Extending the verification approach to finite failure

- a new method which uses different semantics and specifications
Motivations I

• Assume that the semantics of a program $P$ is defined as least fixpoint of a continuous operator $T$.
  Let $S$ be an interpretation which specifies the expected program semantics.
  
  – the program is partially correct w.r.t. $T$ iff $\text{lfp}(T) \subseteq S$.
  – a sufficient partial correctness condition is $T(S) \subseteq S$.

• Several verification methods, based on semantics modeling different observable properties of logic programming as
  
  – the ground success set [Shapiro82],
  – the correct answers [Ferrand87],
  – computed answers and their abstractions [Comini et al.99]

• In [Levi et al.98] it has been showed that all the existing methods can be reconstructed as an instance of a general verification technique where the property one wants to verify is simply an abstract semantics on a suitable abstract domain.

• There is one interesting property, finite failure, which is not an abstraction of none of the semantics used above.
Motivations II

- Which semantics for finite failure?
  - The ground finite failure set $\text{FF}_p$ is not correct w.r.t. finite failure.
  - The non ground finite failure set $\text{NGFF}_p$ is correct, fully abstract, w.r.t. finite failure and it is AND-compositional [Gori et al.97].

- **The problem**: there exist no fixpoint characterization of $\text{NGFF}_p$.

- **The idea**: use abstract interpretation to derive such fixpoint semantics.
  - start from a concrete traces semantics, which extends with infinite computations the concrete semantics of the Abstract Interpretation Framework [Comini et al.99]
  - define an abstract domain $\mathcal{S}$, chosen so as to model finite failure and to make the abstract operator $T^{\text{ff}}_p$ precise
  - the corresponding abstract fixpoint semantics $\text{lfp}(T^{\text{ff}}_p)$ is the Non-Ground Finite Failure set
  - it correctly models finite failure and is AND-compositional

- we can use the standard condition $T^{\text{ff}}_p(\mathcal{S}) \subseteq \mathcal{S}$ as a sufficient condition for the correctness w.r.t. finite failure

- we can define stronger conditions using Ferrand’s approach, based on two specifications
The Semantic Domain

Let $P$ be a program and $R$ be the set of atoms which finitely fail in $P$. Then

- $R$ is a downward closed set, i.e., if $A \in R \Rightarrow A \theta \in R$.
- **The key point:** $R$ enjoys a kind of “upward closure” property.
  **Example**
  Assume $\{p(a), p(f(a)), p(f(f(X))), p(f(f(a))), \ldots\} \in R$.
  Which behavior for $p(X)$?

  - Suppose $p(X)$ has a successful derivation.
    $p(X) \xrightarrow{\sigma_1} G_1 \xrightarrow{\sigma_2} \ldots, G_{n-1} \xrightarrow{\sigma_n} \Box$
    Let $\theta = \sigma_1 \ldots \sigma_n$.
    $\forall p(t) \in R, \forall \delta = \mathrm{mgu}(p(t), p(X)\theta), \text{ otherwise } p(t)\delta$

  - Suppose $p(X)$ has an infinite derivation.
    $p(X) \xrightarrow{\sigma_1} G_1 \xrightarrow{\sigma_2} \ldots, G_{n-1} \xrightarrow{\sigma_n} \ldots$
    Let $\theta_i = \sigma_1 \ldots \sigma_i$.
    $\forall p(t) \in R, \forall i \forall \delta_i = \mathrm{mgu}(p(t), p(X)\theta_i), \text{ otherwise } p(t)\delta_i$

    $\Downarrow$

    if $\forall$ possible sequences $\theta_1 :: \ldots :: \theta_n :: \ldots p(X)\theta_i \leq p(X)\theta_{i+1}$
    $\exists p(t) \in R, \text{ s.t. } \forall i \exists \delta_i = \mathrm{mgu}(p(t), p(X)\theta_i)$,
    then
    $p(X) \in R$. 

On the verification of finite failure
The Semantics Domain II

\[ \uparrow_{p(x)}^{ff} (R) = R \cup \{ p(x) \theta \mid \text{for all (possibly infinite) sequences} \]
\[ \theta_1 :: \ldots :: \theta_n :: \ldots, p(x) \theta_i \leq p(x) \theta_{i+1} \]
\[ \exists p(t) \in R \text{ s.t.} \]
\[ \forall i, p(t) \text{ unifies with } p(x) \theta \theta_i \}

\[ \bigcup_{p(x)} \uparrow_{p(x)}^{ff} \] is a closure operator.

\[ \mathcal{S} \] is the domain of downward closed sets of atoms, which are also closed w.r.t. \[ \bigcup_{p(x)} \uparrow_{p(x)}^{ff} \].

