SILK: Knowledge Representation for Project Halo

Benjamin Grosof*

September 21, 2012

Invited presentation (30-min.) at the international symposium “Logic Programming: Systems and Applications” held at Stony Brook University (Stony Brook, New York, USA) in honor of David S. Warren’s retirement

*Vulcan Inc. (Seattle, WA), benjaming@vulcan.com, http://www.mit.edu/~bgrosof/


© Copyright Vulcan Inc. and Benjamin Grosof, 2012. All rights reserved.
Intro: SILK = Semantic Inferencing on Large Knowledge

Highly expressive, yet practical, logic language and system for rules and ontologies

- Rule-based reasoning, knowledge authoring/UI, and web interchange
- Based on declarative logic programs (LP), primarily with well founded semantics
  - “The world runs on Oracle … LP”
- New logic: RuleLog, a kind of meta LP*. Extends normal LP. Defeasible. Reactive.
  - Provides core of what’s needed to reason over/with rules (not just to query over/with data, as in Datalog)
- Transforms knowledge from higher-abstraction authored form into operational lower form
- Built in Java over XSB Prolog and Flora-2, with a UI (incl. Eclipse plug-in)
- Portions are open IP: XSB enhancements, SILK KR language & theory, (coming) API
- Part of Vulcan’s overall Project Halo aimed at “Digital Aristotle”
- Largest rule research program in USA (that we’re aware of)
- Multi-institutional: primarily via contractors (above, + several others in smaller roles)
  - And in cooperation with:

* LP stands for declarative logic programs. RuleLog was formerly called “Hyper LP”.
SILK Objectives

Address fundamental requirements for scaling Semantic Web to widely-authored Very Large KBs in business and science that answer questions, proactively supply info, and reason powerfully

- **Expressiveness** + **Semantics** + **Scalability**
- Better knowledge representation and reasoning (KR), for esp. defaults & processes, incl. deeply chained complex causality
  - Performance scalability of reasoning, incl. knowledge updates
- **Social** scalability of reasoning and Knowledge Acquisition (KA)
  - Authoring and debugging by Subject Matter Experts (SMEs), not only knowledge engineers (KEs) or programmers. “The KR is the deep UI … for KA”.

Serve the Digital Aristotle vision of Vulcan’s overall Project Halo

- Put the bulk of the world’s **scientific** and similar knowledge (K) on-line, and …
- Do **question answering** (QA), with deep reasoning. Exam Q’s. Act as personal tutor.
- 1st year college level **Biology** is current domain focus
- App. pilot is **e-textbook** with suggested Q’s & A’s, on iPad [AAAI-12 winner best demo]
Unique Features of SILK KR … & KA

Core

- **Meta expressiveness of knowledge, via higher-order (HiLog) and rule id's**
- **Defeasibility:** Expressive power, via courteous argumentation theories
- **“Omnis”**: Complex head formulas, via omni-directionality and skolemization
  - FOL-looking: freely combine \( \forall, \exists, \land, \lor, \neg, \Rightarrow \) in head. In body too, extending Lloyd-Topor.
  - Stays intuitionistic: lacks “reasoning by cases”*. Sound import from FOL (even tho’ nonmon).
- **Computational scalability via restraint, a kind of bounded rationality**
  - Can ensure tractability, despite heavy use of (logical) functions
- … In combination with **reactiveness**: external queries, actions, events
- … Keeping: strong model-theoretic semantics, and same complexity as Horn LP

More

- **Rich knowledge interchange:** load-time import; dynamic queries; ontology mapping
  - Via KR mappings from/to: RDF, OWL, RIF, SPARQL, SQL; also Cyc, AURA
- **Supports Textual Logic approach:** “Logic for the text, of the text, by the text”
  - Questions and answers in text, authoring of knowledge in text
  - Logic-based mappings of text-to-logic, logic-to-text. Disambiguation assisted by user, tool.
- **Deep explanation of defeat and why-not.** (Critical for knowledge debugging.)
- Tabling/perf. advances: incremental; call abstraction; termination analysis

* As does unit resolution strategy in FOL. But this enables defeasibility and aids tractability.
Textual Logic: concept & vision

• **Text is the language of “Business Users” / SME’s:** for QA and KA
  • Scientists, business process owners, executives, lawyers, doctors, …

• **How to map text → logic and map logic → text?**
  • NLP field is messy: weak architecturally wrt flexibly composable, reusable components

• **Vision: logic-based mappings for restricted NL in KA and QA**
  • Uniform architecture with fine-grain composable, reusable rule/axiom sets
  • Build up KB’s collaboratively/communally, reuse/adapt NL resources such as WordNet
  • Progressively increase scope of logic-based mappings, relax restrictions on NL
  • Scope goal: eventually most interpretation/generation for very useful subset of NL

