“SILK – *Transforming Knowledge*”

Logic Programming with Defaults and Argumentation Theories

Paper presentation at ICLP-2009

25th International Conference on Logic Programming

July 17, 2009, Pasadena, California

Hui Wan*, Benjamin Grosof**, Michael Kifer*,
Paul Fodor*, Senlin Liang*

* Stony Brook University
** Vulcan Inc., http://silk.semwebcentral.org

Summary of LPDA Approach

- Use "meta-" rules in the LP itself to specify when rules ought to be defeated
- Well developed model theory, reducible to Normal LP
- Generalizes defeasible LP to HiLog higher-order and F-Logic frames
  - E.g., higher-order courteous defaults
- Unifies almost all previous defeasible LP approaches
- Simplifies implementation of defeasible LP
- Leverages most previous LP algorithms & optimizations
- Implemented as extension to Flora-2, a.k.a. SILK V1
  - Public release free for research use planned for ~ fall 2009
  - Use cases in science and business

Outline

- Introduction
  - Defeasible reasoning
  - Difficulties in defeasible reasoning
- LPDA framework
  - Syntax
  - Semantics
- Advantages of LPDA
- Implementation

Defeasible reasoning

- A form of common sense reasoning: rules can be true by default but may be defeated

- Application domains:
  - policies, regulations, and law
  - actions, change, and process causality
  - Web services
  - inductive/scientific learning
  - natural language understanding

- Existing approaches:
  - Courteous Logic Programs (Grosof)
    - The main approach used commercially to date (since IBM Common Rules 1999)
  - Defeasible logic (Nute et al.)
  - Prioritized defaults (Gelfond & Son)
  - Preferred answer sets (Brewka & Eiter)
  - Compiling preferences (Delgrande et al.)
  - … …

Examples of defeasible reasoning

Example 1:

@d1 *flies(X) :- bird(X).*

@d2 *neg flies(X) :- penguin(X).*

bird(tweety).

penguin(tweety).

overrides*(d2,d1).*

Explicit negation

Rule d2 has higher priority than rule d1

Answer: { *bird(tweety), penguin(tweety), neg flies(tweety)* }

Examples of defeasible reasoning

Example 2:

$@a$ p.
$@b$ q.
$@c$ s.
opposes(p, s).
opposes(q, s).
overrides(a, c).
overrides(c, b).

p and *s* cannot be true together.
Ditto for *q* and *s*.

Answer 1: $\{p\}$

Intuition: rule b is defeated by rule a, rule c is defeated by rule b.

Answer 2: $\{p, q\}$

Intuition: rule c is defeated by rule a, so rule b is not defeated.

- *No single intuition works for all application domains.*

Difficulties in defeasible reasoning

- How to do defeasible reasoning for higher-order logics, e.g., HiLog and F-logic?
  - They don't have the notion of a head-predicate in a rule

- How to *lift and reuse* the analysis and reasoning techniques from LP with NAF to LP with defaults ?
  - Proof theory. Efficient algorithms, e.g., for updating a rule base.
  - Semantics. Well-behavior. Complexity bounds.
  - *Would like this to be systematic and straightforward.*

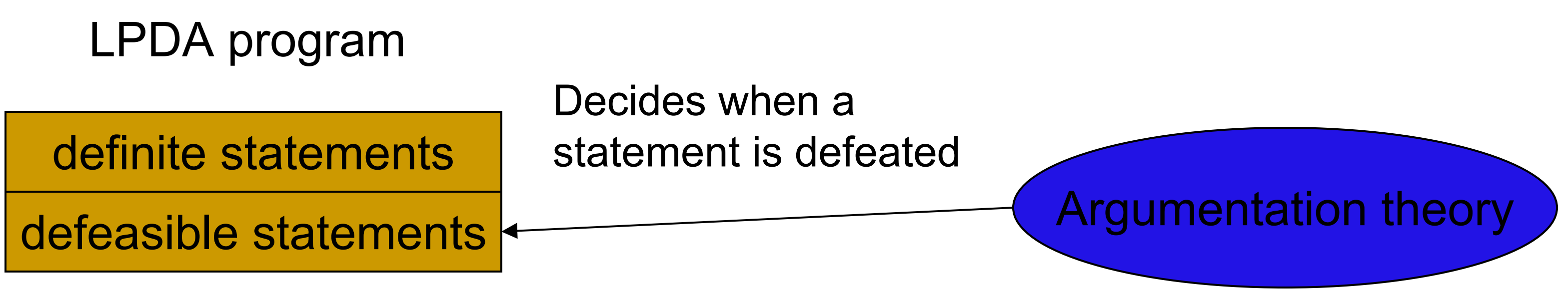
- How to integrate different intuitions and approaches, about the behavior of defeasibility itself, in a single reasoning system?
  - What is the "right" intuition, to apply in a given context?

