A Fixed Point Theorem for Non-Monotonic Functions

Zoltán Ésik[1] and P. Rondogiannis[2]

[1]University of Szeged, Hungary

[2]University of Athens, Greece

July 15-18, 2013

Outline

Outline

Logic programming supports a form of negation known as "negation-as-failure".

Intuitively:

The query $\sim A$ succeeds iff our attempt to prove A terminates and fails.

Example

```
works ← ~ sleeps.
sleeps.
talks ← ~ studies.
```

According to negation-as-failure, `works` should be taken as false because sleeps is true and `talks` should be taken as true because we can not prove the truth of `studies`.

Negation-as-failure has very important practical applications.

Example

In a data base for a university department there exists a relation `enrolled(Student,Course)`. If we do not use negation-as-failure, we must also have a relation `not-enrolled(Student,Course)`. Relations of this kind may be huge (without conveying essential information).

Main applications:

Logic Programming, Data Bases, Artificial Intelligence.

Main Semantic Approach:

Well-founded semantics [van Gelder, Ross and Schlipf, 1991]. It uses a logic based on three truth values (`True`, 0 and `False`). It can be proved that every logic program with negation has a distinguished *well-founded* model.

Example

Consider the program:

```
works ← ~ tired.
tired ← ~ sleeps.
sleeps.
```

The well-founded model of the program is:

$$M = \{(\texttt{sleeps}, \mathit{True}), (\texttt{tired}, \mathit{False}), (\texttt{works}, \mathit{True})\}$$

The well-founded model is usually constructed based on the syntax of the program. The program is partitioned into strata according to the dependencies through negation, and the computation of the model is performed starting from the lower strata and moving towards the upper ones.

Example

Consider the program:

$$\boxed{\texttt{p} \leftarrow \sim \texttt{p}.}$$

The well-founded model of the program is:

$$M = \{(\texttt{p}, 0)\}$$

The program can not be partitioned into strata. The value `0` assigned to `p` has the meaning "I can not decide if `p` is `True` or `False`".

Two problems with the well-founded approach:

- It does not give us any insight regarding the distinction between classical negation and negation-as-failure.
- Many of the properties of classical logic programming (without negation) seem to be lost under the well-founded semantics.

A logical approach to negation-as-failure:

P. Rondogiannis and W. W. Wadge. Minimum model semantics for logic programs with negation-as-failure. *ACM Transactions on Computational Logic* 6(2): 441-467 (2005).

Outline

Example

Consider the program:

```
works ← ~ sleeps.
sleeps.
talks ← ~ studies.
```

In the well-founded model the atoms `sleeps` and `talks` are both true. However, `sleeps` seems to be "truer" than `talks` (because there is a fact that asserts beyond any doubt that `sleeps` is true, while `talks` is true just because there is no indication that `studies` is true).

The above example seems to imply that we need different levels of True and False values:

$$\mathsf{F}_0 < \mathsf{F}_1 < \cdots < \mathsf{F}_\alpha < \cdots < 0 < \cdots < \mathsf{T}_\alpha < \cdots < \mathsf{T}_1 < \mathsf{T}_0$$

Example

The program:

$$\begin{array}{l} \texttt{works} \leftarrow \sim \texttt{sleeps}. \\ \texttt{sleeps}. \\ \texttt{talks} \leftarrow \sim \texttt{studies}. \end{array}$$

has as "special" model the following:

$$M = \{(\texttt{sleeps}, \mathsf{T}_0), (\texttt{studies}, \mathsf{F}_0), (\texttt{talks}, \mathsf{T}_1), (\texttt{works}, \mathsf{F}_1)\}$$

Definition:

An interpretation I of program P is a function from the set of atoms of P to the set of truth values $V = \{F_0, F_1, \ldots, F_\alpha, \ldots, 0, \ldots, T_\alpha, \ldots, T_1, T_0\}$.

Definition:

Let I be an interpretation of P. We extend I as follows:

- For every literal $\sim p$:

$$I(\sim p) = \begin{cases} T_{\alpha+1} & \text{if } I(p) = F_\alpha \\ F_{\alpha+1} & \text{if } I(p) = T_\alpha \\ 0 & \text{if } I(p) = 0 \end{cases}$$

- For every conjunction of literals:

$$I(l_1, \ldots, l_n) = \min\{I(l_1), \ldots, I(l_n)\}$$

Definition:

Let P be a program and I an interpretation of P. We will say that I *satisfies* a rule of P of the form $p \leftarrow l_1, \ldots, l_n$ if $I(p) \geq I(l_1, \ldots, l_n)$. Moreover, I is a *model* of P if I satisfies all the rules of P.

Definition:

Let P be a program, I an interpretation of P and $v \in V$. Then $I \parallel v = \{p \text{ in } P \mid I(p) = v\}$.

Definition:

Let I and J be interpretations of a given program P and α be a countable ordinal.

