On Correctness of Normal Logic Programs

Włodek Drabent

Institute of Computer Science, Polish Academy of Sciences (IPI PAN);

IDA, Linköpings universitet, Sweden

www.ipipan.waw.pl/~drabent/

LOPSTR 2022 2022-09-23

Version 1.1 compiled October 5, 2022

Outline
proving correctness of normal logic programs

- Introduction (. . . , semantics, specifications, correctness)
- Proving correctness, approach 1
- Proving correctness, approach 2
- Comparison with proving correctness w.r.t. the well-founded semantics
- Summary

Introduction

Reasoning about program properties

Correctness

(program results compatible with the specification)

In LP (logic programming), also

Completeness

(the program produces everything required by the specification)

This work – correctness of normal logic programs

(programs with negation as finite failure)

Note (on logic)

Natural to use a 4-valued logic (of Belnap)

$\mathbf{t}$ – success
$\mathbf{f}$ – failure
$\mathbf{u}$ – divergence
$\mathbf{tf}$ – success or failure

Will be encoded in the standard 2-valued logic.

Semantics

NAF, NAFF (negation as finite failure), SLDNF-resolution

We have it in Prolog

when sound usage of negation:

A fails $\to$ $\neg A$ succeeds
A succeeds with a most general answer
$\to$ failure of $\neg A$
otherwise $\to$ floundering

Declarative semantics [Kunen]. 3-valued logic ($\mathbf{t}$,$\mathbf{f}$,$\mathbf{u}$).

$comp(P) \models_3 Q$ for answers Q
$comp(P) \models_3 \neg Q$ for failed queries Q
of P

Th.: $comp(P) \models_3 F$ iff $T3_P \uparrow n \models_3 F$ for some $n < \omega$

$$\Updownarrow$$

$$T4_P \uparrow n \models_4 F$$

Semantics

NAF, NAFF (negation as finite failure), SLDNF-resolution

We have it in Prolog

when sound usage of negation:

A fails $\rightarrow$ $\neg A$ succeeds
A succeeds with a most general answer
$\rightarrow$ failure of $\neg A$
otherwise $\rightarrow$ floundering

Declarative semantics [Kunen].

3-valued logic ($\mathbf{t}$,$\mathbf{f}$,$\mathbf{u}$).

$comp(P) \models_3 Q$ for answers Q
$comp(P) \models_3 \neg Q$ for failed queries Q
of P

Th.: $comp(P) \models_3 F$ iff $T3_P \uparrow n \models_3 F$ for some $n < \omega$

$$\Updownarrow$$

$$T4_P \uparrow n \models_4 F$$

Program correctness, specifications

Without negation

Specification: an Herbrand interpretation $St \in \mathcal{HB}$
– the ground atoms allowed to be true

Program correctness, specifications

Without negation

Specification: an Herbrand interpretation $\mathit{St} \in \mathcal{HB}$
– the ground atoms allowed to be true

P correct w.r.t. St : $\mathit{St} \models Q$ for each answer Q of P.

Proving correctness [Clark'78]

Th.: P correct w.r.t. St if $\mathit{St} \models P$.

Obvious, important, neglected
Applicable in practice

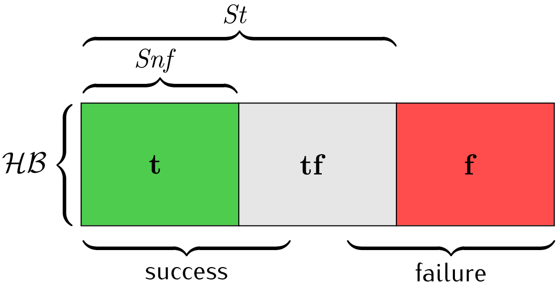
Program correctness, specifications

Without negation

Specification: an Herbrand interpretation $St \in \mathcal{HB}$
– the ground atoms allowed to be true

With negation

Specification: $(St, Snf) \in \mathcal{HB}^2$
St – as above
Snf – atoms not allowed to be false