LPDA Framework

- **L**ogic **P**rogramming with **D**efaults and **A**rgumentation theories

- LPDA program
  - Plain rules: non-defeasible statements
  - Labeled rules: defeasible statements

- Argumentation theory:
  - Defines reasoning arguments for defeating labeled rules



Argumentation Theory (AT)

- Composed of plain* rules
- A unary predicate, $defeated
- May also contain auxiliary predicates used in axioms that define $defeated.

  E.g., in <u>courteous</u> AT's:
  - overrides (prioritization), opposes (exclusion constraint)
    - These appear in user-authored domain-knowledge axioms
  - $refuted, $rebutted, $conflict
    - These typically appear only within the AT itself

- Most existing defeasible LP approaches can be described by some AT

* More generally, one can relax this.

AT Example

- $\mathcal{AT}^{\mathcal{GCLP}}$: captures the original generalized courteous LP (Grosof)

```
$defeated(?R)        :- $defeats(?S, ?R).
$defeats(?R, ?S)     :- $refutes(?R, ?S) or $rebuts(?R, ?S).

$refutes(?R, ?S)     :- $conflict(?R, ?S), overrides(?R, ?S).
$refuted(?R)         :- $refutes(?R2, ?R).
$rebuts(?R, ?S)      :- $conflict(?R, ?S),
                        not $refuted(?R), not $refuted(?S).

$candidate(?R)       :- body(?R, ?B), call(?B).
$conflict(?R, ?S)    :- $candidate(?R), $candidate(?S), opposes(?R, ?S).
opposes(?R, ?S)      :- opposes(?S, ?R).
opposes(?L1,?L2)     :- head(?L1, ?H), head(?L2, neg ?H).
```

Default negation (NAF)

Meta predicates (“Reflection”)

Explicit negation

LPDA Semantics: Least Model

- (Follows the general outline of Przymusinski for well-founded negation.)

  P : an lpda over language L
  AT : an AT over language L
  B_L : the Herbrand Base over language L
  M : a partial Herbrand interpretation – a set of literals in B_L
  Consider ground cases.

- M is a model of (P, AT) when it satisfies
  - every plain rule in $P \cup AT$
  - every labeled rule r in P such that \$defeated(r) is *not* in M

