Hilog, Defeasibility, and the Foundations of Practical Meta-Knowledge: A Brief Introduction

Benjamin Grosof*

October 31, 2013

Ontolog Forum‡

Globally accessible webconference session

* Benjamin Grosof & Associates, http://www.mit.edu/~bgrosof/

and

Coherent Knowledge Systems http://www.coherentknowledge.com

‡ http://ontolog.cim3.net/cgi-bin/wiki.pl?ConferenceCall_2013_10_31

Meta in Rulelog – *Extension of LP*

Rulelog has several expressive features for **meta** knowledge

- Overall: mix meta knowledge with "base" knowledge, in fine grain
  - Just as the web/markup mixes meta in *data* with "base" data, in fine grain
- **Hilog**: any atom can be treated as a term. Used also in Common Logic.
  - Provides higher-order syntax (bit restricted)
  - Semantics reduces (transforms) to first-order, and uses logical functions.
- Reification: any formula can be treated as a term. A.k.a. *quoting*.
  - Provides modal syntax
- Rule id's: enables meta-statements about assertions (i.e., about rules)
  - Every assertion has a rule id, that is a constant in the logical language
  - Useful for provenance, defeasibility, restraint, and other purposes
- **Defeasibility**: any rule can have exceptions (non-monotonically)
  - Strong negation (neg). Prioritized conflict handling. Cancellation of rules.
  - Argumentation-theory approach: specify via rules the principles of defeat
- Restraint: bounded rationality, using the "undefined" (u) truth value
  - u represents "not bothering"
  - Specify via rules the principles of such "not bothering"
  - Radial restraint: treat as u every atom/literal whose size exceeds a fixed radius

Examples of Reification

- Reification (a.k.a. quoting) makes a term out of a formula:

  believes(john, **${** likes(mary,bob) **}**)

  Term made out of the formula likes(mary,bob)

- Variables can be back-quoted:

  jealousOf(john,?X) :- believes(john, ${likes(mary, `?X)}.

  Back-quoting of ?X makes its scope be outside the quoted formula that ?X appears within

- See, e.g., [Yang & Kifer, ODBASE 2002]

- Rules, not just formulas, can be reified as well

Examples of Hilog

Hilog permits predicates and functions to be any term: a variable or a complex term, not just a constant

p(?X,?Y) :– ?X(a,?Z) and ?Y(f(?Z)(b)).

***Variable as predicate**: ranges over predicate names of arity 2*

***Variable as function:** ranges over function names of arity 1*

***Complex-term as function:** ranges over function names of arity 1*

Hilog also permits variables over atomic formulas. This is a kind of reification:

p(q(a)).
r(?X) :– p(?X) and ?X.

***Meta-variable**: ranges over unary method names*

Introduced in [Chen, Kifer, Warren, "HiLog: A Foundation for Higher-Order Logic Programming", J. of Logic Programming, 1993]

Rule ID's

- Simple, but important, feature
- Each (assertion) statement gets a unique rule id
- The id can be explicitly specified
  - @!{myRule17} H :- B.
- Or if implicit, is a skolem essentially
  - H :- B. → gets treated as: @!{gensym0897} H :- B.

- Enables various useful kinds of meta-knowledge, by asserting properties of the rule id
  - Provenance, e.g., createdBy(myRule17, Benjamin)
  - Defeasibility
  - Rule-based transformations, e.g., for language extensibility, UI, NLP

Uses of Hilog and Reification and Rule ID's

Overall: for knowledge exchange and introspection

- Ontology mappings
- KB translation/import
- KR macros
- Modals (incl. deontic, alethic)
- Multi-agent belief
- Provenance and other aspects of context
- Reasoning control, incl. restraint bounded rationality
- KB modularization
- Navigation in KA (knowledge acquisition)
- …

- Argumentation-theory approach to defeasibility
  - Principles of defeat (i.e., of debate) are meta rules that use Hilog and rule id's