$St \setminus Snf$ – **tf**
$St \cap Snf$ – **t**
$\mathcal{HB} \setminus Snf \setminus St$ – **f**
$Snf \setminus St$ – **u**

Specification, example

A specification for a program defining a list membership predicate m is (St_m, Snf_m), where

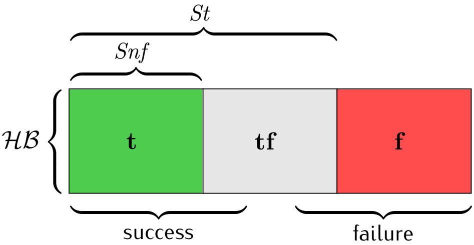
$$\begin{aligned} Snf_m &= \{\, m(e_i, [e_1, \ldots, e_n]) \in \mathcal{HB} \mid 1 \leq i \leq n \,\}, \\ St_m &= Snf_m \cup \{\, m(e, t) \in \mathcal{HB} \mid t \text{ is not a list} \,\}. \end{aligned}$$

So

$m(e, t)$ may be true when if t is a list then e is a member of t

$m(e, t)$ may be false when it is not of the form $m(e_i, [e_1, \ldots, e_n])$ $(1 \leq i \leq n)$

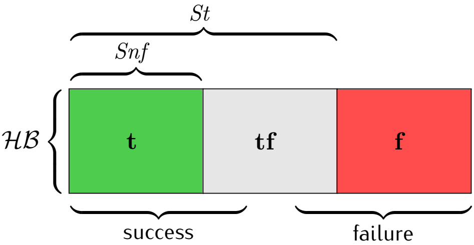
Correctness, definition



P correct w.r.t. (St, Snf) :
for each atom A
A is an answer of P
$\Rightarrow \; St \models A$
A fails $\Rightarrow \; Snf \models \neg A$

Non atomic queries?

Correctness, definition



P correct w.r.t. (St, Snf):
for each atom A
A is an answer of P
$\Rightarrow \; St \models A$
A fails $\Rightarrow \; Snf \models \neg A$

Non atomic queries?

Notation: New predicate symbol p' for each p

For programs, queries, ...

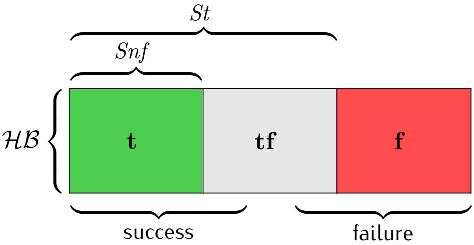
Q' – Q with $p \rightsquigarrow p'$ in each negative literal

Q'' – Q with $p \rightsquigarrow p'$ in each positive literal

For interpretations

St' – St with $p \rightsquigarrow p'$ in each literal

Correctness, definition



P correct w.r.t. (St, Snf) :
for each atom A
A is an answer of P
$\Rightarrow \; St \models A$
A fails $\Rightarrow \; Snf \models \neg A$

Non atomic queries?

P correct w.r.t. (St, Snf) : For each query Q
$St \cup Snf' \models Q'$ if Q is an answer of P
$St \cup Snf' \models \neg Q''$ if Q fails

Formally:

$$comp(P) \models_3 Q \quad \Rightarrow \quad St \cup Snf' \models Q'$$
$$comp(P) \models_3 \neg Q \quad \Rightarrow \quad St \cup Snf' \models \neg Q''$$

A detail for the next slide

$St \cup Snf' \models \vec{L}''$ means that in $\vec{L}$

each positive literal $L_i \in Snf$
each negative literal $L_j = \neg A_j$: $A_j \notin St$

Proving correctness. Approach 1

Df.: Atom $A \in \mathcal{HB}$ weakly covered by clause C w.r.t. $spec = (St, Snf)$
if $\exists$ a ground instance $A \leftarrow \vec{L}$ of C such that $St \cup Snf' \models \vec{L}''$.

