Adding Knowledge Representation & Reasoning to Machine Learning: Why and How

Benjamin Grosof*

1 hour; presented July 30, 2018 at Microsoft Research, Redmond, WA, USA,
hosted by Tom Ball

These slides to be available via link at:
http://benjamingrosof.com/misc-publications/

* Chief Scientist, Kyndi Inc.; based also largely on work done while previously at Coherent Knowledge
https://kyndi.com
http://coherentknowledge.com
http://benjamingrosof.com

Bio – Benjamin Grosof

- AI researcher, executive, and entrepreneur
- Chief Scientist at Kyndi – AI startup on NL question answering using ML+KRR
- Co-founder & Board of Coherent Knowledge – AI startup on KRR
- *Previously:*

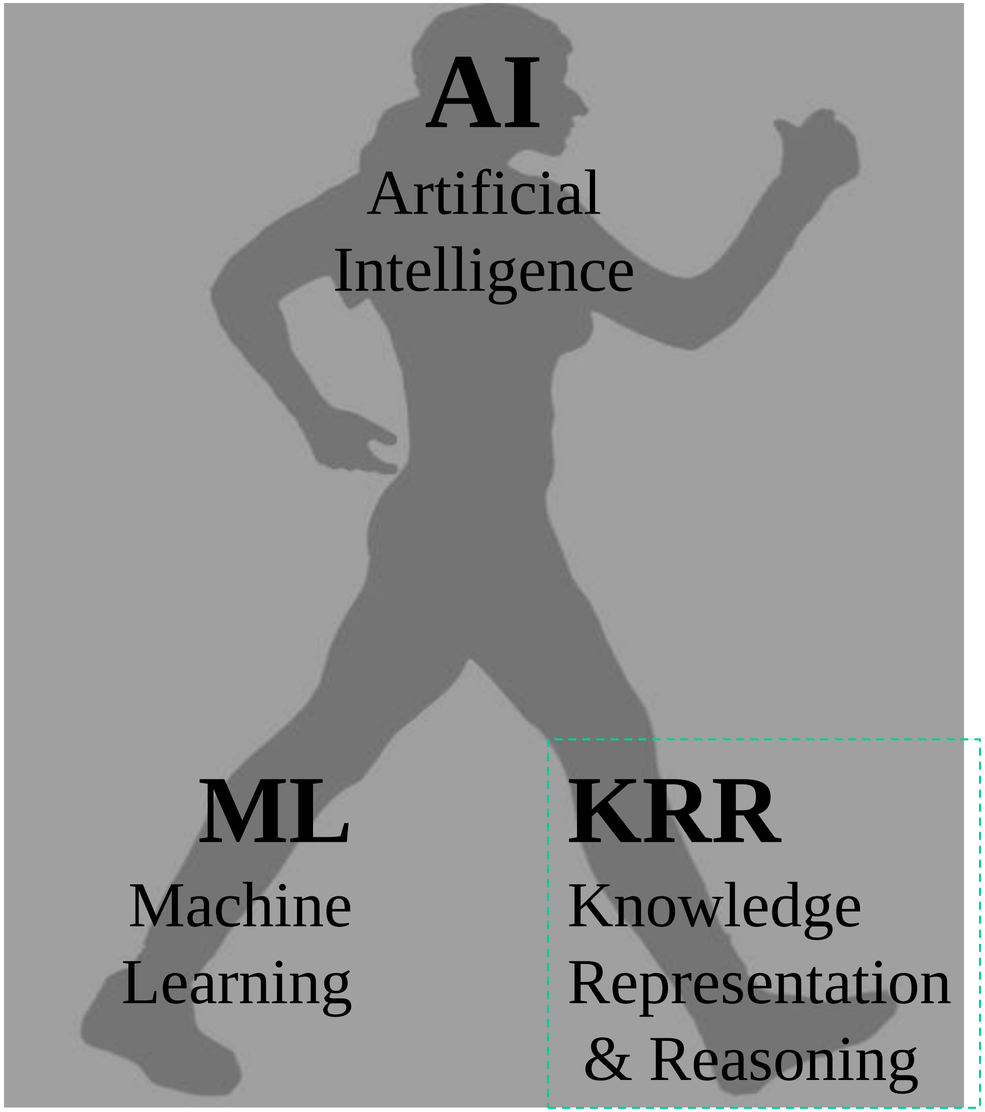
  - Principal Director & Research Fellow in AI at Accenture on BPM
  - CTO & CEO of Coherent Knowledge
  - Directed advanced AI research at predecessor of Allen Inst. for AI
    - Developed Rulelog KRR theory, algorithms, UI approach
  - MIT Sloan professor and DARPA PI
    - Co-Founder of RuleML, key contributor to W3C OWL-RL and RIF standards
  - IBM Research, creator IBM Common Rules
    - 1st successful semantic rules product in industry
  - Stanford AI PhD, combining ML with logical and probabilistic reasoning
- *Themes: flexible clean KRR + NL + ML; many app domains & tasks*

- http://www.linkedin.com/in/benjamingrosof
- http://benjamingrosof.com

Outline

- Intro
  - Core of AI. What are KRR and ML.
- Why to combine KRR + ML
  - 10 reasons to add KRR to ML
- How to add KRR to ML – deeper dive
  - Which kinds suitable for adding to ML
  - Background: Logic Programs (LP)
  - KRR requirements for ML
  - Rulelog KRR – extension of LP
  - Probabilistic LP – extension of LP
- Directions for future research