- We write $I =_\alpha J$, if for all $\beta \leq \alpha$, $I \parallel T_\beta = J \parallel T_\beta$ and $I \parallel F_\beta = J \parallel F_\beta$.
- We write $I \sqsubset_\alpha J$, if for all $\beta < \alpha$, $I =_\beta J$ and either $I \parallel T_\alpha \subset J \parallel T_\alpha$ and $I \parallel F_\alpha \supseteq J \parallel F_\alpha$, or $I \parallel T_\alpha \subseteq J \parallel T_\alpha$ and $I \parallel F_\alpha \supset J \parallel F_\alpha$. We write $I \sqsubseteq_\alpha J$ if $I =_\alpha J$ or $I \sqsubset_\alpha J$.
- We write $I \sqsubset J$, if there exists a countable ordinal α such that $I \sqsubset_\alpha J$.
- We write $I \sqsubseteq J$ if either $I = J$ or $I \sqsubset J$.

We define an *immediate consequence operator* for logic programs:

$$T_P(I)(p) = lub\{I(l_1, \ldots, l_n) \mid (p \leftarrow l_1, \ldots, l_n) \in P\}$$

Theorem [R&W 2005]:

For every logic program P, T_P has a least fixed-point M_P (with respect to $\sqsubseteq$) which is the least infinite-valued model of P (again with respect to $\sqsubseteq$).

Remark 1:

It can easily be seen that T_P is **not** monotonic with respect to $\sqsubseteq$ (and therefore one can not use the Knaster-Tarski theorem to get the fixed-point).

Remark 2:

In our comparison of interpretations we use $\sqsubseteq$ and not the obvious pointwise comparison $\leq$.

Outline

Motivation:

The proof of the above theorem was performed using techniques that were specifically tailored to the case of logic programs. Can we abstract away from the issues regarding logic programming, in order to obtain a general fixed-point theorem which can potentially be used in other research areas?

Abstract Setting:

Suppose that $(L, \leq)$ is a complete lattice in which the least upper bound operation is denoted by $\bigvee$. We assume that for each countable ordinal α, there is a preordering $\sqsubseteq_\alpha$ on L (subject to certain conditions to be described shortly), where $=_\alpha$ is the equivalence relation determined by $\sqsubseteq_\alpha$. We define $x \sqsubset_\alpha y$ iff $x \sqsubseteq_\alpha y$ but $x =_\alpha y$ does not hold. Define $\sqsubset = \bigcup_\alpha \sqsubset_\alpha$ and let $x \sqsubseteq y$ iff $x \sqsubset y$ or $x = y$.

Required Properties:

Given a countable ordinal α and $x \in L$, define

$$(x]_\alpha \quad = \quad \{y \in L : \forall \beta < \alpha \quad x =_\beta y\}.$$

We assume the following properties:

- **Property 1:** If $\alpha < \beta$, then $\sqsubseteq_\beta$ is included in $=_\alpha$.
- **Property 2:** $\bigcap_\alpha =_\alpha$ is the equality relation on L.
- **Property 3:** For each x and α and for any $X \subseteq (x]_\alpha$ there is some $y \in (x]_\alpha$ such that $X \sqsubseteq_\alpha y$, and for all $z \in (x]_\alpha$, if $X \sqsubseteq_\alpha z$ then $y \sqsubseteq_\alpha z$ and $y \leq z$. The element y is unique and is denoted by $\bigsqcup_\alpha X$.
- **Property 4:** If $X \subseteq L$ is not empty and $y =_\alpha x$ for all $x \in X$ then $y =_\alpha (\bigvee X)$.

Results:

Lemma:

$(L, \sqsubseteq)$ is a complete lattice.

We say that $f : L \to L$ is α-*continuous* if for all increasing ω-chains $x_0 \sqsubseteq_\alpha x_1 \sqsubseteq_\alpha x_2 \sqsubseteq_\alpha \ldots$, it holds that $f(\bigsqcup_\alpha x_n) =_\alpha \bigsqcup_\alpha f(x_n)$.

Theorem 1:

Suppose that $f : L \to L$ preserves the relation $\sqsubseteq_\alpha$ and is α-continuous, for all α. Then f has a least pre-fixed point with respect to the relation $\sqsubseteq$, which is also a fixed-point of f.

Results (continued):

The α-continuity requirement can be dropped:

Theorem 2:

Suppose that $\mathsf{f} : \mathsf{L} \to \mathsf{L}$ preserves the relation $\sqsubseteq_\alpha$ for all α. Then f has a least pre-fixed point with respect to the relation $\sqsubseteq$, which is also a fixed-point of f.

It can be shown that Theorem 1 gives as a special case Kleene's fixed-point theorem and Theorem 2 gives as a special case the Knaster-Tarski fixed-point theorem.

Outline

What has been achieved:

We have defined a fixed-point theorem for a class of non-monotonic functions over specially structured complete lattices.

Possible Applications:

- Higher-order logic programming with negation.
- Logic programming with preferences.
- Formal language theory.

Thank you!