Informally: A can be produced by C out of literals which cannot be false
(according to $spec$).

Df.: A weakly covered by program P if covered by some $C \in P$.

Intuition: Such A cannot be made false.

Th. (Cor. 1): P is correct w.r.t. $spec = (St, Snf)$ if

1. $St \cup Snf' \models P'$, and
2. each atom $A \in Snf$ is weakly covered by P w.r.t. $spec$.

Similarities with methods for programs without negation
1. – condition for correctness
2. – part of condition for completeness

Proving correctness. Approach 1

Df.: Atom $A \in \mathcal{HB}$ weakly covered by clause C w.r.t. $spec = (St, Snf)$ if $\exists$ a ground instance $A \leftarrow \vec{L}$ of C such that $St \cup Snf' \models \vec{L}''$.

Informally: A can be produced by C out of literals which cannot be false (according to $spec$).

Df.: A weakly covered by program P if covered by some $C \in P$.

Intuition: Such A cannot be made false.

Th. (Cor. 1): P is correct w.r.t. $spec = (St, Snf)$ if

1. $St \cup Snf' \models P'$, and
2. each atom $A \in Snf$ is weakly covered by P w.r.t. $spec$.

Similarities with methods for programs without negation

1. – condition for correctness
2. – part of condition for completeness

Example

Program SS:

$$ss(L, M) \leftarrow \neg\, nss(L, M).$$ % subset

$$nss(L, M) \leftarrow m(X, L), \neg\, m(X, M).$$ % non subset

$$m(X, [X|L]).$$
$$m(X, [Y|L]) \leftarrow m(X, L).$$ % member

Specification (St, Snf),

$$St = St_{ss} \cup St_{nss} \cup St_m, \qquad Snf = Snf_{ss} \cup Snf_{nss} \cup Snf_m,$$

$$\begin{array}{lcl}
St_{ss} & = & \{\, ss(l, m) \in \mathcal{HB} \mid l \text{ and } m \text{ are lists} \rightarrow l \subseteq m \,\}, \\
Snf_{ss} & = & \{\, ss(l, m) \in \mathcal{HB} \mid l \text{ and } m \text{ are lists} \land l \subseteq m \,\}, \\
St_{nss} & = & \{\, nss(l, m) \in \mathcal{HB} \mid l \text{ and } m \text{ are lists} \rightarrow l \not\subseteq m \,\}, \\
Snf_{nss} & = & \{\, nss(l, m) \in \mathcal{HB} \mid l \text{ and } m \text{ are lists} \land l \not\subseteq m \,\}, \\
St_m & = & Snf_m \cup \{\, m(e, t) \in \mathcal{HB} \mid t \text{ is not a list} \,\}. \\
Snf_m & = & \{\, m(e_i, [e_1, \ldots, e_n]) \in \mathcal{HB} \mid 1 \le i \le n \,\},
\end{array}$$

Example (2)

Let us take the least obvious part of the proof.

$$C \;=\; nss(L,M) \leftarrow m(X,L), \neg\, m\,(X,M).$$

$$\begin{array}{lcl} St_{nss} & = & \{\, nss(l,m) \in \mathcal{HB} \mid l \text{ and } m \text{ are lists} \rightarrow l \not\subseteq m \,\}, \\ Snf_m & = & \{\, m(e_i,[e_1,\ldots,e_n]) \in \mathcal{HB} \mid 1 \leq i \leq n \,\}, \\ St_m & = & Snf_m \cup \{\, m(e,t) \in \mathcal{HB} \mid t \text{ is not a list} \,\}. \end{array}$$

Showing 1. $St \cup Snf' \models C'$

Example (2)

Let us take the least obvious part of the proof.