The Core of AI



It takes two legs to walk

Machine Learning

From data and previous knowledge,

learn additional generalizations as new reusable knowledge

= hypotheses that enable prediction (errorfully)

Logic's role in Computer Science

"Logic is the calculus of computer science" [Z. Manna & R. Waldinger]

- Computer science originally invented largely by logicians
- Foundations of hardware; gates, circuits
- Foundations of programming languages; verification
- Databases
- Conceptual modeling
- KRR in AI; business rules; ontologies, semantic web

Concept of logical Knowledge Representation (KR)

A given KR logical system S has …

1. Formal language L_S for <u>assertions</u> and <u>conclusions</u>
   - Assertions LA_S and conclusions LC_S may be different!!
     - E.g., in LP and Rulelog
2. Semantics: <u>entailment</u> relation $|=_S$
   - An assertion set *A* entails a* conclusion set *C*
     - * We assume here exactly 1
   - Typically, entailment is defined formally in terms of <u>models</u>
     - Truth assignments on LC_S that meet criteria based on the assertions *A*
       - E.g., in FOL and LP and Rulelog

Reasoning implements the semantics, e.g., to answer queries

- KR<u>R</u> <u>software</u> system: knowledge representation & <u>r</u>easoning (proving)
- Typically KRR systems are sound but often incomplete

Semantic

- "Semantic" tech/rules/web means: based on logic
- Advantages for communication across systems and organizational boundaries
- Meaning is shared notion of what is/is-not inferrable
- Abstracts away from implementation

- Relational DB was 1st successful semantic tech

Directions of Reasoning

- *Forward* direction: start from assertions
  - Draw some conclusions,
    then recursively work to draw more conclusions
- *Backward* direction: answer *queries*
  - Start from goal, recursively work backwards
    via sub-goals to assertions
- Both: *chain* through a series of
  intermediate conclusion steps
  - “Backward chaining”, “forward chaining”
- Underlying: *search* for useful valid chains
  - Pure backward is usually more efficient than pure forwards,
    because it is more focused
- *Mixed* direction – hybrid – is often superior to
  both pure backward and pure forward

Focal Kinds of KRR, in this talk

- **Rulelog** – highly expressive extension of logic programs
  - Blends with higher-order classical logic
  - Flexible yet scalable
  - Family of KRs that are subsets (fragments)
- **Probabilistic logic networks**
  - A.k.a. graphical probabilistic models
  - Family of KRs
  - Close relationship to logic programs

- Background: **LP** = (well-founded declarative) **logic programs** is the core KR of the entire ***IT*** world, not just of AI
  - It's a logic, despite the "programs" in its name
  - Invented by computer scientists not mathematicians
  - Developed to formalize relational databases and unify that with the pure subset of Prolog

Logic Programs Example

- “:-” means “if”, i.e., \impliedby
- *Assertions:*
  - human(Socrates). human(Anne). human(Peter).
  - modern(Anne). educated(Socrates).
  - mortal(?x) :- human(?x).
  - fallible(?z) :- mortal(?z).
  - educated(?x) :- human(?x) \and modern(?x).
  - humble(?w) :- fallible(?w) \and educated(?w).
- *Forward chaining:* mortal(Socrates). mortal(Anne). mortal(Peter). fallible(Socrates). fallible(Anne). fallible(Peter). educated(Anne). humble(Socrates). humble(Anne).
- *Query:* ?- humble(?p). **Answer:** {Socrates, Anne}
  - Backward chaining subgoals:
    ?- fallible(?p), ?- educated(?p).
  - ?- mortal(?p). ?p / S
  - ?- human(?p). ?- human(?p), ?- modern(?p).
  - ?p / S, A, P ?p / S, A, P ?p / A

The Kind of Logic You Learned in School

- Classical logic, a.k.a. mathematical logic
  - Goal: support mathematics
- \forall(?x)^(human(?x) \implies exists(?y)^(mother(?x,?y)).
- \exists(?z)^(owns(Doug,?z) \and (dog(?z) \or cat(?x))).
- \forall(?w)^(number(?w) \implies number(successor(?w)).
- \forall(?p)^(disjoint(?c1,?c2) \implies \neg(?c1(?x) \and ?c2(?x))).
- \forall(?n1,?n2,?n3,?b)^(?r(?n1,?n2) \and ?r(?n2,?n3) \implies transitiveClosure(?r)(?n1,?n3)).
- Connectives: \and, \or, \neg, \implies, \equivalent
- Quantifiers: \forall, \exists
- Variables: ?x, ?y, …
- Predicates (e.g., mother): map tuple of arguments to a truth value
- Logical functions (e.g., successor): map tuple of arguments to a term
- Higher-order: a predicate or function can be a variable or term
- First-order: a predicate or function must be a constant symbol
  - FOL = First-Order Logic