HiLog Transformation

- HiLog semantics is defined via a transformation
- A simplified version of that, which gives intuition:
  - Rewrite each atom p(a,b) → holds_2(p,a,b)
    - Generic predicate constants holds_1, holds_2, …
  - Treat each term in similar manner
    - f(a,b) → apply_2(f,a,b)
    - Generic function constants apply_1, apply_2, …
- General case of transformation heavily uses logical functions
  - ⇒⇒ creates a challenge in implementation

Knowledge often has Exceptions

- **A.k.a. knowledge is *defeasible* (i.e., can be "defeated")**
- **"A (eukaryotic) cell has a nucleus." … Except when it doesn't ☺**
  - A cell has no nucleus during anaphase. Red blood cells have no nuclei.
  - A cell has two nuclei between mitosis and cytokinesis. Some fungi are multinucleate.
- **Exceptions / special cases are inevitably realized over time**
  - E.g., knowledge is incomplete, multiple authors contribute, …
- **Requiring entered knowledge to be strictly / universally true (exception-free) is impractical**
  - Precludes stating generalities (the typical) and thus the population of authors
  - "The perfect is the enemy of the good"
- **Exceptions manifest as contradictions, i.e., conflict**
- **Leveraging multiple sources of knowledge (e.g., KB merging) requires conflict resolution**
  - Errors. Confusions. Omitted context.

Defeasibility is Indicated When…

- **Useful generalities – <u>and</u> potential exceptions – coexist**
  - Specify knowledge in detail/precision <u>appropriate</u> for various circumstances
- **Governing doctrine, definitions, or other knowledge, cannot be assured to be conflict-free, e.g.:**
  - Multiple sources of governing doctrine exist
    - Typically, no central authority resolves all conflict promptly
  - Truth depends on context
    - Yet context is rarely made fully explicit
- **Many broad realms are full of exceptions**
  - Policies, regulations, laws — and the workflows they drive
    - Multiple jurisdictions, organizations, contracts, origins
  - Learning and science. Updating. Debate.
    - May falsify previous hypotheses after observation or communication
  - Causal processes: changes to state, from interacting/multiple causes
  - Natural language (text interpretation): “there’s a gazillion special cases”

EECOMS Example of Conflicting Rules: Ordering Lead Time

- Vendor's rules that prescribe how buyer must place or modify an order:
- A) 14 days ahead if the buyer is a qualified customer.
- B) 30 days ahead if the ordered item is a minor part.
- C) 2 days ahead if the ordered item's item-type is backlogged at the vendor, the order is a modification to reduce the quantity of the item, and the buyer is a qualified customer.
- D) 45 days ahead if the buyer is a walk-in customer.

- Suppose more than one of the above applies to the current order? **Conflict!**
- Helpful Approach: **<u>precedence</u>** between the rules.
  - E.g., D is a catch-case: A > D , B > D , C > D
- Often only *partial* order of precedence is justified.
  - E.g., C > A , but no precedence wrt B vs. A, nor wrt C vs. B.

Ordering Lead Time Example in LP with Courteous Defaults

```
@prefCust   orderModifNotice(?Order,14days)   :-
        preferredCustomerOf(?Buyer,SupplierCo),   purchaseOrder(?Order,?Buyer,SellerCo) .
@smallStuff   orderModifNotice(?Order,30days)  :-
        minorPart(?Buyer,?Seller,?Order),   purchaseOrder(?Order,?Buyer,SupplierCo) .
@reduceTight   orderModifNotice(?Order,2days)   :-
        preferredCustomerOf(?Buyer,SupplierCo) and
        orderModifType(?Order,reduce) and
        orderItemIsInBacklog(?Order) and
        purchaseOrder(?Order,?Buyer,SupplierCo) .
\overrides(reduceTight,  prefCust) .    // reduceTight has higher priority than prefCust
// The below  exclusion constraint specifies that orderModifNotice is unique, for a given order.
\opposes(orderModifNotice(?Order,?X), orderModifNotice(?Order,?Y))   :-   ?X != ?Y .
```

- Rule D, and prioritization about it, were omitted above for sake of brevity.
- Above rules are represented in Logic Programs KR, using the Courteous defaults feature
- Notation:
  - ":-" means "if". "@…" declares a rule tag. "?" prefixes a logical variable.
    "\overrides" predicate specifies prioritization ordering.
    An exclusion constraint specifies what constitutes a conflict.
    "!=" means ≠ .