$$C' = nss(L, M) \leftarrow m(X, L), \neg\, m'(X, M).$$

$$\begin{array}{rcl} St_{nss} & = & \{\, nss(l, m) \in \mathcal{HB} \mid l \text{ and } m \text{ are lists} \rightarrow l \not\subseteq m \,\}, \\ Snf_m & = & \{\, m(e_i, [e_1, \ldots, e_n]) \in \mathcal{HB} \mid 1 \leq i \leq n \,\}, \\ St_m & = & Snf_m \cup \{\, m(e, t) \in \mathcal{HB} \mid t \text{ is not a list} \,\}. \end{array}$$

Showing 1. $St \cup Snf' \models C'$

Take a ground instance $C'\theta = nss(l, m) \leftarrow m(x, l), \neg\, m'(x, m)$.
Assume the body is true, $m(x, l) \in St$, $m'(x, m) \notin Snf'$.
l, m are lists $\Rightarrow$ $x \,\epsilon\, l$ and $x \not\epsilon\, m$ $\Rightarrow$ $l \not\subseteq m$
$\Rightarrow$ $nss(l, m) \in St_{nss}$.

Example (2)

Let us take the least obvious part of the proof.

$$C' \;=\; nss(L,M) \leftarrow m(X,L), \neg\, m'(X,M).$$

$$\begin{array}{lcl} St_{nss} & = & \{\, nss(l,m) \in \mathcal{HB} \mid l \text{ and } m \text{ are lists} \rightarrow l \not\subseteq m \,\}, \\ Snf_m & = & \{\, m(e_i, [e_1, \ldots, e_n]) \in \mathcal{HB} \mid 1 \leq i \leq n \,\}, \\ St_m & = & Snf_m \cup \{\, m(e,t) \in \mathcal{HB} \mid t \text{ is not a list} \,\}. \end{array}$$

Showing 1. $St \cup Snf' \models C'$

Take a ground instance $C'\theta = nss(l,m) \leftarrow m(x,l), \neg\, m'(x,m)$.
Assume the body is true, $m(x,l) \in St$, $m'(x,m) \notin Snf'$.
l, m are lists $\;\Rightarrow\; x \in l$ and $x \not\in m \;\Rightarrow\; l \not\subseteq m$
$\Rightarrow\; nss(l,m) \in St_{nss}$.

Example (2)

Let us take the least obvious part of the proof.

$$
\begin{aligned}
C' &= nss(L, M) \leftarrow m(X, L), \neg\, m'(X, M). \\
St_{nss} &= \{\, nss(l, m) \in \mathcal{HB} \mid l \text{ and } m \text{ are lists} \rightarrow l \not\subseteq m \,\}, \\
Snf_m &= \{\, m(e_i, [e_1, \ldots, e_n]) \in \mathcal{HB} \mid 1 \leq i \leq n \,\}, \\
St_m &= Snf_m \cup \{\, m(e, t) \in \mathcal{HB} \mid t \text{ is not a list} \,\}.
\end{aligned}
$$

Showing 1. $St \cup Snf' \models C'$

Take a ground instance $C'\theta = nss(l, m) \leftarrow m(x, l), \neg\, m'(x, m)$.
Assume the body is true, $m(x, l) \in St,\ m'(x, m) \notin Snf'$.
l, m are lists $\Rightarrow\ x \mathrel{\epsilon} l$ and $x \not\mathrel{\epsilon} m\ \Rightarrow\ l \not\subseteq m$
$\Rightarrow\ nss(l, m) \in St_{nss}$.

Example (2)

Let us take the least obvious part of the proof.

$$
\begin{array}{rcl}
C' & = & nss(L,M) \leftarrow m(X,L), \neg\, m'(X,M). \\[6pt]
St_{nss} & = & \{\, nss(l,m) \in \mathcal{HB} \mid l \text{ and } m \text{ are lists} \rightarrow l \not\subseteq m \,\}, \\
Snf_m & = & \{\, m(e_i, [e_1, \ldots, e_n]) \in \mathcal{HB} \mid 1 \leq i \leq n \,\}, \\
St_m & = & Snf_m \cup \{\, m(e,t) \in \mathcal{HB} \mid t \text{ is not a list} \,\}.
\end{array}
$$

