Constructive negation with the well-founded semantics for CLP

Włodek Drabent

Institute of Computer Science
Polish Academy of Sciences, Warszawa,
and IDA, Linköping University

Jan Maluszynski, Jakob Henriksson

SweCons May 2009

Summary

- Initial motivation: adding rules to ontologies for Semantic Web.
- **General framework: combining**
  - normal logic programs (non monotonic)
  - first order logic (monotonic).
- Instance: XSB Prolog + ontology reasoners.
- **Applicable** for adding negation for **CLP**
- Semantics – based on well-founded sem. of LP.
  - Efficient top-down operational semantics.
  - Re-use of existing engines possible.

Outline

- Related work
- The well-founded semantics
- Our framework
  - declarative semantics
  - operational semantics
-

Related work – negation for CLP

Stuckey '91,'95 – CLP with completion semantics

Fages '97 – CLP with comp. sem,
(based on Drabent '93,'95 – CLP($\mathcal{H}$), comp. sem., WFS)

Dix+Stolzenburg '98 – CLP, WFS, restricted class of programs, not goal-driven.

Negation in logic programming

Three semantics

- Negation as finite failure
  E.g. $P_1 = \{p \leftarrow p\}$, p neither true nor false
  - **Completion** semantics, Kunen, 3-valued
- Negation as (possibly) infinite failure
  E.g. Above: p false w.r.t. P_1.
  - **Well-founded** semantics
  - **Stable model** semantics (answer set sem.)

Well-founded semantics, informally

Facts of P – true
Ground A, not an instance of a rule head – false.

Iterate using in rule bodies the obtained results.

Well-founded vs **stable model semantics**

WF	**AS**
3-valued	2-valued
t,**u**,**f**	**t**,**f**

- Equivalent for stratified programs
- Ex. $\{a \leftarrow \neg b.\ \ b \leftarrow \neg a.\}$

  a,b **u**ndefined — two stable models $\{a,\neg b\}\ \ \{b,\neg a\}$
- Ex. $\{a \leftarrow \neg a\}$

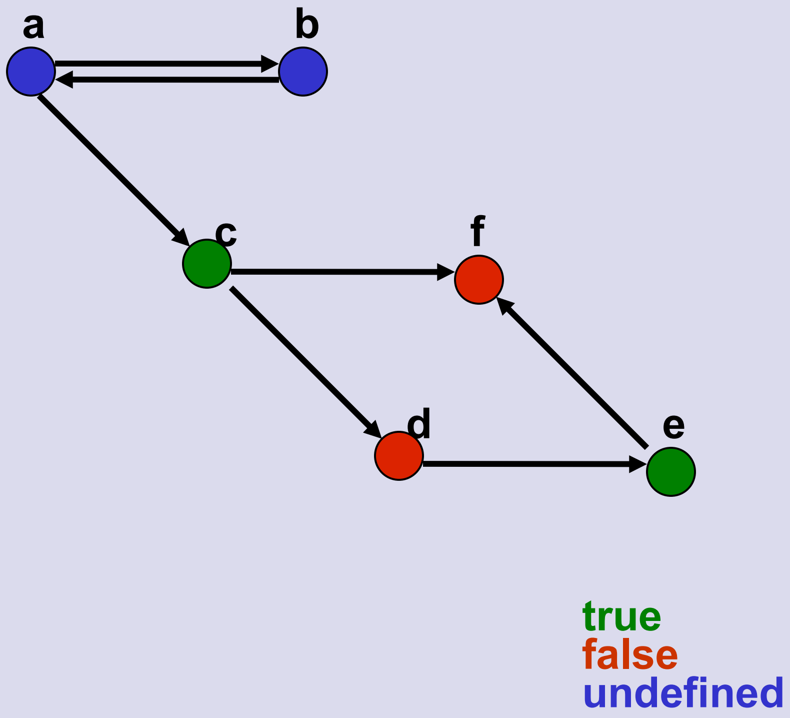
  a **u**ndefined — no stable models

Example: Two-person game

Program P:

win(X) :- move(X,Y), ~ win(Y).

move(a,b).
move(b,a).
move(a,c).
move(c,d).
move(d,e).
move(c,f).
move(e,f).