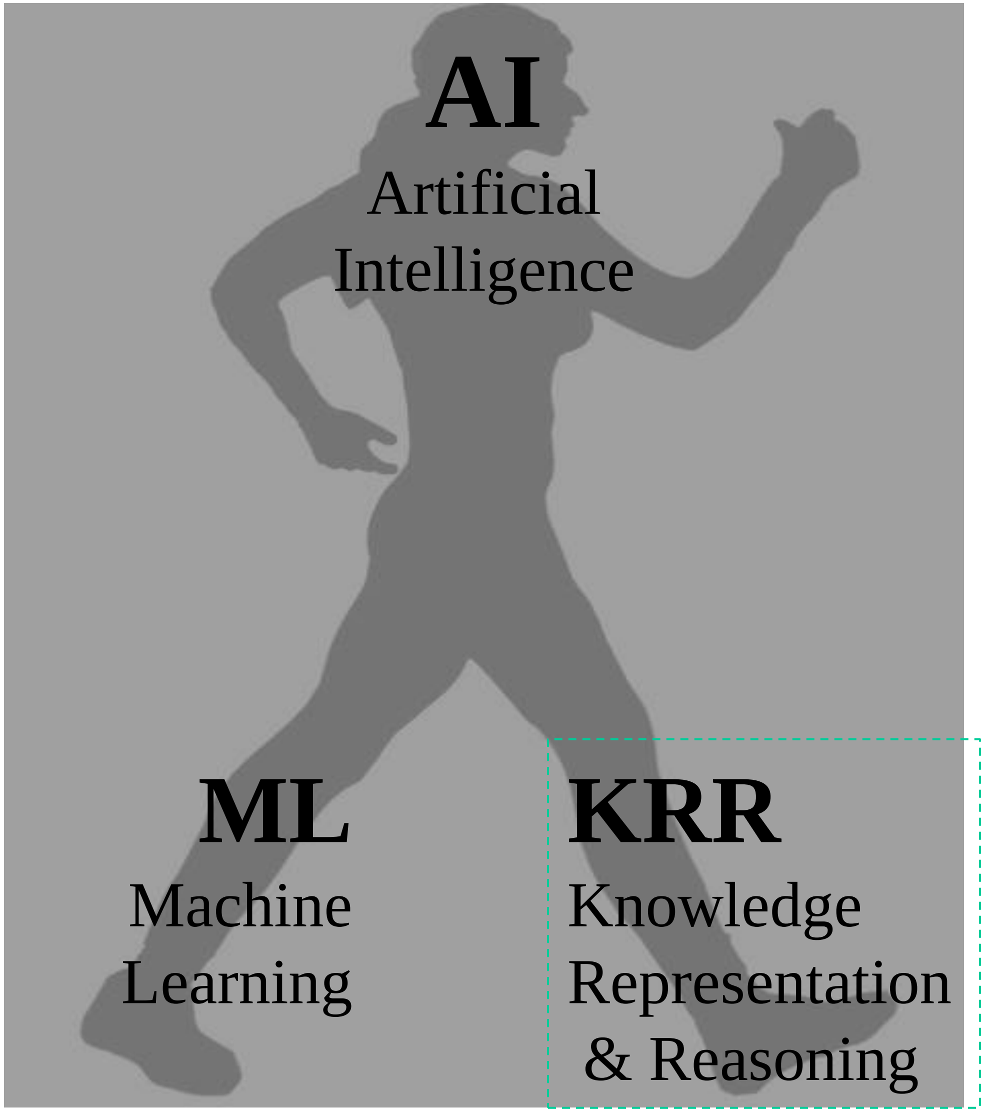
Practical Logic, vs. Classical Logic

- Goal: support IT, vs. mathematics
  - E.g., Databases, Rules
  - Central: declarative logic programs (LP) KR
    - “Well-founded” semantics

- Requirements:
  - Scalable computationally
  - Robust in face of human errors and miscommunications
  - →→ “Humble”
    - Avoid general reasoning by cases
    - Avoid general proof by contradiction

What is “reasoning by cases”: (background)
Assertions: if A then C. if B then C. A <u>or</u> B.
Conclude: C.

The Core of AI



Commercial focus

Staggering / Hopping:

- 1980-1995: KRR-alone is dominant
  - expert systems, rules
  - (~1990-1995: AI winter)
- 1995-2005: ML-alone is dominant
  - data mining
  - (business rules sector also thrives)

Walking:

- 2005-2018: ML starts adding KRR
  - probabilistic nets
    - Bayesian
      + simple logic eg frames
  - IBM Watson Jeopardy
  - deep neural nets
    - learn simple representations
      - eg word embeddings
  - NL parsers and entity recognition
    - ML + grammars
      + terminology hierarchies
  - chatbots: ML-based NLP
    + simple ontological hierarchies

Outline

- Intro
  - Core of AI. What are KRR and ML.
- Why to combine KRR + ML
  - 10 reasons to add KRR to ML
- How to add KRR to ML – deeper dive
  - Which kinds suitable for adding to ML
  - Background: Logic Programs (LP)
  - KRR requirements for ML
  - Rulelog KRR – extension of LP
  - Probabilistic LP – extension of LP
- Directions for future research

KRR's Roles in AI

- **Complements ML … in sense of induction from data … to enable ML in broader sense**
- **The power of cultural transmission**
  - **"Evolution's lesson" (Wolfgang Bibel)**
- **Accumulate knowledge coherently**
- **Communicate with humans: expertise, questions**
- **"Inject" ML results into predictable software**