• **Newly coined name. Somewhat generalized concept.**
  • Much previous work in the same general direction of logic-based NLP, e.g.,
    AURA, Cyc, Prolog-y, Attempto, SBVR, Watson, Powerset, Len Schubert, Chris Manning, …
  • Focus on semantic aspects of NLP, well beyond lexical/syntactic
SILK Example using Textual Style

• @[id->g1] txtbk(“A eukaryotic cell has a nucleus.”); // except when it doesn’t 😊 ...
• @[id->g2] txtbk(“A red blood cell has no nucleus.”); // exception
• @[id->g3] txtbk(“A eukaryotic cell during anaphase has no nucleus.”); // exception

<Reformulated semi-automatically into … >

• @[id->i1, tag->r1] ?x(has(a(nucleus))) :- ?x(is(a(eukaryotic(cell))));
• @[id->i2, tag->r2] neg ?x(has(a(nucleus))) :- ?x(is(a(red(blood(cell))))) ;
• @[id->i3, tag->r3] neg ?x(has(a(nucleus))) :-
  ?x(is(a(eukaryotic(cell)(during(anaphase)))));
• @[id->i4] silk:overrides(r2,r1) and silk:overrides(r3,r1) ; // prioritization

sourceText(i1,g1); sourceText(i2,g2); sourceText(i3,g3);

<Additional background knowledge>

• @[strict] red(blood(cell)) ## eukaryotic(cell) ;
• ?x(is(a(?c))) <=> ?x # ?c ;
• eukaryotic(cell)(during(anaphase)) ## eukaryotic(cell) ;

<Some instance facts>

• cell41(is(a(eukaryotic(cell)))); cell52(is(a(red(blood(cell))))) ;

| = cell41(has(a(nucleus))) , neg cell52(has(a(nucleus)))
| =/= cell52(has(a(nucleus)))
Example Cont.’d; **SILKish**

- <More background knowledge, and another instance fact>
- @[strict] ?x(has(no(nucleus))) \(\iff\) neg ?x(has(a(nucleus))) ;
- exist(?y)^(?y(is(a(part(of(?x)))))) and ?y(is(a(nucleus)))) :- ?x(has(a(nucleus))) ;
- |= exist(?z)^(?z(partOf(?z,cell41) and ?z # nucleus) ;
- cell63(is(a(eukaryotic(cell)(during(anaphase)))))) ;
- |= cell63(has(no(nucleus)))

**The SILKish feature of SILK’s KR language helps support textual logic**

- A syntax extension that tightly mixes English textual syntax with logical syntax
- Defined, and implemented, via a transformation, like the rest of SILK’s KR
- Useful for intermediate steps within text interpretation and generation, in KA and QA
- E.g., during the semi-automatic reformulation from source text g1..g3 into i1..i4:

```
?x has a nucleus :- ?x is a eukaryotic cell
  :but ?x has no nucleus :- ?x is a red blood cell

\lor ?x is a eukaryotic cell (during anaphase);
```

A lot fewer parentheses!

“:but” specifies relative priority
“\" prefix escapes a keyword
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

- We’ve piloted SILK in biology, focusing on such causal process reasoning

- 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.]
SILK Component Architecture today

External Actions & Events

External Knowledge & Reasoners

KB #1 ... KB #n

Engine #1 ... Engine #m

KR Languages

• SILK, RIF-SILK
• RIF-BLD, OWL-RL
• SPARQL, RDF(S)
• SQL, AURA

API

(User Interface
- Query-Answer & Explanation
- Author, Update, & Browse KB
- Test & Debug (IDE)

Language
- Interchange
- Transform
- Parse & Serialize
- Abstract Syntax

Engine
- Query
- Update
- Act (externally)
- Trace

Sub-Reasoners

Flora-2

Cyc

XSB* (InterProlog) (ODBC)

* XSB does most of the heavy lifting in inference

External UI tools
- Protégé (OWL)
- Text Interpretation

External Actions & Events

External Knowledge & Reasoners

KB #1 ... KB #n

Engine #1 ... Engine #m

KR Languages

• SILK, RIF-SILK
• RIF-BLD, OWL-RL
• SPARQL, RDF(S)
• SQL, AURA

API

(User Interface
- Query-Answer & Explanation
- Author, Update, & Browse KB
- Test & Debug (IDE)

