedFn ?LENGTH1 ?TIME)))
      (holdsDuring
         (WhenFn ?RUN)
         (measure ?AGENT
            (SpeedFn ?LENGTH2 ?TIME))))
   (greaterThan ?LENGTH2 ?LENGTH1))
```

Merge.kif 8833-8841

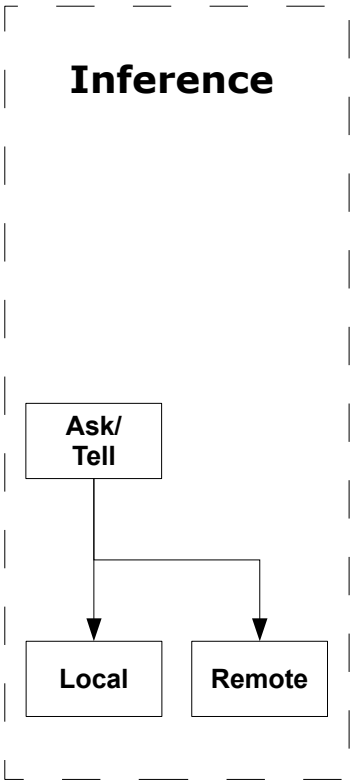
- If a process is an instance of walking and process is an instance of running and an agent is an agent of process and agent is an agent of process and the measure of agent is a length measure per a time duration holds during the time of existence of process and the measure of agent is length measure per time duration holds during the time of existence of process
- then length measure is greater than length measure

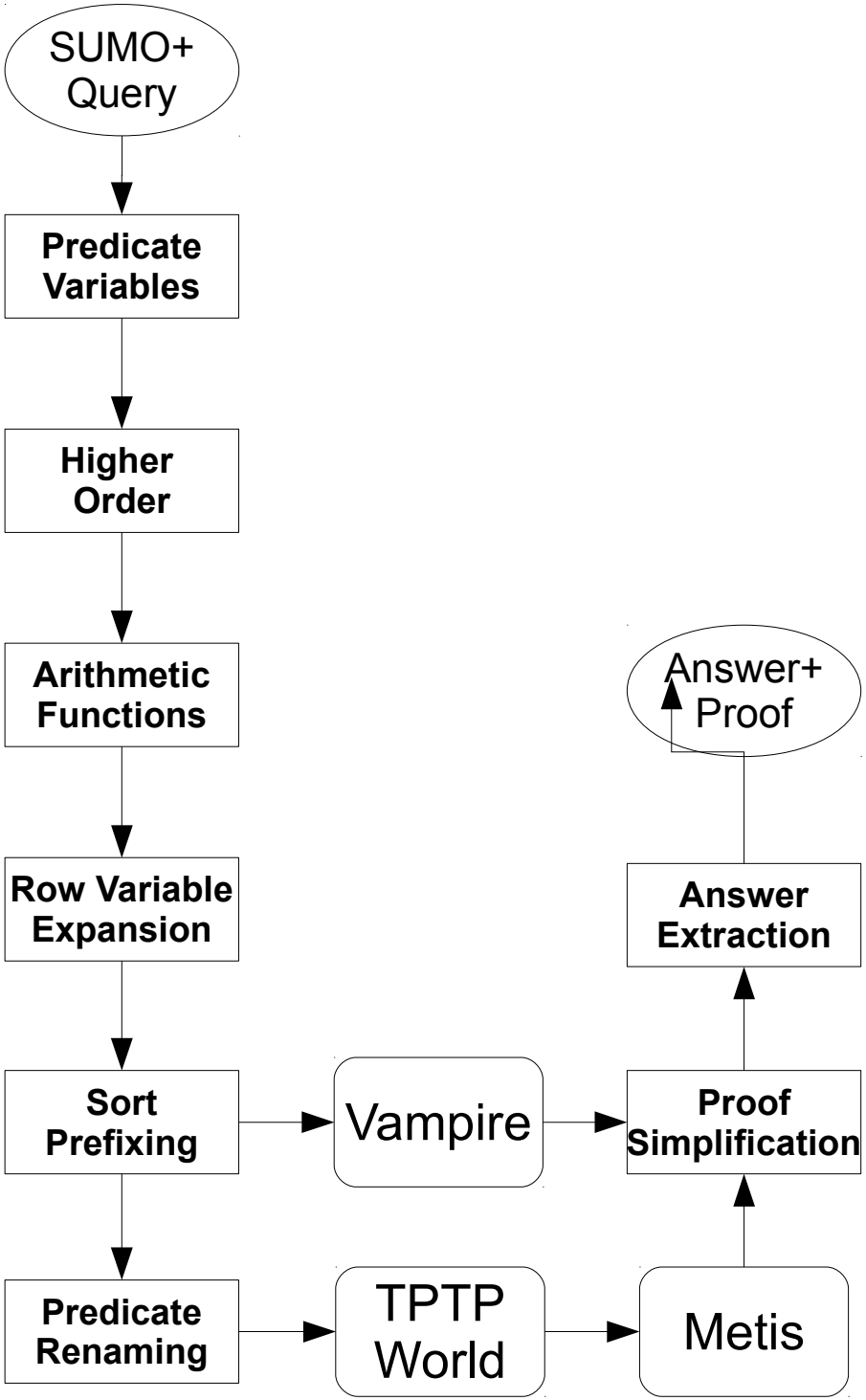
consequent

- Consistency check
  - Attempt to prove inconsistency
  - Incomplete
- Rootless term
- No documentation
- Term with no axioms
- Disjoint parents
- File dependency
- WordNet-SUMO hierarchy compare



- Local inference engines
  - KIF-Vampire, LEO-II, Metis, SInE
  - 40+ TPTP engines remote at U Miami





- Pre- and post-processing to interface with standard provers
- Metis needed for answer extraction and proof presentation with many provers

Sigma Knowledge Engineering Environment - Ask/Tell - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Sigma Knowledge Engineering ...

http://localhost:8080/sigma/AskTell.jsp

Sigma knowledge engineering environment
Inference Interface

[Home | Graph | Prefs] KB: SUMO Language:

```
(instance ?X Relation)
```

Maximum answers: 1 Query time limit: 30

Choose an inference engine:

- (•) Vampire
- ( ) SInE (+Vampire) (experimental)
- ( ) STP (experimental)
- ( ) STP2 (experimental)
- ( ) LEO-II with SInE (experimental)
- ( ) LEO-II local (experimental)
- ( ) LEO-II global (experimental)
- ( ) System on TPTP

Ask Tell

Answer 1. [definite] ?X = partition

1.	(instance partition VariableArityRelation)	[KB]	
2.	(not (instance ?VAR1 Relation))	[KB]	
3.	(subclass VariableArityRelation Relation)	[KB]	
4.	(forall (?VAR1 ?VAR2) (=> (subclass ?VAR1 ?VAR2) (and (instance ?VAR1 SetOrClass) (instance ?VAR2 SetOrClass))))	[KB]	
5.	(or (instance ?VAR1 SetOrClass) (not (subclass ?VAR1 ?VAR2)))	4	
6.	(or (instance ?VAR1 SetOrClass) (not (subclass ?VAR2 ?VAR1)))	4	
7.	(forall (?VAR1 ?VAR2 ?VAR3) (=> (and (instance ?VAR2 SetOrClass)	[KB]	

SUO-KIF

- variant of the KIF language (Genesereth, 1991)
- LISP-like syntax
- only logical operators in the language itself
  - Original KIF had ”definition” and class-forming operators

SUO-KIF (continued)

- “free” syntax
  - variables in the predicate position
  - quantification over formulas
  - predicates and instances may share names
- empty conjunctions etc not allowed
- Variables denoted by “?” character
- Sequence variables
- “forall”, “exists”, “=>” and “<=>”
- quantified variables have no explicit sort syntax

Class and Instance Creation Predicates

```
(instance Adam Human)
(subclass Human Mammal)

not

(Human Adam)
(Mammal Human)
```

Sigma Inference

- Since 2002 using a customized version of Vampire
  - Treat sequence variables as macros
  - Quantification of free variables
  - Quoting second order
  - “holds” prefixes (for functions too)
  - Adding explicit sorts (* new)

Sequence Variables

- Useful convenience for knowledge engineer

```
(=>
    (and
        (subrelation ?REL1 ?REL2)
        (?REL1 @ROW))
    (?REL2 @ROW))
```

becomes