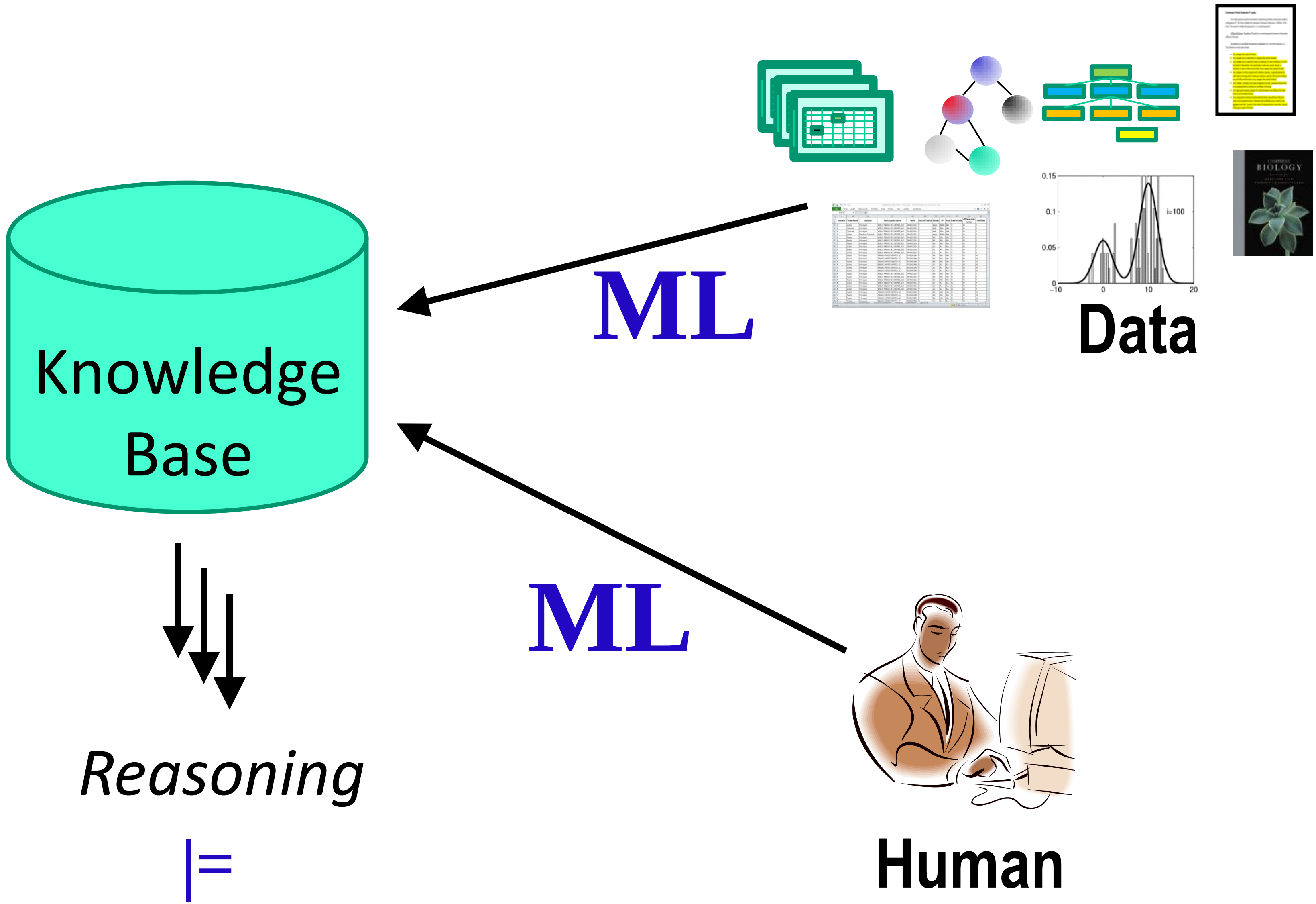
Why Add ML to KRR

2 ways it's useful or even required,
from the viewpoint of KRR, i.e., "for KRR's sake":

1. KB construction: ML is useful to *supply* knowledge

2. *Improve the process of knowledge acquisition*
   - (Can view this as supplying a kind of meta-knowledge)
   - From manual entry of knowledge, e.g., encoding NL into rules
   - From knowledge interchange

Why Add ML to KRR – Summary Diagram



Why Add KRR to ML (I)

10 ways it's useful or even required,
from the viewpoint of ML, i.e., "for ML's sake":

1. The *prediction* step of ML requires reasoning
   - This could be pulled by an ML system via backchaining
   - Why not hook up various external programs such as reasoners, to ML components, e.g., neural networks, to evaluate some nodes/functions?

2. The *target* of ML is a representation

3. Getting business *value* from ML requires reasoning for analysis and decisions

Why Add KRR to ML (II)

4. KRR is required to *combine* results of ML from
    a. Multiple episodes
    b. Multiple sources
    c. Multiple methods

5. KRR is required to *accumulate* knowledge coherently
    - Weakness of ML today
    - Think cultural transmission

Why Add KRR to ML (III)

6. KRR is required to *explain* knowledge understandably to humans
   - Weakness of ML today
   - Needed for humans to trust an automated system
   - Often part of required/desired analysis functionality for own sake

Why Add KRR to ML (IV)

7. Reasoning to *supply derived facts* for ML to chew on as training examples or background info
   - This could be pulled by an ML system via backchaining