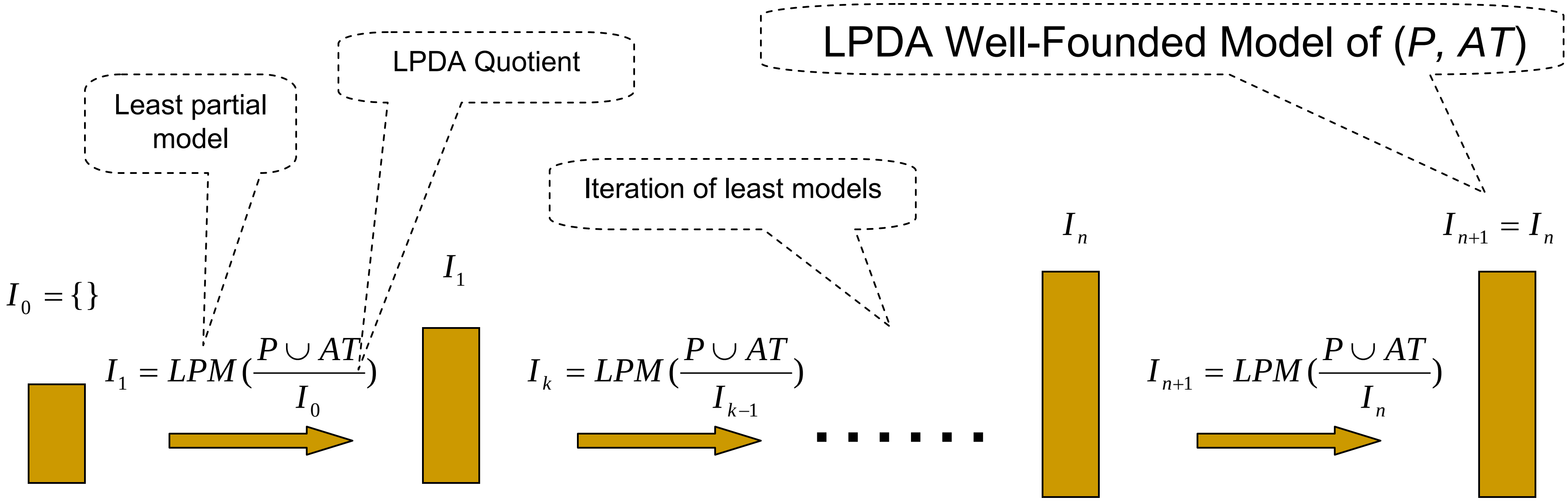
- M is a least model of (P, AT) when it is minimal with respect to ≤
  - $M1 \leq M2$ iff $M1^+ \subseteq M2^+$ and $M1^- \supseteq M2^-$
    - M^+ = the set of *not*-free literals in M; M^- = the set of *not* literals in M
    - (I.e., the usual notion of minimality for LP models; *not* here means NAF)

Iterative Quotient for LPDA

- The well-founded semantics for LPDA, like the WFS for NAF, is definable as an <u>iterated</u> least model
- Przymusinski's formulation of WFS for NAF uses a notion of the <u>quotient</u> of a rule set w.r.t. a partial interp.
  - Let Q be a set of rules, and J be a partial Herbrand interpetation for Q
  - The quotient $\frac{Q}{J}$ is obtained by:
    - In the body of each rule in Q, replace $not\ L$ by the truth value of $not\ L$ in J
- LPDA modifies this quotient notion so as to incorporate defeasibility. It adds:
    - For every labeled rule $@r\ L$:- $Body$ in Q,
      - If J($defeated($r$))=t, replace the rule with L :- $Body$, f
      - If J($defeated($r$))=u, replace the rule with L :- $Body$, u
      - (If J($defeated($r$))=f, replace the rule with L :- $Body$, t)
    - Remove rule labels

The resulting LPDA quotient is a set of plain rules without *not*

Well-Founded Model of LPDA



- <u>Thm</u>: LPDA WFM of (*P*, *AT*) = usual WFM of ($P' \cup AT$)
  - Where *P'* is obtained from *P* by:
    - For every labeled rule *@r L :- Body* ,
      - Replace it by: (see paper for details)

@r L :- Body, $\boxed{\textit{not \$defeated(r)}}$

Well-Behavior of AT

- Consistency:
  X and *neg* *X* cannot both be true

- Strong consistency:
  If *L1* and *L2* oppose each other
  then *L1* and *L2* cannot both be true

- Overriding property:
  @r1 *L1 :- Body1*
  @r2 *L2 :- Body2*
  If *L1* and *L2* oppose each other and *r1* overrides *r2* and *r1* is not defeated, then *r2* is defeated.