Example: Ontology Translation, leveraging hilog and exceptions

/* Company BB reports operating earnings using R&D operating cost which includes price of a small company acquired for its intellectual property. Organization GG wants to view operating cost more conventionally which excludes that acquisition amount. We use rules to specify the contextual ontological mapping. */

```
@{normallyBringOver}  ?categ(GG)(?item)  :- ?categ(BB)(?item).
@{acquisitionsAreNotOperating}   neg ?categ(GG)(?item) :-
     acquisition(GG)(?item) and (?categ(GG) :: operating(GG)).
\overrides(acquisitionsAreNotOperating, normallyBringOver).  /* exceptional */
acquisition(GG)(?item) :- price_of_acquired_R_and_D_companies(BB)(?item).
R_and_D_salaries(BB)(p1001).   p1001[amount -> $25,000,000].
R_and_D_overhead(BB)(p1002).   p1002[amount -> $15,000,000].
price_of_acquired_R_and_D_companies(BB)(p1003).   p1003[amount -> $30,000,000].
R_and_D_operating_cost(BB)(p1003).  /* BB counts the acquisition price item in this category */
R_and_D_operating_cost(GG) :: operating(GG).
Total(R_and_D_operating_cost)(BB)[amount -> $70,000,000].  /* rolled up by BB cf. BB's definitions */
Total(R_and_D_operating_cost)(GG)[amount -> ?x] :- … .  /* roll up the items for GG cf. GG's definitions */
```

As desired: |= R_and_D_salaries(GG)(p1001)

|= neg R_and_D_operating_cost(GG)(p1003) /* GG doesn't count it */

|= Total(R_and_D_operating_cost)(GG)[amount -> $40,000,000]

Notation: @{…} declares a rule tag. ? prefixes a variable. :- means if. X :: Y means X is a subclass of Y. \overrides(X,Y) means X is higher priority than Y.

Ex.'s: Causal Chains & Change in Biology

- **The change of state effected by process causality requires defeasibility in KR**
  - A cause's effect is an exception to the persistence of previous state
  - When two causes interfere, one's effect is an exception to the other's effect
- **Causal process reasoning is a large portion of AP Biology, often requiring multi-step causal chains and/or multiple grain sizes of description to answer a question**
- **E.g., Rulelog was piloted on such causal process reasoning in biology using SILK**
- **Hypothetical question about causal interference in an experiment:**
  1. "A researcher treats cells with a chemical that prevents DNA synthesis from starting.
  2. This treatment traps the cells in which part of the cell cycle?"

  Answer: G1 [which is a sub-phase of interphase]
- **Counterfactual hypothetical question:**
  1. " Suppose the typical number of chromosomes in a human liver cell was 12. [It's actually 46.]
  2. How many chromosomes would there be in a human sperm cell?"

  Answer: 6. [I.e., half the number in the liver and most organs.]

Priorities are available and useful

- Priority information is naturally available and useful. E.g.,
  - recency: higher priority for more recent updates
  - specificity: higher priority for more specific cases (e.g., exceptional cases, sub-cases, inheritance)
  - causality: higher priority for causal effects (direct or indirect) of actions than for inertial persistence of state ("frame problem")
  - authority: higher priority for more authoritative sources (e.g., legal regulations, organizational imperatives)
  - reliability: higher priority for more reliable sources (e.g., security certificates, via-delegation, assumptions, observational data).
  - closed world: lowest priority for catch-cases
- Many practical rule systems employ priorities of some kind, often implicit. E.g.,
  - rule sequencing in Prolog and production rules
    - Courteous LP subsumes this as a special case (totally-ordered priorities)
    - Also Courteous LP enables: merging, more flexible & principled treatment