Well-founded model of P:

WF(**P**) = move/2 ∪
{ win(c), win(e),
~ win(d), ~ win(f) }

Well-founded semantics generalized

- Program = set P of hybrid rules + external theory T
- Constraints – formulae of T allowed in rules,
  closed under ∃, ¬, ∧
- Hybrid rule – H :- C, $L_1,\ldots,L_n$
  normal clause, constraint allowed
- M – a model of T
- P/M – ground(P) with the constraints interpreted in M (i.e. replaced by **true** or **false**)
- WF(P/M) – the well-founded model of P/M
- $(T,P) \models_{wf} F$ iff F true in all well-founded models

WFS generalized, example; CLP(FD) or CLP($\mathbb{N}$)

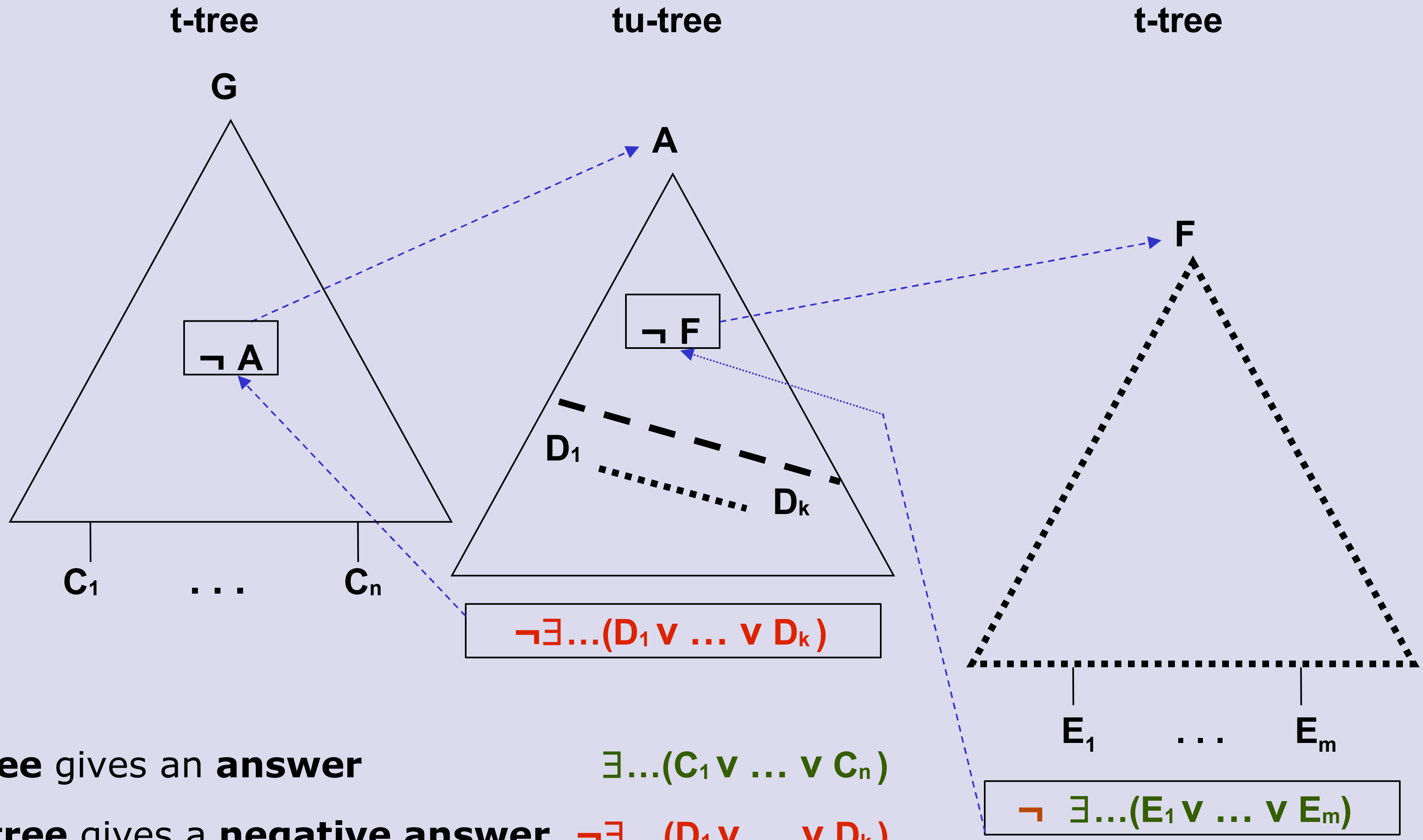
$$win(X) \leftarrow C(X,Y), \neg win(Y)$$