Showing 1. $St \cup Snf' \models C'$

Take a ground instance $C'\theta = nss(l,m) \leftarrow m(x,l), \neg\, m'(x,m)$.
Assume the body is true, $m(x,l) \in St$, $m'(x,m) \notin Snf'$.
l,m are lists $\Rightarrow$ $x \,\epsilon\, l$ and $x \not\epsilon\, m$ $\Rightarrow$ $l \not\subseteq m$

$$\Rightarrow \; nss(l,m) \in St_{nss}.$$

Example (3)

$$C \;=\; nss(L,M) \leftarrow m(X,L), \neg\, m(X,M).$$

$$\begin{array}{lcl}
Snf_{nss} & = & \{\, nss(l,m) \in \mathcal{HB} \mid l \text{ and } m \text{ are lists} \wedge l \not\subseteq m \,\}, \\
Snf_{m} & = & \{\, m(e_i, [e_1, \ldots, e_n]) \in \mathcal{HB} \mid 1 \leq i \leq n \,\}, \\
St_{m} & = & Snf_m \cup \{\, m(e,t) \in \mathcal{HB} \mid t \text{ is not a list} \,\}.
\end{array}$$

Showing 2. Each $A \in Snf_{nss}$ weakly covered by C.

Assume $nss(l,m) \in Snf_{nss}$.
$\Rightarrow\; l, m$ are lists, $l \not\subseteq m \;\;\Rightarrow\;\; \exists x\; x \,\epsilon\, l,\; x \not\epsilon\, m$
$nss(l,m) \leftarrow m(x,l), \neg\, m(x,m)$ is the required instance of C,
as $m(x,l) \in Snf$, $m(x,m) \notin St$.

Example (3)

$$C \;=\; nss(L,M) \leftarrow m(X,L), \neg\, m(X,M).$$

$$\begin{array}{lcl} Snf_{nss} & = & \{\, nss(l,m) \in \mathcal{HB} \mid l \text{ and } m \text{ are lists} \wedge l \not\subseteq m \,\}, \\ Snf_m & = & \{\, m(e_i,[e_1,\ldots,e_n]) \in \mathcal{HB} \mid 1 \leq i \leq n \,\}, \\ St_m & = & Snf_m \cup \{\, m(e,t) \in \mathcal{HB} \mid t \text{ is not a list} \,\}. \end{array}$$

Showing 2. Each $A \in Snf_{nss}$ weakly covered by C.

Assume $nss(l,m) \in Snf_{nss}$.
$\Rightarrow$ l, m are lists, $l \not\subseteq m$ $\Rightarrow$ $\exists x\; x \,\epsilon\, l,\; x \not\epsilon\, m$
$nss(l,m) \leftarrow m(x,l), \neg\, m(x,m)$ is the required instance of C,
as $m(x,l) \in Snf$, $m(x,m) \notin St$.

Example (3)

$$C \;=\; nss(L,M) \leftarrow m(X,L), \neg\, m(X,M).$$

$$\begin{array}{lcl} Snf_{nss} & = & \{\, nss(l,m) \in \mathcal{HB} \mid l \text{ and } m \text{ are lists} \wedge l \not\subseteq m \,\}, \\ Snf_{m} & = & \{\, m(e_i, [e_1, \ldots, e_n]) \in \mathcal{HB} \mid 1 \leq i \leq n \,\}, \\ St_{m} & = & Snf_m \cup \{\, m(e,t) \in \mathcal{HB} \mid t \text{ is not a list} \,\}. \end{array}$$

Showing 2. Each $A \in Snf_{nss}$ weakly covered by C.

Assume $nss(l,m) \in Snf_{nss}$.
$\Rightarrow\; l, m$ are lists, $l \not\subseteq m \;\;\Rightarrow\;\; \exists x\; x \in l,\; x \notin m$
$nss(l,m) \leftarrow m(x,l), \neg\, m(x,m)$ is the required instance of C,
as $m(x,l) \in Snf$, $m(x,m) \notin St$.