- Theorem: The courteous AT's (e.g., $\mathcal{AT}^{\mathcal{GCLP}}$) each ensure
  - consistency and strong consistency
    for the atoms appearing only in labeled rules
  - overriding property

Advantages of LPDA

- Generalizes defeasible LP to:
  - HiLog-style higher-order logics
  - F-logic style object-oriented features
- Unifies almost all previous defeasible LP approaches in one KR
  - Can combine multiple approaches in one system
  - Much simpler to analyze theoretically
- Simplifies implementation
  - LPDA In FLORA-2:
    - Only 20-30 rules per argumentation theory
      vs. 1000's of lines of code (in previous works*)
  - Easy to debug and experiment with argumentation theories
- Improves performance radically when updating rules in courteous
  - Eliminates need to re-run a complex transformation
- Reuses most previous LP algorithms and optimizations

* E.g., Delgrande *et.al* A framework for compiling preferences in logic programs, TPLP 2003

LPDA example with HiLog and F-logic

HiLog: higher-order variable

F-logic: *object*[*method*->*value*]

- Example 3

@perm(?t) *?priv(?usr) :- ?adm*[*states(?t)->?priv(?usr)*], *?adm[controls->?priv].*

// An administrator who controls some privilege can grant that privilege to any user.

overrides(perm(?t1), perm(?t2)) :- ?t1 > ?t2.

// More recent privilege assignments have higher priority.

Kevin[*states(2008)->neg print(Al)*].

Bob[*states(2009)->print(Al)*].

Kevin[controls->?].

Bob[*controls ->*{*print(?), neg print(?)*}].

With $\mathcal{AT}^{\mathcal{GCLP}}$, we get: *print(Al)*

Advantages of LPDA

- Generalizes defeasible LP to:
  - HiLog-style higher-order logics
  - F-logic style object-oriented features
- Unifies almost all previous defeasible LP approaches in one KR
  - Can combine multiple approaches in one system
  - Much simpler to analyze theoretically
- Simplifies implementation
  - LPDA In FLORA-2:
    - Only 20-30 rules per argumentation theory

      vs. 1000's of lines of code (in previous works*)
  - Easy to debug and experiment with argumentation theories
- Improves performance radically when updating rules in courteous
  - Eliminates need to re-run a complex transformation
- Reuses most previous LP algorithms and optimizations

* E.g., Delgrande *et.al* A framework for compiling preferences in logic programs, TPLP 2003

Implementation in Flora-2 / SILK V1

- LPDA implemented as extension to Flora-2 (version Cornsilk, a.k.a. SILK V1)
  - Part of Vulcan's SILK project (Semantic Inferencing on Large Knowledge) http://silk.semwebcentral.org
  - Free for research use, public release planned for ~ fall 2009
    - Selected users sooner
- Implemented 3 courteous argumentation theories
  - Original GCLP (1999)
  - A variant with alternative edge-case behavior/intuition
  - An expressive extension having generalized exclusion constraints that handle multi-way conflicts (not just pairwise)
- Use cases in college-level science, e-commerce, trust, …

Thank you!

Questions?

Reserve slides

Adjust AT Behavior

- Adjustment to $\mathcal{AT}^{\mathcal{GCLP}}$:

```
$defeated(?R)        :-  $defeats(?S, ?R), not $compromised(?S).
$defeated(?R)        :-  $disqualified(?R).
$defeats(?R, ?S)     :-  $refutes(?R, ?S) or $rebuts(?R, ?S).

$compromised(?X)     :-  $refuted(?R), $defeated(?R)..
$disqualified(?X)    :-  $defeats*(?X, ?X).

$refutes(?R, ?S)     :-  $conflict(?R, ?S), overrides(?R, ?S).
$refuted(?R)         :-  $refutes(?R2, ?R).
$rebuts(?R, ?S)      :-  $conflict(?R, ?S), not $compromised(?R),
                          not $refuted(?R), not $refuted(?S).

$candidate(?R)       :- body(?R, ?B), call(?B).
$conflict(?R, ?S)    :-  $candidate(?R), $candidate(?S), opposes(?R, ?S).
opposes(?R, ?S)      :-  opposes(?S, ?R).
opposes(?L1,?L2)     :-  head(?L1, ?H), head(?L2, neg ?H).
```

Transitive closure of $*defeats*