Language
- Interchange
- Transform
- Parse & Serialize
- Abstract Syntax

Engine
- Query
- Update
- Act (externally)
- Trace

Sub-Reasoners

Flora-2

Cyc

XSB* (InterProlog) (ODBC)

* XSB does most of the heavy lifting in inference

External Actions & Events

External Knowledge & Reasoners

KB #1 ... KB #n

Engine #1 ... Engine #m

KR Languages

• SILK, RIF-SILK
• RIF-BLD, OWL-RL
• SPARQL, RDF(S)
• SQL, AURA

API

(User Interface
- Query-Answer & Explanation
- Author, Update, & Browse KB
- Test & Debug (IDE)

Language
- Interchange
- Transform
- Parse & Serialize
- Abstract Syntax

Engine
- Query
- Update
- Act (externally)
- Trace

Sub-Reasoners

Flora-2

Cyc

XSB* (InterProlog) (ODBC)

* XSB does most of the heavy lifting in inference

External Actions & Events

External Knowledge & Reasoners

KB #1 ... KB #n

Engine #1 ... Engine #m

KR Languages

• SILK, RIF-SILK
• RIF-BLD, OWL-RL
• SPARQL, RDF(S)
• SQL, AURA

API

(User Interface
- Query-Answer & Explanation
- Author, Update, & Browse KB
- Test & Debug (IDE)

Language
- Interchange
- Transform
- Parse & Serialize
- Abstract Syntax

Engine
- Query
- Update
- Act (externally)
- Trace

Sub-Reasoners

Flora-2

Cyc

XSB* (InterProlog) (ODBC)

* XSB does most of the heavy lifting in inference

External Actions & Events

External Knowledge & Reasoners

KB #1 ... KB #n

Engine #1 ... Engine #m

KR Languages

• SILK, RIF-SILK
• RIF-BLD, OWL-RL
• SPARQL, RDF(S)
• SQL, AURA

API

(User Interface
- Query-Answer & Explanation
- Author, Update, & Browse KB
- Test & Debug (IDE)

Language
- Interchange
- Transform
- Parse & Serialize
- Abstract Syntax

Engine
- Query
- Update
- Act (externally)
- Trace

Sub-Reasoners

Flora-2

Cyc

XSB* (InterProlog) (ODBC)

* XSB does most of the heavy lifting in inference

External Actions & Events

External Knowledge & Reasoners

KB #1 ... KB #n

Engine #1 ... Engine #m

KR Languages

• SILK, RIF-SILK
• RIF-BLD, OWL-RL
• SPARQL, RDF(S)
• SQL, AURA

API

(User Interface
- Query-Answer & Explanation
- Author, Update, & Browse KB
- Test & Debug (IDE)

Language
- Interchange
- Transform
- Parse & Serialize
- Abstract Syntax

Engine
- Query
- Update
- Act (externally)
- Trace

Sub-Reasoners

Flora-2

Cyc

XSB* (InterProlog) (ODBC)

* XSB does most of the heavy lifting in inference

External Actions & Events

External Knowledge & Reasoners

KB #1 ... KB #n

Engine #1 ... Engine #m

KR Languages

• SILK, RIF-SILK
• RIF-BLD, OWL-RL
• SPARQL, RDF(S)
• SQL, AURA

API

(User Interface
- Query-Answer & Explanation
- Author, Update, & Browse KB
- Test & Debug (IDE)

Language
- Interchange
- Transform
- Parse & Serialize
- Abstract Syntax

Engine
- Query
- Update
- Act (externally)
- Trace

Sub-Reasoners

Flora-2

Cyc

XSB* (InterProlog) (ODBC)

* XSB does most of the heavy lifting in inference

External Actions & Events

External Knowledge & Reasoners

KB #1 ... KB #n

Engine #1 ... Engine #m

KR Languages

• SILK, RIF-SILK
• RIF-BLD, OWL-RL
• SPARQL, RDF(S)
• SQL, AURA

API

(User Interface
- Query-Answer & Explanation
- Author, Update, & Browse KB
- Test & Debug (IDE)

Language
- Interchange
- Transform
- Parse & Serialize
- Abstract Syntax

Engine
- Query
- Update
- Act (externally)
- Trace

Sub-Reasoners

Flora-2

Cyc

XSB* (InterProlog) (ODBC)

* XSB does most of the heavy lifting in inference
Core Approach – some details

- **HiLog** [Chen, Kifer, Warren J. Logic Programming 1993] [used in ISO Common Logic]
  - Syntactically higher-order but reduces to first-order
  - Predicate or function can itself be a complex term, e.g., a variable
  - Essence: atom $p(a,b)$ is transformed into $\text{holds3}(p,a,b)$
  - Reification: reified formula (even when non-atomic) is just a term