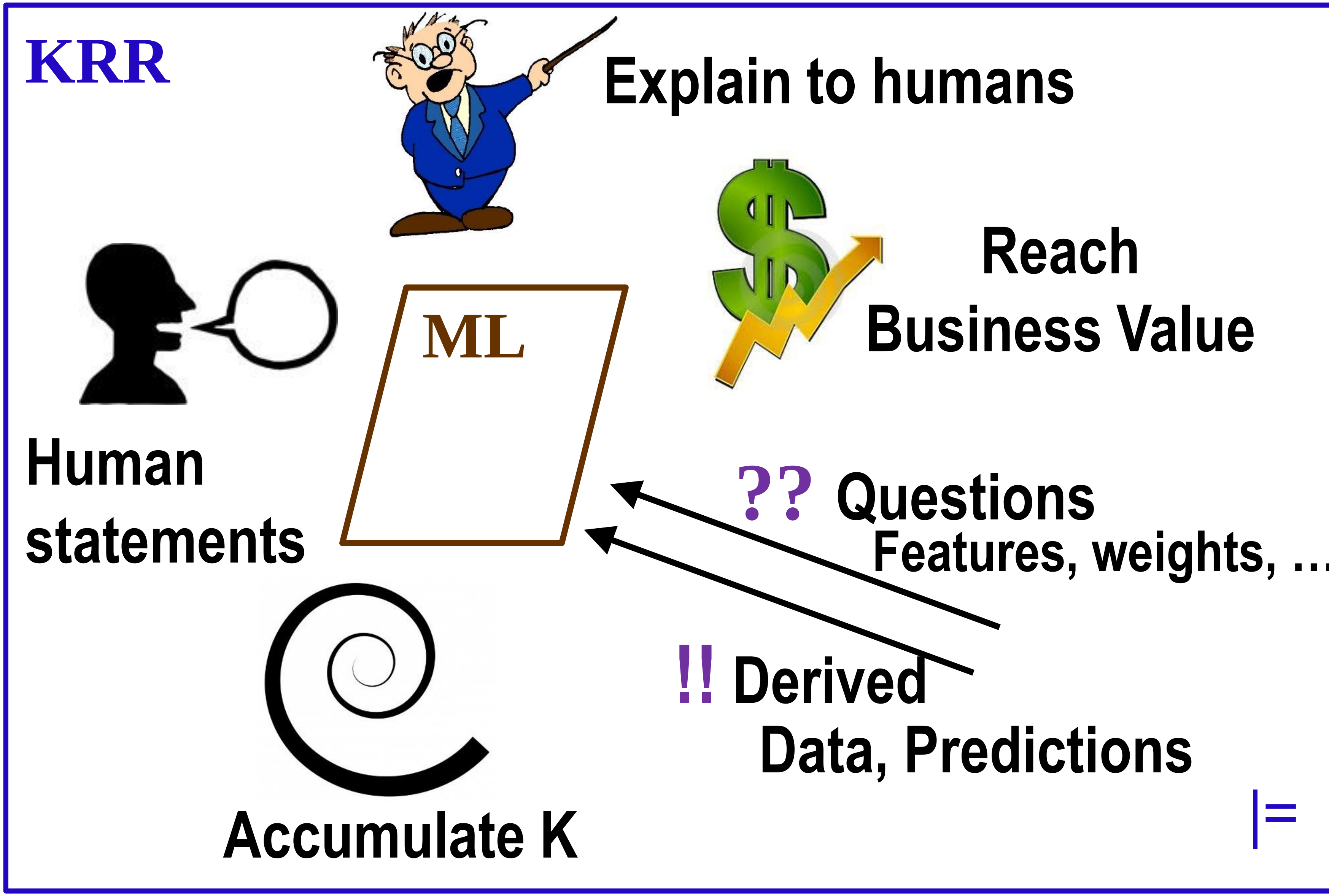
8. *Humans know stuff* beyond what's available via ML training data, and such knowledge is often complex to state / enter
   - KRR methods for entry are often more cost-effective than programming

Why Add KRR to ML (V)

9. Reasoning is desirable to *pose questions* (tasks) to ML
    - as reasoning (sub)goals from KRR

10. Reasoning is desirable to *provide sets of relevant features, (hyper-)parameters, and/or weights* to ML

Why Add KRR to ML – Summary Diagram



Outline

- Intro
    - Core of AI. What are KRR and ML.
- Why to combine KRR + ML
    - 10 reasons to add KRR to ML
- How to add KRR to ML – deeper dive
    - Which kinds suitable for adding to ML
    - Background: Logic Programs (LP)
    - KRR requirements for ML
    - Rulelog KRR – extension of LP
    - Probabilistic LP – extension of LP
- Directions for future research

KRR <u>Expressive</u> Flexibility Requirements for combination with ML (I)

- Higher-order syntax
  - Critical for natural language, modalities, ontology mappings
- Strong meta (statements about statements)
  - E.g., Inductive LP has evolved into Meta Interpretive Learning
- Numeric uncertainty (including weighting)
  - Including Bayesian-probabilistic and fuzzy
  - Critical for differentiability and neural-network
- Defeasibility (exceptions, argumentation, defaults)
  - Treat the evolving character of knowledge and of the world
  - Critical for natural language, science, legal/policy
- Quantified formulas
  - Critical for natural language

KRR Scalability Requirements for combination with ML (II)

- Scalable computationally
  - To large amounts of asserted and concluded knowledge
  - I.e., to "*volume*" and "*velocity*"

- Scalable "socially" to multiplicity of diverse knowledge
  - To multiplicity of diverse ML/etc. sources, e.g., organizations
  - To multiplicity of diverse ML/etc. algorithmic methods
  - To multiplicity of diverse underlying ML data samples
  - I.e., to "*variety*" including heterogeneity

LP is the Central Form of Practical Logic

- ***LP is the core KR of structured knowledge management today***
- A non-classical logic invented by computer scientists