Semantic KR Approaches to Prioritized LP

The currently most important for Semantic Web are:

1. [Courteous LP]
    - KR extension to normal LP
    - In RuleML, since 2001; in LegalRuleML, since 2012
    - Commercially implemented and applied
        - IBM CommonRules, since 1999
2. Defeasible Logic
    - Closely related to Courteous LP
        - Less general wrt typical patterns of prioritized conflict handling needed in e-business applications
        - In progress: theoretical unification with Courteous LP [Wan, Kifer, Grosof RR-2010]

Argumentation Theories approach to Defaults in LP

- **Combines Courteous + Hilog, and generalizes**
- **New approach to defaults: "argumentation theories"**
  - Meta-rules, in the LP itself, that specify when rules ought to be defeated
  - [Wan, Grosof, Kifer, *et al*. ICLP-2009; RR-2010]
- **Extends straightforwardly to combine with other key features**
  - E.g., Frame syntax, external Actions, Omniformity, …
- **Significant other improvements on previous Courteous**
  - Eliminates a complex transformation
  - Much simpler to implement
    - 20-30 background rules instead of 1000's of lines of code
  - Much faster when updating the premises
  - More flexible control of edge-case behaviors
  - Much simpler to analyze theoretically

Argumentation Theories approach, Continued*

- **More Advantages**
  - 1st way to generalize defeasible LP, notably Courteous, to HiLog higher-order and F-Logic frames
  - Well-developed model theory, reducible to normal LP
  - Reducibility results
  - Well-behavior results, e.g., guarantees of consistency
  - Unifies almost all previous defeasible LP approaches
    - Each reformulated as an argumentation theory
    - E.g., Defeasible Logic (see Wan, Kifer, and Grosof RR-2010 paper)
  - Cleaner, more flexible and extensible semantics
    - Enables smooth and powerful integration of features
    - Applies both to well founded LP (WFS) and to Answer Set Programs (ASP)
  - Leverages most previous LP algorithms & optimizations

- **Implemented** in Flora-2; used in SILK and Coherent Knowledge Systems

* Original name: LPDA = LP with Defaults and Argumentation Theories

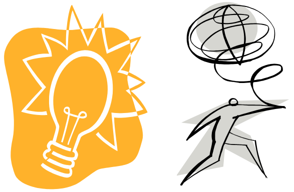
For More Info

- See the ff. longer AAAI-13 Rules tutorial, available at http://coherentknowledge.com/publications :
  - Benjamin Grosof, Michael Kifer, and Mike Dean. Semantic Web Rules: Fundamentals, Applications, and Standards (abstract). Conference Tutorial (Slides for 4-hour tutorial), 27th AAAI Conference on Artificial Intelligence (AAAI-13), Bellevue, Washington, July 15, 2013.
  - This is the latest iteration of a tutorial that since 2004 has been presented at numerous scientific conferences on web, semantic web, and AI.
  - A book is in early stages of preparation based on this tutorial.

- For Survey of KR's: also see 10/24/2013 session of Ontolog Forum
- For Rulelog overview: also see 6/20/2013 session of Ontolog Forum
- For Restraint: see [Grosof & Swift, AAAI-13] and [Andersen et al, RuleML-2013 and similar WLPE-2013] (all available at http://coherentknowledge.com/publications)

Acknowledgements

- Thanks to Michael Kifer and Mike Dean, co-authors of longer tutorial presentations upon which this presentation was based.



Thank You

Disclaimer: The preceding slides represent the views of the author(s) only.
All brands, logos and products are trademarks or registered trademarks of their respective companies and organizations.