- **Defeasibility via courteous argumentation theories** [Wan, Grosof, Kifer ICLP-2009]
  - Each defeasible rule is transformed by adding a body literal: ..., naf defeated(<that rule>)
  - ~10-20 background (meta) rules specify principles of defeat, i.e., argumentation cf. debate
  - Leverages meta expressiveness
  - Far superior to previous approaches wrt development and computational effort (esp. KB updates)

- **“Omnis”: complex head formulas, via omni-directionality and skolemization**
  - Drive NEG inward. Reduce $(G \iff F)$ into $(G \text{ or neg F})$.
  - Skolemize EXIST (but “late” – after, rather than before, directionnalization).
  - OR is treated by **directionalizing**. E.g.,
    $\left[\text{tag} \rightarrow r\right] L_1 \text{ or } L_2 \text{ or } \ldots \text{ or } L_k : B$; where each $L_i$ is a strong literal (atom or neg of an atom) is transformed into a set of $k$ directional-variant rules, one per choice of head literal:
    $\left[\text{tag} \rightarrow r\right] L_1 : \text{neg } L_2, \text{neg } L_3, \ldots, \text{neg } L_k, B$ ;
    $\left[\text{tag} \rightarrow r\right] L_2 : \text{neg } L_1, \text{neg } L_3, \ldots, \text{neg } L_k, B$ ;
    $\ldots$
    $\left[\text{tag} \rightarrow r\right] L_k : \text{neg } L_1, \text{neg } L_2, \ldots, \text{neg } L_{k-1}, B$ ;
Rulelog Computational Complexity

- Rulelog transforms into (well founded) Normal LP, in low-degree polynomial time
- Rulelog complexity class is same as Normal LP, i.e., same as Horn LP
  - ... Under non-onerous restrictions, notably ...
  - VB == # of distinct variables per rule is bounded by a constant $v$. $v \sim 10$ in practice.
  - EP == each external query or action requires only polynomial time
  - Complexity == worst-case, for inferencing – to compute the whole model or to answer a query
- Normal LP complexity is polynomial in $H = | \text{Herbrand universe} |$
  - Under the non-onerous VB restriction. Inferencing is $O(H^{(2v+2)})$.
- Function-free case: $H$ is $O(N)$, where $N = | \text{KB} |$. Inferencing is polynomial.
  - E.g., data base management systems (DBMS): SQL and SPARQL (DataLog)
  - E.g., business rule management systems (BRMS): production rules, event-condition-action
- Functions imply infinite Herbrand universe. Inferencing is undecidable.
  - Recursion thru head can result in a query having an infinite # of answers
  - E.g.: $p(f(?x)) : - p(?x); p(a); \text{ Query } ?- p(?y);$ yields answers: a, f(a), f(f(a)), f(f(f(a))), ...
- Yet functions are desirable for higher-abstraction KR/KA
  - HiLog for meta knowledge, e.g., defeasibility and textual logic
  - Skolemization for existential knowledge
- How to cope, practically?
Restraint – bounded rationality

- **Voluntary restraint** as an approach to bounded rationality. Fully semantic.
- Utilize the $u$ truth value: $u$’s replace some $t$’s and/or $f$’s
  - $u = \text{undefined}$. $t = \text{true}$. $f = \text{false}$ cf. naf.
- Specify **declaratively** via (meta-)rules: **skip** some rules – in spirit of “don’t bother”
  - A skipped rule is treated as $u$ in truth value. By contrast, a defeated rule is treated as $f$.
  - The rules that specify skipping represent meta-K about what’s relevant, what matters
- Get weaker well-founded model
  - This *restrained* model is sound wrt the unrestrained model, even tho’ the KR is nonmonotonic
- Norse creation myth analogy: literals’ truth values emerge from the undefined mist

- Can use to ensure **tractability** of the model, and thus of inferencing (and # answers)
  - Limit the non-$u$ part of the restrained model to tractable size, instead of (potentially) infinite
- Basic special case: limit term depth of every head literal by a constant bound $d$
  - In forward/backward inferencing, just give up quickly if head/subgoal term depth exceeds limit
  - Implemented in XSB (version 3.3.7, details in manual)
- Bounded term depth ensures tractability. **Key observations underlying proof:**
  - The Herbrand universe (incl. literals) can be viewed as a dag of terms, rooted at the empty term
  - Its size $H$ is $O(N^d)$. Inferencing is thus polynomial, rather than undecidable.
    - We have generalizations of this case, as well
Classical Logic is a “Bubble”