- Subsets of LP – important in industry landscape today
  - Relational databases (SQL) *[Datalog subset of LP]*
  - Graph databases, a.k.a. knowledge graphs (SPARQL) *[Datalog]*
    - Also: XML databases, object-oriented databases, other semi-structured databases
  - Production rules, Event-Condition-Action rules. More precisely: their logical subsets.
  - Prolog. More precisely: its "pure" logical subset.
  - *Also industry standards for semantic rules and ontologies:*
    - Many RuleML & RIF dialects, e.g., RIF-BLD, RIF-Core, SWRL
    - Many ontology standards, e.g., OWL-RL, RDF-Schema *[Datalog]*

Logic Programs: technical overview (I)

- Knowledge base (KB) is a set of rules, each of form:
  - *Head_formula* :- *Body_formula*.
  - Intuitively: OK to infer (establish) the head if can infer the body
- Basic normal LP: each rule has form:
  - atom :- literal_1 \and … \and literal_m.
  - Plus: atoms are all first-order
- *atom* has form: (predicate(arg_1,…,arg_k)), where each arg_i is a term
- *literal* has form: (atom) or (\naf atom)
- Weak negation: \naf p – p is not believed (essentially, known to be not provable)
- Strong negation: \neg p – p is believed to be strongly false (i.e., opposite of true)
  - Not permitted in normal LP. But permitted in extensions of normal LP, e.g., in Rulelog.
- Aggregation: setof{?x | *condition*}, where ?x appears in *condition* formula
  - Enabled by \naf. Aggregate operators also include avg, max, min, count.
  - average_salary(?co,?amt) :-
    company(?co) \and avg(?amt | employee(?co,?e) \and salary(?e,?amt)}.

Logic Programs: technical overview (II)

- Horn subset: body literals are restricted to be atoms
- Datalog subset: Horn, and function-free
- Full normal LP permits also:
  - in head: \and
  - in body, freely nested: \or, \forall, \exists, aggregates, \and, \naf
  - Integrity constraints via *violation(…)* as a head atom predicate
  - Reduces via transformation to basic normal LP
- Semantics (well-founded) is based on:
  - An alternating least fixed point construction in 3-valued logic
  - Each instantiated atom is assigned to 1 of 3 truth values {t,f,u}:
    - t = *true*; f = "*false*" (cf. \naf); u = "*undefined*" (don't-care).
    - *undefined* is useful for paradox and restraint bounded rationality
- Function-free case is polynomial time
- Functions (thru recursion) lead to undecidability

The "Spirit" of LP

The following summarizes the "spirit" of how LP differs from FOL:

- "Avoid Disjunction"
  - <u>Avoid disjunctions of positive literals</u> as expressions
    - In premises, intermediate conclusions, final conclusions
    - (conclude (A or B)) only if ((conclude A) or (conclude B))
  - Permitting such disjunctions creates exponential blowup
    - In propositional FOL: 3-SAT is NP-hard
    - In the leading proposed approaches that expressively add disjunction to LP with negation, e.g., propositional Answer Set Programs
  - No "reasoning by cases", therefore
- "Stay Grounded"
  - Avoid (irreducibly) non-ground conclusions

LP, unlike FOL, is straightforwardly extensible, therefore, to:

- Nonmonotonicity – defaults, incl. NAF
- Procedural attachments, esp. external actions

Important Extensions of LP (I)

- Rulelog – reduces by efficient transformation to normal LP
  - Higher-order, reification, rule identifiers
    - Higher-order relies on (logical) functions
  - Defeasibility: prioritized defaults, exceptions, argumentation
    - \neg in literals (not outside of \naf). Flexible behavior, efficient approach.
  - Restraint bounded rationality: guarantee polynomial time
    - Specify *undefined*-ness in various circumstances
  - Head quantifiers; \exists treated via skolemization
  - Head \or, treated as "omnidirectional" (weak)
  - Object-oriented ("frame") syntax
  - External queries and import of most kinds of enterprise info
  - Probabilistic via: higher-order, defeasibility
    - But current implementations not optimized
    - Flexible: can have tuple of parameters for the probability
      - pr(*formula1*)[low->0.91,hi->0.94]. pr(*formula2*)[mu->0.925,sigma->0.008].

Rulelog and Textual Rulelog

- Rulelog reasoning scales well: polynomial-time, as in databases
  - Millions of sentences concluded/asserted on a single processor
  - Up to trillions by orchestrating database etc. systems in distributed settings
- *Textual* Rulelog extends Rulelog with natural language processing (NLP)
  - Logic itself is utilized to map between English syntax and logic syntax
  - Textual templates aid knowledge entry and explanation generation
- Examples:
  - \(The individual affiliate threshold for transaction under Regulation W by ?Bank with ?Counterparty is ?Amount\) :-