OPTIONAL SLIDES FOLLOW

Declarative Logic Programs (LP) is the Core KR today

- **LP is the core KR of structured knowledge management today**
  - **Databases**
    - Relational, semi-structured, RDF, XML, object-oriented
    - SQL, SPARQL, XQuery
    - Each fact, query, and view is essentially a rule
  - **Business Rules – the commercially dominant kinds** (production/ECA rules, Prolog)
  - **Semantic Rules**
    - RuleML standards design, incl. SWRL. The main basis for RIF.
    - W3C Rule Interchange Format (RIF): -BLD, -Core. E.g., Jena tool.
  - **Extension: Rulelog.** E.g., Coherent's tool.
  - **Semantic Ontologies**
    - W3C RDF(S)
    - W3C OWL-RL (= the Rules subset). E.g., Oracle's tool for OWL.
  - **Overall: LP is "the 99%", classical logic is "the 1%"**

- **Relational DB's were the first successful semantic technology**
  - LP is the KR/logic that was invented to formalize them
- **The Semantic Web today is mainly based on LP KR** … and thus essentially equivalent to semantic rules
  - **You might not have realized that!**

Declarative *Logic Programs (LP) – Family of KR's*

- Normal LP
  - Rule syntax: $H \leftarrow B_1 \wedge \ldots \wedge B_k \wedge$ naf $B_{k+1} \wedge \ldots \wedge$ naf B_m . $(m \geq 0)$
    - H and Bi's are atoms.
    - $\leftarrow$ is a kind of implication that lacks contraposition.
      Its lhs and rhs are called the rule's "head" and "body", respectively.
    - naf ("negation-as-failure") is a kind of negation that is logically non-monotonic. Intuitively, naf Bi means "not believe Bi".
  - Semantics (well-founded) is defined constructively via an iterated fixed point.
    - It has 3 truth values: *true*; *false* in the naf sense; and an intermediate "*undefined*", which can represent paradoxicality.

HiLog

- A higher-order extension of predicate logic, which has a tractable first-order syntax
  - Allows certain forms of logically clean, yet tractable, meta-programming
  - Syntactically appears to be higher-order, but semantically is first-order and tractable
- Used in ISO Common Logic to syntactically extend FOL
  - Also appears promising for OWL Full and its use of RDF [Kifer; Hayes]
- Implemented in Flora-2 and SILK
  - Also partially exists in XSB, others
- [Chen, Kifer, Warren, "HiLog: A Foundation for Higher-Order Logic Programming", J. of Logic Programming, 1993]

Courteous LP: Advantages

- Facilitate updating and merging, modularity and locality in specification.
- Expressive: strong negation, partially-ordered prioritization, reasoning to infer prioritization.
- Guarantee consistent, unique set of conclusions.
  - E.g., never conclude both p and ¬p, nor that discount is both 5% and that it is 10%.
- Scalable & Efficient: low computational overhead beyond ordinary LPs.
  - Tractable given reasonable restrictions (VB + function-free):
    - extra cost is equivalent to increasing v to (v+2) in normal LP, worst-case.
  - By contrast, more expressive prioritized rule representations (e.g., Prioritized Default Logic) add NP-hard overhead.
- Modular software engineering:
  - Transform into normal LP, via argumentation theory approach

Ubiquity of Priorities
in Commercially Important Rules -- and Ontologies

- Updating in relational databases
  - more recent fact *overrides* less recent fact
- Static rule ordering in Prolog
  - rule earlier in file *overrides* rule later in file
- Dynamic rule ordering in production rule systems (OPS5)
  - "meta-"rules can specify agenda of rule-firing sequence
- Event-Condition-Action rule systems rule ordering
  - often static or dynamic, in manner above
- Exceptions in default inheritance in object-oriented/frame systems
  - subclass's property value *overrides* superclass's property value, e.g., method redefinitions
- All lack Declarative KR Semantics

Thank You

Disclaimer: The preceding slides represent the views of the author(s) only.
All brands, logos and products are trademarks or registered trademarks of their respective companies and organizations.