• The semantic web demands logical reasoning

• Classical logic is the basis for most of today’s semantic web reasoning except SPARQL
  - W3C OWL, W3C RIF-BLD
  - OMG SBVR, ISO Common Logic

• In classical logic, unlike SILK, any contradiction makes everything garbage
  - Total brittleness
  - The odds of consistency drop almost exponentially with the # of axioms
Above:
http://www.dailymail.co.uk/sciencetech/article-1199149/Super-slow-motion-pictures-soap-bubble-bursting-stunning-detail.html
Advantages of SILK’s KR

• First to combine several advances on semantic rules from 20 years of research
• Higher-abstraction KR closer to human cognition and social pragmatics
  • Critical for business users (SMEs) and NL KA/UI
• Represents debate, trust, communication, evolving-ness of K, e.g., in science
• Elevates meta-data to meta-knowledge. Motto: “puts the meta in knowledge”.
• Breakthrough: graceful, robust handling of exceptions and contradictory conflicts
  • Prioritization knowledge arises naturally from: specificity, recency, authority, causality, reliability
  • Exceptions are easy and succinct to add. Prioritization tames conflict.
  • Minimizes need to modify previous knowledge, when adding new K
  • Critical for leveraging multiple source of knowledge (e.g., KB merging)
    • Errors. Confusions. Omitted context.
• Radically extends expressive power of SPARQL/SQL, RDF(S), OWL-RL, RIF-BLD
• Remedies major limitations of semantic web’s current KR foundation
  • Copes with knowledge quality and context. Escapes from FOL Bubble.
  • Profoundly more web-scalable: socially and computationally
• Redefines the KR playing field for semantic web, business rules, and rule-based process management

Limitations – sacrifices:
• Reasoning by cases. But can introduce selectively.
• Term depth – beyond what’s anticipated as relevant
Defeasibility cf. SILK is Indicated When…

- Useful generalities – and potential exceptions – coexist
  - Specify knowledge in detail/precision appropriate 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”
Potential Applications in Business, Gov’t., and Science

• Horizontal
  • Policies, incl. law, and policy-based workflows
    • Monitor, report, react, handle exceptions, execute, enforce, customize
    • Trust: confidentiality, authorization, compliance, governance
  • Ontology mapping/mediation and knowledge integration
    • Perspective: the mappings themselves constitute ontological knowledge. E.g., a dictionary.

• Vertical
  • Financial: reporting, regulatory compliance, systemic risk management
  • Biomed: pharma, e-science, clinical records and guidance, insurance
  • E-commerce: shopping & advertising, contracts, customer care, catalogs
  • Mobile and Social Networking: personalize communication

• Many use cases in RIF, RuleML, SWSL documents & prototypes
Future Work Directions

- KA experiment wrt {KA cost, QA competence}
  - Biology textbook chapter on Membranes [Campbell, 9th ed., ch. 7]
- Standardization of KR: RIF-RuleLog dialect submission (in draft)
  - In cooperation with RuleML and W3C
- More on defeasibility: deeper conflict handling
- More on restraint
- More on textual (a big area!)
- Commercialization* of SILK system
- Probabilistic, inductive & abductive (another big area!)
- Parallelization of inferencing

* Portions are open IP
SILK Contributors current/past (partial list)

- Vulcan (Benjamin Grosof, Mark Greaves, Dave Gunning, Peter Clark)
- Stony Brook University (Michael Kifer, P. Fodor; students H. Wan, S. Liang, P. Kuz.)
- Raytheon BBN Technologies (Mike Dean, Brett Benyo, C. Andersen, B. Ferguson)
- Terrance Swift, consultant
- Declarativa (Miguel Calejo)
- Cycorp (Keith Goolsbey, Doug Lenat, Jon Curtis)
- SRI International (Vinay Chaudhri, David Martin, Ken Murray)
- Richard Fikes, consultant (Stanford University)
- University of Toronto (Sheila McIlraith, S. Sohrabi, H. Ghaderi)
- Texas Tech University (Michael Gelfond, D. Inclezan)
- University of Amsterdam (Bert Bredeweg)
- University of Freiburg (Georg Lausen)
- University of Michigan (Michael Wellman)
- Boeing. University of Texas. Ontoprise GmbH.
- Raphael Volz, consultant
Acknowledgements

• SILK contributors
  • (previously listed)

• Other contributors to several key previous KR efforts
  • RuleML and SWSL (Semantic Web Services Language) standards designs
    • Notably: Harold Boley and Said Tabet
  • SweetRules and Flora-2 systems
Thank You

SILK – Transforming Knowledge

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

© This presentation is copyright Vulcan Inc., and Benjamin Grosof. All rights reserved.