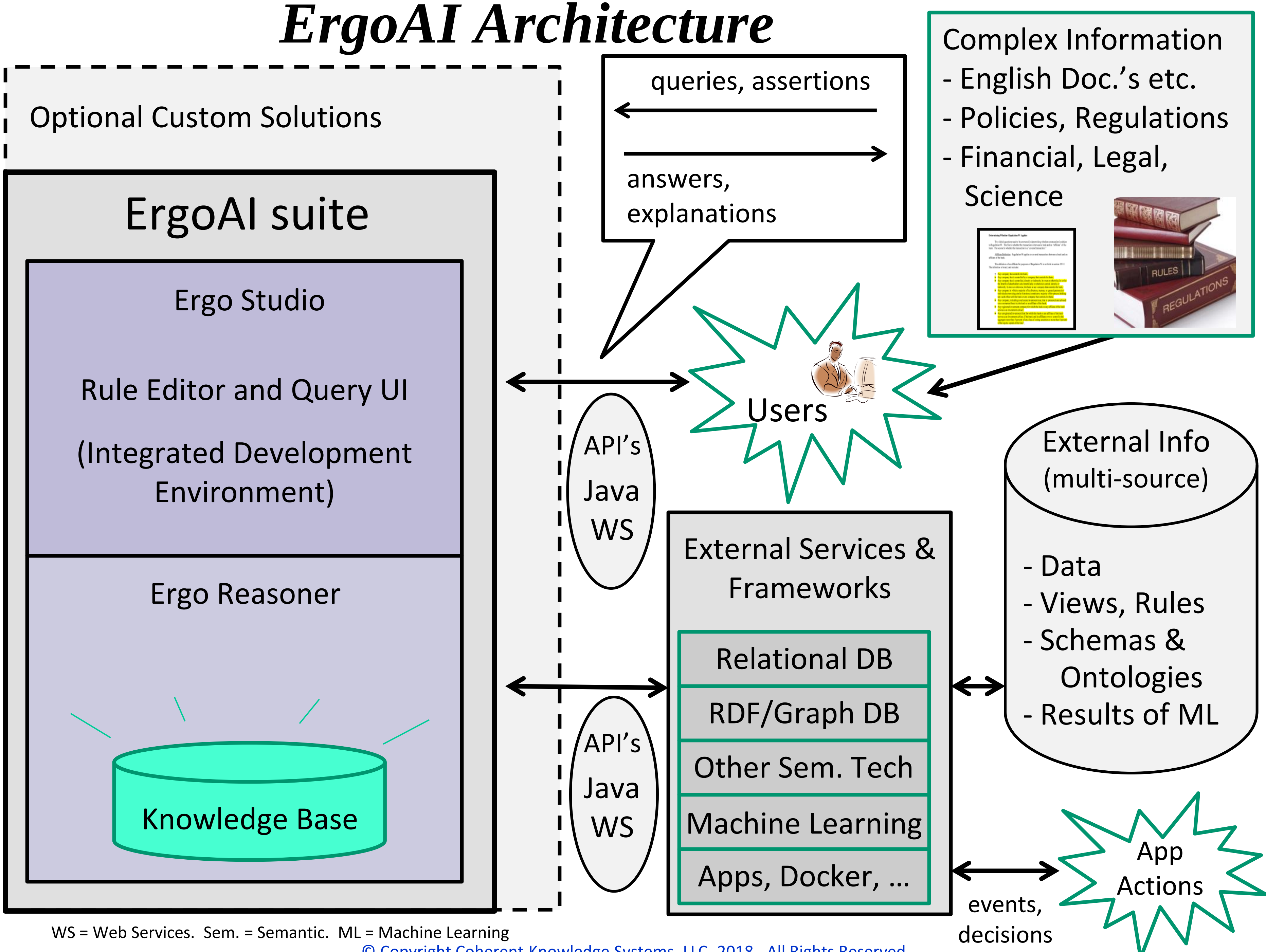
        \(?Counterparty is deemed an affiliate of ?Bank under Regulation W\) \and
        \(?Bank has capital stock and surplus ?Capital\) \and
        \(the threshold percentage for an individual affiliate is ?Percentage\) \and
        ?Amount = ?Capital * ?Percentage/100.

  - @{‘each large company has some talented CEO’}
    - forall(?x)^((?x \isa \(large company\)) ==>
      exists(?y)^((\?x has ?y\) \and
      (?y \isa \(talented CEO\)))).

ErgoAI: Reasoner, Studio, Connectors

- Ergo Reasoner has sophisticated algorithms & data structures
  - Smart cacheing with dependency-aware updating. Leverages LP & DBMS techniques.
  - Transformation, compilation, reordering, indexing, modularization, dependency/loop analysis, performance monitoring/analysis, pausing, virtual machine, programming kernel, external import/querying
  - Java API. Other interfaces: command line, web, C. Additional APIs for Python, REST, more.
  - <u>Scales</u> well: Millions of sentences on 1 processor; Trillions on distributed nodes
- Explanations: for every answer; interactively drill down tree; in NL
  - Rulelog enables natural deduction style proofs, automated NL generation via rules
- Ergo Studio is a graphical Integrated Development Environment
  - Interactive editing, querying, visualization of knowledge
  - Fast edit-test loop with award-winning advanced knowledge debugging/monitoring
- Ergo Connectors federate knowledge & reasoning
  - Import/query dynamically via: SPARQL, OWL, RDF; SQL; CSV; JSON; more
  - Federation distributes reasoning (i.e., its processing) across multiple nodes
- Open, standards-based approach; a portion is open source
  - Rulelog is draft industry standard from RuleML (submission to W3C & Oasis)

ErgoAI Architecture



WS = Web Services. Sem. = Semantic. ML = Machine Learning

Why is the proposed transaction prohibited by Regulation W?

3. Why is the aggregate-affiliates limit $10 million?

Why 'What proposed transactions are prohibited by RegW? Show '('Pacific Bank','Maui Sunset',23.0) ?