```
(=>
    (and
        (subrelation ?REL1 ?REL2)
        (?REL1 ?ARG1))
    (?REL2 ?ARG1))
(=>
    (and
        (subrelation ?REL1 ?REL2)
        (?REL1 ?ARG1 ?ARG2))
    (?REL2 ?ARG1 ?ARG2))
```

etc.

Quantify Free Variables

- Universal quantification in assertion, existential in query

```
(=>
    (and
        (subrelation ?REL1 ?REL2)
        (?REL1 ?ARG1))
    (?REL2 ?ARG1))
```

becomes

```
(forall (?REL1 ?REL2 ?ARG1)
    (=>
        (and
            (subrelation ?REL1 ?REL2)
            (?REL1 ?ARG1))
        (?REL2 ?ARG1)))
```

“holds” prefixing

- Prepend a “dummy” predicate to every clause with a non-logical operator
- Forces any predicate variables into the first argument
- A single predicate name ruins performance
- Including number of arguments in name helps (and use apply_ for functions)

```
(=>
  (inverse ?REL1 ?REL2)
  (forall (?INST1 ?INST2)
     (<=>
        (?REL1 ?INST1 ?INST2)
        (?REL2 ?INST2 ?INST1))))
```

```
(=>
   (holds_3__ inverse ?REL1 ?REL2)
   (forall (?INST1 ?INST2)
      (<=>
         (holds_3__ ?REL1 ?INST1 ?INST2)
         (holds_3__ ?REL2 ?INST2 ?INST1))))
```

Quoting Second Order

- Unification still works

```
(believes Mary
  (likes Mary Bill))                    ;; fact

(believes Mary (likes ?X Bill))         ;; query

(likes Mary Bill)                       ;; result
```

- But logical operators lose their meaning

```
(believes Mary
  (and
    (likes Mary Bill)
    (likes Sue Bill)))

(believes Mary (likes ?X Bill))         ;; query doesn't unify
```

Sortals

```
(=>
    (and
        (instance ?TRANSFER Transfer)
        (agent ?TRANSFER ?AGENT)
        (patient ?TRANSFER ?PATIENT))
    (not
        (equal ?AGENT ?PATIENT)))
```

```
(domain agent 2 Agent)
(domain patient 2 Object)
```

- Use argument type signatures to define variable sorts

```
(=>
    (and
        (instance ?AGENT Agent)
        (instance ?PATIENT Object))
    (=>
        (and
            (instance ?TRANSFER Transfer)
            (agent ?TRANSFER ?AGENT)
            (patient ?TRANSFER ?PATIENT))
        (not
            (equal ?AGENT ?PATIENT)))
```

TPTP Syntax Translation

```
(forall (?REL ?OBJ ?PROCESS)
   (=>
      (and
         (holds_3__ instance ?REL CaseRole)
         (holds_3__ instance ?OBJ Object)
         (holds_3__ ?REL ?PROCESS ?OBJ))
      (exists (?TIME)
         (holds_3__ overlapsSpatially
             (apply_3__ WhereFn ?PROCESS ?TIME) ?OBJ))))
```

```
fof(name,axiom,
    ! [V_REL,V_OBJ,V_PROCESS] :
      ( ( holds_3__(s_instance,V_REL,s_CaseRole)
        & holds_3__(s_instance,V_OBJ,s_Object)
        & holds_3__(V_REL,V_PROCESS,V_OBJ) )
     => ? [V_TIME] :
          holds_3__(s_overlapsSpatially,
              apply_3__(s_WhereFn,V_PROCESS,V_TIME),V_OBJ) )).
```

Optimization – Predicate Variable Instantiation

- Instantiate predicate variables to eliminate “holds”

```
(=>
    (instance ?REL TransitiveRelation)
    (forall (?INST1 ?INST2 ?INST3)
        (=>
            (and
                (?REL ?INST1 ?INST2)
                (?REL ?INST2 ?INST3))
            (?REL ?INST1 ?INST3))))

 (=>
    (instance subclass TransitiveRelation)
    (forall (?INST1 ?INST2 ?INST3)
        (=>
            (and
                (subclass ?INST1 ?INST2)
                (subclass ?INST2 ?INST3))
            (subclass ?INST1 ?INST3))))
```

Optimization

- Cache transitive relations
- (subclass A B) (subclass B C)
  - Cache (subclass A C)