Some Title...

submitted 1 January 2003; revised 1 January 2003; accepted 1 January 2003

Mention that we may get an interval of uncertainty.
Maybe discuss frequentist and subjectivist interpretations. (2)

Preliminaries

FIRST PASS: ASSUME FIXED GODEL T NORM – WE CAN EXTEND TO OTHER TNORMS AND TO DISTRIBUTION SEMANTICS LATER.

Throughout this paper, we assume a general knowledge of logic programming terminology, including tabled resolution and the well-founded semantics including both default and explicit negation as in (1). Here, we include notation and definitions that will be used for our proof of the correctness of Plow.

An *annotated atom* is an atom A associated with an annotation n that is either a variable or $0 \leq n \leq 1$, denoted $A : n$. From an annotated atom *A:n*, an *objective literal* O is formed as either $O = A : n$, termed a positive objective literal with $sign(O) = pos$; or aa $O = negA : n$, termed a negative objective literal with $sign(O) = neg$, and the neg symbol denotes *explicit negation*. In either case the annotation, n, is denoted as $annotation(O)$, while the underlying atom, A, is $atom(A)$. Two objective literals O_1 and O_2 with the same underlying atom are *homologs* if they have the same sign and *conjugates* otherwise. An objective literal $A : n$ is ground if both A and n are ground. From an objective literal O a default literal is formed as either a positive default literal O or as a negative default literal $naf\, O$, where naf denotes *default negation*. A default literal is sometimes simply called a *literal*. [1]

A rule has the form

$$r = O \leftarrow L_1, \ldots, L_n$$

where O is an objective literal and $L_0, \ldots, L_n$ are default literals.

A rule R is ground if all literals in R are ground; a program $\mathcal{P}$ is ground if all rules in $\mathcal{P}$ are ground.

Our attention is restricted to three-valued (partial) interpretations and models such as those extending the well-founded model. Each such interpretation $\mathcal{I}$ is represented as a pair of sets of ground objective literals: ($\mathcal{T}$,$za\mathcal{F}$).

Definition 1
Let P be a ground objective literal O with $annotation(O) = n$, and $\mathcal{I}$ an interpretation.
O is true in $\mathcal{I}$ if

- $\mathcal{T}$ contains an O_T such that O_T is a homolog of O and $annotation(O_T) \geq n$; and
- $\mathcal{T}$ does not contain a conjugate O_C of O with $annotation(O_C) \geq (1 - n)$; and

[1] When convenient, an objective literal $A : 1$ ($\neg A : 1$) is denoted simply as A ($\neg A$).

- $\mathcal{F}$ does not contain a homolog O_F of O with $annotation(O_F) \geq (1-n)$.

O is false in $\mathcal{I}$ if

- $\mathcal{T}$ does not contain a homolog O_T of O with $annotation(O_T) \geq (1-n)$; and either
  — $\mathcal{F}$ contains a homolog O_F of O with $annotation(O_F) \geq n$; or
  — $\mathcal{T}$ contains a conjugate O_C of O with $annotation(O_C) \geq n$.

O is $\mathbf{u}$ in $\mathcal{I}$ if it is neither true nor false in $\mathcal{I}$.

A positive literal O is true (false) in $\mathcal{I}$ if O is true (false) in $\mathcal{I}$; a negative literal $nafO$ is true in $\mathcal{I}$ if O is false in $\mathcal{I}$ and is false in $\mathcal{L}$ if O is true in $\mathcal{I}$. A literal is **u** in $\mathcal{I}$ if it is neither true nore false in $\mathcal{I}$.

Note that in the above definition the truth value **u** captures both the traditional case where a literal is undefined, along with the case where a literal is overdefined.

Example 1
Consider the interpretation $\mathcal{I}_1$ where

- $\mathcal{T} = \{p : 0.7, q : 0.5, r : 0.6, negr : 0.5\}$
- $\mathcal{F} = \{p : 0.4, q : 0.5\}$.

$p : 0.6$ is true in $\mathcal{I}$ and $p : 0.8$ false, but $p : 0.7$ is **u**; $q : 0.5$ is **u** in $\mathcal{I}$. $r : 0.4$ is true in $\mathcal{I}$ but $r : 0.7$ is false; $r : n$ is **u** for $0.5 \leq n \leq 0.6$.

Well-Founded Model

Motivation: consider the program

```
p:0.8:- not p:0.8.
p:0.5.
neg p:0.3.
```

We want a model with p:0.5 as true, p:0.7 as false and p:0.6 as u. However, if the rule `p:0.8:- not p:0.8.` were removed, p:0.6 would be false.

One of the most important formulations of stratification is that of *dynamic* stratification (3), which shows that a program has a 2-valued well-founded model iff it is dynamically stratified, so that it is the weakest notion of stratification consistent with the well-founded semantics. The original definition of dynamic stratification included neither explicit negation (neg) nor annotations; however the encapsulation of these features within the interpretations of Definition 1 allows the definitions in this section to be mostly unchanged from their original formulation.

As presented in (3), dynamic stratification computes strata via operators on interpretations of the form ($\mathcal{T}$,$\mathcal{F}$) where $\mathcal{T}$ and $\mathcal{F}$ are subsets of $\mathcal{H}_P$. Given a set $\mathcal{S}$ of ground objective literals, a ground objective literal $A : m \hat{\in} \mathcal{S}$ if $A : n \in \mathcal{S}$ with $n \geq m$.

Dynamic stratification is based on a series of reduction of a program in the usual manner.