Example (3)

$$C \;=\; nss(L,M) \leftarrow m(X,L), \neg\, m(X,M).$$

$$\begin{array}{lcl} Snf_{nss} & = & \{\, nss(l,m) \in \mathcal{HB} \mid l \text{ and } m \text{ are lists} \wedge l \not\subseteq m \,\}, \\ Snf_m & = & \{\, m(e_i, [e_1,\ldots,e_n]) \in \mathcal{HB} \mid 1 \le i \le n \,\}, \\ St_m & = & Snf_m \cup \{\, m(e,t) \in \mathcal{HB} \mid t \text{ is not a list} \,\}. \end{array}$$

Showing 2. Each $A \in Snf_{nss}$ weakly covered by C.

Assume $nss(l,m) \in Snf_{nss}$.
$\Rightarrow$ l, m are lists, $l \not\subseteq m$ $\Rightarrow$ $\exists x\; x \,\epsilon\, l,\; x \not\epsilon\, m$
$nss(l,m) \leftarrow m(x,l), \neg\, m(x,m)$ is the required instance of C,
as $m(x,l) \in Snf$, $m(x,m) \notin St$.

Example (3)

$$C \;=\; nss(L,M) \leftarrow m(X,L), \neg\, m(X,M).$$

$$\begin{array}{lcl} Snf_{nss} & = & \{\, nss(l,m) \in \mathcal{HB} \mid l \text{ and } m \text{ are lists} \wedge l \not\subseteq m \,\}, \\ Snf_{m} & = & \{\, m(e_i, [e_1, \ldots, e_n]) \in \mathcal{HB} \mid 1 \leq i \leq n \,\}, \\ St_{m} & = & Snf_m \cup \{\, m(e,t) \in \mathcal{HB} \mid t \text{ is not a list} \,\}. \end{array}$$

Showing 2. Each $A \in Snf_{nss}$ weakly covered by C.

Assume $nss(l,m) \in Snf_{nss}$.
$\Rightarrow$ l, m are lists, $l \not\subseteq m$ $\Rightarrow$ $\exists x \; x \,\epsilon\, l,\; x \not\epsilon\, m$
$nss(l,m) \leftarrow m(x,l), \neg\, m(x,m)$ is the required instance of C,
as $m(x,l) \in Snf$, $m(x,m) \notin St$.

Limitation of Approach 1

Some facts cannot be proved.

Roughly
– we prove correctness w.r.t. the least fixed point of $T3_P$
– the semantics of P is given by $comp(P) \models_3$, or $T3_P \uparrow n$ $(n < \omega)$

Ex.: P: $p \leftarrow q(X).$
$q(s(X)) \leftarrow q(X).$

Correct w.r.t. $(St, Snf) = (\emptyset, \{p\})$. ($p$ is $\mathbf{u}$, each $q(t)$ is $\mathbf{f}$.)
The l.f.p. – everything is $\mathbf{f}$

Proving correctness. Approach 2

(Slightly modified w.r.t. the paper)

Introducing

level mappings, $|\ |\colon \mathcal{HB} \cup \neg\mathcal{HB} \;\to\; \mathbb{N} \cup \{\omega\}$,

restrictions on $|L|$ and the levels of L_i on which L depends

(in P).

Adjusted $|\ |$

Df.: $|\ |$ adjusted to P and $spec = (St, Snf)$ if

1. for each $A \in St$,
$$|A| \leq 1 + \min \left\{ \max\{|L| : L \in \vec{L}\} \;\middle|\; \begin{array}{l} A \leftarrow \vec{L} \in ground(P), \\ St \cup Snf' \models \vec{L}' \end{array} \right\}$$

2. for each $A \in \mathcal{HB} \setminus Snf$,
$$|\neg A| \leq$$
$$1 + \max \left\{ \min\left\{ |L| \;\middle|\; \begin{array}{l} L \in \vec{L}, \\ St \cup Snf' \models (\neg L)' \end{array} \right\} \;\middle|\; A \leftarrow \vec{L} \in ground(P) \right\},$$

3. for each $A \in Snf \setminus St$, $|A| = |\neg A| = \omega$.