$\neg win(X)$	$win(X)$
should be implied by	
$\forall X_1 . \neg C(X, X_1)$	$\exists X_1 . C(X, X_1), \forall X_2 . \neg C(X_1, X_2)$
$\forall X_1 . \neg C(X, X_1) \vee \exists X_2 . C(X_1, X_2), \forall X_3 . \neg C(X_2, X_3)$	$\exists X_1 . C(X, X_1), (\forall X_2 . \neg C(X_1, X_2) \vee \exists X_3 . C(X_2, X_3), \forall X_4 . \neg C(X_3, X_4))$
$\ldots$	$\ldots$

Operational semantics for hybrid rules

- Generalizes SLS-resolution
- SLS-resolution: SLD-resolution + “infinite failure”
  - goals (conjunctions of literals) + substitutions
- Generalization:
  - goals include constraints, over
    original constraint domain + Herbrand domain
    - Usually CLP(X) means CLP($\mathcal{H}$+X)
- Top-down, goal-driven
  A tree of trees;   2 kinds of trees needed
  Non trivial handling of constraints, based on
  constructive negation for LP [D_'95]

Operational semantics, trees



t-tree gives an **answer** $\exists\ldots(C_1 \vee \ldots \vee C_n)$

tu-tree gives a **negative answer** $\neg\exists\ldots(D_1 \vee \ldots \vee D_k)$

Operational semantics, example

win(X) :- C(X,Y), ~ win(Y).

The constraints from the previous example obtained as answers / negative answers.

Operational semantics for hybrid rules (3)

- **Sound** under rather weak conditions
  ($\exists C \Rightarrow C\theta$ for some θ, or P safe, or special $\exists$)
- **Complete** when
  (1) decidability of constraints
  (2) no function symbols
  (3) safeness
- (1),(2) $\Rightarrow$ declarative semantics **decidable**

- H :- C, $L_1,\ldots,L_n$ **safe** iff each variable occurs
  (or C bounds it to a variable occuring)
  in a positive literal L_i.

Soundness (formally)

(P,T) a hybrid program, G goal, . . .

If

- C is an answer of a t-tree for G
- $T \models C\theta$

then

- $(T,P) \models_{\mathrm{wf}} G\theta$.

If

- C is a negative answer of a tu-tree for G
- $T \models C\theta$

then

- $(T,P) \models_{\mathrm{wf}} \neg G\theta$.

The computed answers are correct w.r.t. all well-founded models of (T,P).

Completeness (formally)

(P,T) a hybrid program, G goal

Additional requirements:

- Finite Herbrand universe
- P and G *safe*

- If $(T,P) \models_{wf} G\theta$ then there exists a t-tree for G with an answer C such that $T \models C\theta$
- If $(T,P) \models_{wf} \neg G\theta$ then there exists a tu-tree for G with a negative answer C such that $T \models C\theta$

Implementation

- Easy implementation by re-using reasoners

  for LP and external theory
- Prototype: XSB Prolog + Pellet DL reasoner
  - Constraints: DL concepts
  - Compilation to Prolog
  - Change of the DL reasoner – easy
- http://www.ida.liu.se/hswrl,
  usable, almost finished

  but constructive negation for LP omitted
- Written (mainly) in XSB. Compiling *P* to XSB.
  Run-time system in XSB (+ Pellet interface in Java).

Publications

www.ipipan.waw.pl/~drabent/

2 conference papers (RR2007), best short paper award,
1 workshop paper (ALPSWS2007).

Journal paper, invited to special issue of
“Knowledge and Information Systems”, delayed reviews.

The framework – properties

- Negation: monotonic for constraint predicates
  non-monotonic for rule predicates
- Normal rules (not disjunctive)
- No restrictions on alphabet,
  on models of external theories,
  on equality in external theories
  (no CET, UNA)

  - Prolog built-ins available
  - Logic + Control for rules (like in logic programming)
- Efficient – Few calls to DL solver

Summary

- Presented:
  - Generalization of WF semantics to CLP (and others)
  - Operational semantics
  - Complements known results for CLP with completion sem.
- Thus we know how negation can be dealt with.
  - except for stable model semantics
- Need for constructive negation? Examples?
  - We learned to live without it
- Use it in your programs!
  - possible even without a general implementation

THANK YOU

A comment on CLP theory

- Usually CLP(X) means CLP($\mathcal{H}$+X)
  - E.g. ?- p(2+2) fails with { p(4).}
  - Two equalities (of $\mathcal{H}$, of X)
- CLP($\mathcal{H}$) dealt with by unification
- CLP + negation
  – dealing with disequalities necessary
  – constructive negation for LP

A comment on the *win* example

For the example program,
the well-founded semantics is equivalent to the 3-valued completion semantics.