Definition 2 (*Reduction of $\mathcal{P}$ modulo $\mathcal{I}$*)
Let $\mathcal{I}$ be an interpretation and $\mathcal{P}$ a program, both over the same langage $\mathcal{L}$. By the *reduction of P modulo $\mathcal{I}$* we mean a new program $\frac{P}{\mathcal{I}}$ obtained from P by performing the following operations:

1. Remove from $\mathcal{P}$ all rules that contain a literal that is false in $\mathcal{I}$.
2. Remove from all the remaining rules those literals that are true in $\mathcal{I}$

Each stratum is then based on the intepretation of the previous stratum if it exists commbined with a reduction.

Definition 3
For a normal program P, sets $\mathcal{T}$ and $\mathcal{F}$ of ground atoms and a 3-valued interpretation $I = (\mathcal{T}, \mathcal{F})$ (sometimes called a pre-interpretation):

$True_I^P(\mathcal{T}) = \{A|A$ is true in I; or there is a clause $B \leftarrow L_1, ..., L_n$ in P, a grounding substitution θ such that $A = B\theta$ and for every $1 \leq i \leq n$ either $L_i\theta$ is true in I, or $L_i\theta \hat{\in} \mathcal{T}\}$;
$False_I^P(\mathcal{F}) = \{A|A$ is false in I; or for every clause $B \leftarrow L_1, ..., L_n$ in P and grounding substitution θ such that $A = B\theta$ there is some i $(1 \leq i \leq n)$ such that $L_i\theta$ is false in I or $L_i\theta \hat{\in} \mathcal{F}\}$.

(3) shows that $True_I^P$ and $False_I^P$ are both monotonic, and defines $\mathcal{TR}_I^P$ as the least fixed point of $True_I^P(\emptyset)$ and $\mathcal{FA}_I^P$ as the greatest fixed point of $False_I^P(\mathcal{H}_P)$. In words, the operator $\mathcal{TR}_I^P$ extends the interpretation I to add the new atomic facts that can be derived from P knowing I; $\mathcal{FA}_I^P$ adds the new negations of atomic facts that can be shown false in P by knowing I (via the uncovering of unfounded sets). An iterated fixed point operator builds up dynamic strata by constructing successive partial interpretations as follows.

Definition 4 (*Iterated Fixed Point and Dynamic Strata*)
For a normal program P let

$$\begin{array}{rcl} WFM_0 & = & \langle \emptyset; \emptyset \rangle; \\ WFM_{\alpha+1} & = & WFM_\alpha \cup \langle \mathcal{TR}^P_{WFM_\alpha}; \mathcal{FA}^P_{WFM_\alpha} \rangle; \\ WFM_\alpha & = & \bigcup_{\beta<\alpha} WFM_\beta, \text{ for limit ordinal } \alpha. \end{array}$$

$WFM(P)$ denotes the fixed point interpretation WFM_δ, where δ is the smallest (countable) ordinal such that both sets $\mathcal{TR}^P_{WFM_\delta}$ and $\mathcal{FA}^P_{WFM_\delta}$ are empty. The *stratum* of atom A, is the least ordinal β such that $A \in WFM_\beta$.

(3) shows that $WFM(P)$ is in fact the well-founded model and that any undefined atoms of the well-founded model do not belong to any stratum – i.e. they are not added to WFM_δ for any ordinal δ. Thus, a program is *dynamically stratified* if every atom belongs to at least one stratum.

This section has considered normal logic programs extended both with explicit negation. We note that if all atoms in a program $\mathcal{P}$ have the annotation 1, Definition 4 reduces to the definition of the Well0founded Semantics with Explicit Negation (1), and if $\mathcal{P}$ also does not contain explicit negation, the definition reduces to the definition of Well-Founded Semantics in (3).

Goal-Driven Evaluation

(1)

Not yet incorporated

TES: not sure this def is necessary.

Definition 5 (*Unfounded Sets*)
Given a program $\mathcal{P}$, there is a dependency edge from an objective literal O_1 to an objective literal O_2 if there is a rule $R \in \mathcal{P}$ such that O_1 is the head of R and O_2 occurs in a literal L in the body of R. If O_2 occurs in a positive default literal the edge is positive; if O_2 occurs in a negative default literal the edge is negative.

There is a path between objective literals O_1 and O_2 in $\mathcal{P}$ if there is an edge bwtween O_1 and

O_2 in $\mathcal{P}$, or if there is a path between O_1 and an objective literal O_3 and there is an edge between O_3 and O_2.

The direct dependencies of an objective literal $\mathcal{O}$ are those literals to which $\mathcal{O}$ has a dependency edge. from $\mathcal{O}$.

An objective literal O is involved in a negative loop if there is a path from O to itself that involves a negative edge.

An objective literal O is *unfounded* if O is involved in a negative loop and every direct dependency of O is involved in a negative loop; or if every rule with head O has a non-empty body, and every direct dependency of O is unfounded.

Given a program $\mathcal{P}$ and interpretation $\mathcal{I}$, the *unfounded set* $\mathcal{U}_{\mathcal{I}}$ consists of all unfounded objective literals in $\frac{\mathcal{P}}{\mathcal{I}}$.

Unfounded sets will be used to separate the truth values of conjugates. If the semantics is equivalent to rewriting each rule r for p so that $naf\ conjugate(p)$ is in the body of r then a program like

```
p.
neg p :- not neg p.
```

will be able to derive neither p nor neg p.

References

J.J. Alferes, C. Damásio, and L. M. Pereira. A logic programming system for non-monotonic reasoning. *Journal of Automated Reasoning*, 14(1):93–147, 1995.

J. Halpern. *Reasoning about Uncertainty*. MIT, 2003.

T. Przymusinski. Every logic program has a natural stratification and an iterated least fixed point model. In *ACM Principles of Database Systems*, pages 11–21. ACM Press, 1989.