$(\max \emptyset = 0, \min \emptyset = \omega)$

Sufficient condition. Approach 2

Th. (11): P correct w.r.t. $spec = (St, Snf)$ if
$\exists\ |\ |$ adjusted to $P, spec$ such that

1. $\forall\ A \leftarrow \vec{L} \in ground(P)$,
   if $St \cup Snf' \models \vec{L}'$ then $A \in St$ or $|A| = \omega$;
2. $\forall\, A \in Snf \quad \forall\, m \in \mathbb{N}$
   $\exists\, A {\leftarrow} \vec{L} \in ground(P) \quad \forall\, L \in \vec{L}$,
   $|L| > m$ or $St \cup Snf' \models L''$.

Ex. (12): Correctness of $\{\, p \leftarrow q(X).\ \ q(s(X)) \leftarrow q(X).\}$
w.r.t. $(St, Snf) = (\emptyset, \{p\})$.

1. holds trivially (clause bodies are false)
2. $|q(s^i(u))| = i$ where u not of the form $s(t)$

The well-founded semantics (WFS) [Ferrand,Deransart'93]

$|\ |$ into a well-ordered set $(W, \prec)$

Th.: P correct w.r.t. $spec = (St, Snf)$ under WFS if

1. $St \cup Snf' \models P'$, and ← as in Th. Cor. 1
2. $\exists$ a level mapping $|\ |: Snf \to W$
   $\forall A \in Snf \ \ \exists\, A \leftarrow \vec{L} \in ground(P)$
   2.1 $St \cup Snf' \models \vec{L}''$, and ← as in Th. Cor. 1
   2.2 for each positive literal L from $\vec{L}$, $|L| \prec |A|$. ← the difference

Ex.: $P = \{p \leftarrow p\}$, $spec = (St, Snf) = (\emptyset, \{p\})$

Correct under Kunen semantics, by Th. Cor. 1
Not correct under WFS; 2.2 does not hold

The well-founded semantics (WFS) [Ferrand,Deransart'93]

$|\ |$ into a well-ordered set $(W, \prec)$

Th.: P correct w.r.t. $spec = (St, Snf)$ under WFS if

1. $St \cup Snf' \models P'$, and ← as in Th. Cor. 1
2. $\exists$ a level mapping $|\ |\colon Snf \to W$
   $\forall A \in Snf \ \ \exists\, A \leftarrow \vec{L} \in ground(P)$
   2.1 $St \cup Snf' \models \vec{L}''$, and ← as in Th. Cor. 1
   2.2 for each positive literal L from $\vec{L}$, $|L| \prec |A|$. ← the difference

Ex.: $P = \{p \leftarrow p\}$, $spec = (St, Snf) = (\emptyset, \{p\})$

Correct under Kunen semantics, by Th. Cor. 1
Not correct under WFS; 2.2 does not hold

Summary

- Normal programs, Kunen semantics (NAFF, SLDNF-resolution) i.e. sound usage of negation in Prolog
- 4-valued logic encoded in standard 2-valued FOL
- Two sufficient conditions for program correctness
  Th. Cor. 1 based on [D_,Miłkowska'05]
  Th. Cor. 2 new
- Th. Cor. 1 can be
  - seen as formalization of common sense reasoning
  - (informally) applied in practice
- Future work (cooperation welcome)
  - Proving program completeness
  - Formalizing specifications and proofs
  - Correctness for ASP (Answer Set Programs)

www.ipipan.waw.pl/~drabent/

Thanks!

for your attention

Note. A limitation

Specifications as used here cannot express that

all ground instances $Q\theta$ of Q are possible answers (of a program) but Q is not.

(Such program/queries exist.)

Because

if $I \models Q\theta$ for each ground $Q\theta$
then $I \models Q$

(Do we need this?)