Edit Operations

- RegW prohibits the proposed transaction by Pacific Bank with Maui Sunset of $23.0 million
  - The proposed transaction by Pacific Bank with Maui Sunset of $23.0 million is a RegW covered transaction
  - There is a limit of $10.0 million for any proposed RegW covered transaction by Pacific Bank with Maui Sunset
    - There is an aggregated-affiliates limit of $10.0 million for any proposed RegW covered transaction by Pacific Bank with any affiliate
      - The aggregated total of previous RegW covered transactions by Pacific Bank with all affiliates is $490.0 million
      - The maximum threshold for aggregate RegW covered transactions by Pacific Bank with all affiliates is $500.0 million
        - The capital stock and surplus of Pacific Bank is $2500.0 million
        - The RegW threshold percentage for aggregate affiliates is 20.0 percent
        - $500.0 million is $2500.0 million multiplied by 20.0 percent
      - The limit of $10.0 million is the result of subtracting the previous RegW covered transactions total of $490.0 million from the RegW threshold $500.0 million
    - There is an individual-affiliate limit of $250.0 million for any proposed RegW covered transaction by Pacific Bank with Maui Sunset
    - The overall RegW limit of $10.0 million is the lesser of $10.0 million and $250.0 million
  - The proposed transaction of $23.0 million is greater than the RegW limit of $10.0 million

Coherent Knowledge

Important Extensions of LP (II)

- Probabilistic LP, in two major flavors
  1. Bayesian flavor with "distribution semantics" (a.k.a. other names)
     - Bayesian Networks are a subset
     - General case is computationally intractable, even for function-free
  2. Fuzzy flavor. Parametrized by choice of "triangular norm" function F.
     - pr(A \and B) = F(pr(A),pr(B)). E.g., F = *min*. Co-norm for \or: e.g., *max* .
     - Function-free case is polynomial time
  - Numerical truth values for atoms (and rules) range on real interval [0..1]
  - *head* formula can be: \or of disjoint atoms/literals whose weights add to 1
     - friendly(?x):0.8 \or unfriendly(?x):0.2 :- student(?x).

- Answer Set Programs – but not so close to ML
  - Head permits \or. Classical-like reasoning-by-cases.
  - Even function-free case is computationally intractable

Industry Landscape of Practical KRR

- LP and subsets *(cf. earlier slide)*
- Subsets of Classical Logic:
  - Propositional. E.g., hardware circuit design, satisfiability for planning.
  - First Order Logic (Common Logic). E.g., for program verification.
  - Description Logic (OWL) subset of FOL. For ontologies.
- Emerging: (in roughly descending order of commercial maturity)
  - Rulelog – extension of LP
    - RIF/RuleML Rulelog dialect standard is in draft
  - Bayesian Probabilistic LP
    - Bayesian Networks are a special case
  - Fuzzy Probabilistic LP
    - Probabilistic Soft Logic – is closely related
  - Other probabilistic graphical models (PGM)
    - Markov Logic Networks – closer to classical; thus more difficult to scale
  - Answer Set Programs – closer to classical, less humble
  - (Others are not so commercially/practically prominent)

Some State-of-the-Art KRR Systems

- LP: XSB (Stonybrook U., Theresa Swift, David Warren, et al)
  - Full programming language that is Prolog+
- Rulelog: ErgoAI (Coherent Knowledge, free for research), and its open-source subset Flora-2 (originally Stonybrook U.)
  - Full programming language that is Prolog++ and XSB++
- Bayesian Probabilistic LP: Problog (Luc de Raedt et al, originally KU Leuven)
- Probabilistic Soft Logic: (Lise Getoor et al, UC Santa Cruz)
- Markov Logic Networks: Alchemy (Pedro Domingos et al, U. Washington)
- Probabilistic Graphical Models generally: See STARAI workshops

Outline

- Intro
  - Core of AI. What are KRR and ML.
- Why to combine KRR + ML
  - 10 reasons to add KRR to ML
- How to add KRR to ML – deeper dive
  - Which kinds suitable for adding to ML
  - Background: Logic Programs (LP)
  - KRR requirements for ML
  - Rulelog KRR – extension of LP
  - Probabilistic LP – extension of LP
- Directions for future research

Future Research Directions

- Add KRR systems to ML, cf. the 10 reasons
- Promising candidates: Rulelog (ErgoAI), Probabilistic LP (Problog), Fuzzy LP and Probabilistic Soft Logic
- Explainability is very beneficial for trustability. E.g., GDPR.
  - With Neural Networks / Deep Learning – address opacity
- Guardrails are very beneficial for trustability
  - Policy/legal compliance. E.g., in Microsoft Tay, regulations, contracts, fraud, fairness/bias
- Differentiability: how to be semantically principled
- Nested graphs with vectors, cf. Kyndi and Vincent Zheng
  - Add to ML: logically structured interpretations of NL phrases, weighted logical querying
- Other Apps: NLU, chatbots, search, question answering

Future Research Directions: on Rulelog KRR itself

- Optimize, and study further: Rulelog reasoning with uncertainty that is numerically weighted, including probabilistic and fuzzy

- Extend Rulelog's expressiveness to selective reasoning-by-bases

Coherent Knowledge

Outline

- Intro
  - Core of AI. What are KRR and ML.
- Why to combine KRR + ML
  - 10 reasons to add KRR to ML
- How to add KRR to ML – deeper dive
  - Which kinds suitable for adding to ML
  - Background: Logic Programs (LP)
  - KRR requirements for ML
  - Rulelog KRR – extension of LP
  - Probabilistic LP – extension of LP
- Directions for future research

Thank You