\[ (\mathcal{S}, \subseteq) \] is a complete lattice,

- the least upper bound of \[ R_1, R_2 \in \mathcal{S} \] is \[ \bigcup_{p(x)} \uparrow_{p(x)}^{ff} (R_1 \cup R_2) \]
- the greatest lower bound of \[ R_1, R_2 \in \mathcal{S} \] is \[ (R_1 \cap R_2) \]
The Fixpoint Semantics

\[ T_p^{ff}(I) = \{ p(\hat{t}) \mid \text{for every clause defining the procedure } p, \]
\[ p(t) : -B \in P \]
\[ p(\hat{t}) \in up_{p(x)}^{ff}(Nunif_{p(x)}(p(t))) \cup \]
\[ \{ p(t)\delta \mid \delta \text{ is a relevant for } p(t), \]
\[ B\delta \in up_B^{ff}([B\sigma \mid B = B_1, \ldots, B_n \exists B_i\sigma \in I]]) \}

- \( T_p^{ff} \) is continuous \( \Rightarrow \) \( lfp(T_p^{ff}) = up_{p(x)}^{ff}(U_{i<\omega} T_p^{ff} \uparrow i) \)

Example

\[ p \]
\[ q(a) : -p(X) \]
\[ p(f(X)) : -p(X) \]
\[ T_p^{ff} \uparrow 1 = \{ q(f(X)), q(f(f(X))), \ldots \]
\[ q(f(a)), q(f(f(a))), \ldots \]
\[ p(a) \}

\[ T_p^{ff} \uparrow 2 = T_p^{ff} \uparrow 1 \cup \{ p(f(a)) \}
\]
\[ \vdots \]
\[ T_p^{ff} \uparrow \omega = T_p^{ff} \uparrow \omega \cup \{ p(f(f(a))), p(f(f(f(a)))), \ldots \} \]
\[ p(X) \not\in up_{p(x)}^{ff}(T_p^{ff} \uparrow \omega) \text{ since} \]
\[ \exists \delta_1 = \{ X/f(Y) \} : \delta_2 = \{ X/f(f(Y)) \} : \delta_3 = \{ X/f(f(f(Y))) \} : \ldots , \]
\[ and \ \forall p(t) \in T_p^{ff} \uparrow \omega \ \forall i \ \not\exists \delta_i = \mgu(p(t), p(X)\delta_i). \]
\[ \downarrow \]
\[ q(a) \not\in T_p^{ff} \uparrow \omega + 1 \]
Ferrand’s approach

- Ferrand in [Ferrand93] uses the standard ground consequence operator \( T_P \)
- The specifications are
  - \( S \), intended \( \text{lfp}(T_P) \)
  - \( S' \), intended \( \text{gfp}(T_P) \)
- \( \text{lfp}(T_P) \subseteq S \).
  The standard sufficient condition for partial correctness \( T_P(S) \subseteq S \) allows us to reason about the ground success set
- \( S' \subseteq \text{gfp}(T_P) \).
  The new sufficient condition \( S' \subseteq T_P(S') \) is somewhat related to missing answers
Verification conditions based on $T_p^{ff}$

- $T_p^{ff}$ is not co-continuous
  - this is also the case for Ferrand’s $T_p$
- we replace $gfp(T_p^{ff})$ by $T_p^{ff} \downarrow \omega$
  - we have proved that $T_p^{ff} \downarrow \omega$ is the complement of the set of (possibly non-ground) atoms which have a successful derivation
- the standard verification condition
  - $S$ is the intended set of (possibly non-ground) atoms which have a finite failure
  - correctness
    $$lfp(T_p^{ff}) \subseteq S$$
  - sufficient condition for correctness
    $$T_p^{ff}(S) \subseteq S$$
- the new verification condition
  - $S'$ is the intended set of (possibly non-ground) atoms which do not have a successful derivation
  - correctness
    $$S' \subseteq T_p^{ff} \downarrow \omega \Rightarrow H_v \setminus T_p^{ff} \downarrow \omega \subseteq H_v \setminus S'$$
  - sufficient condition for correctness
    $$S' \subseteq T_p^{ff}(S')$$
Towards effective verification conditions

- the sufficient conditions $T_{p}^{ff}(S) \subseteq S$ and $S' \subseteq T_{p}^{ff}(S')$ are not effective because
  - $T_{p}^{ff}$ is not finitary
  - both $S$ and $S'$ are infinite sets
- the analysis and verification of properties of finite failure, can be based on effective approximations of the operator $T_{p}^{ff}$
- since we have two semantics and two specifications, we can use two different (related) abstractions
  - an upward approximation (of the least fixpoint semantics)
  - a downward approximation (of $T_{p}^{ff} \downarrow \omega$)
The depth-$k$ domain

- we define the function $\text{depth}$ on terms, atoms and goal of a program.

\[
|t| = \begin{cases} 
1 & \text{if } t \text{ is a constant or a variable} \\
\max\{|t_1|, \ldots, |t_n|\} + 1 & \text{if } t = f(t_1, \ldots, t_n)
\end{cases}
\]

The downward approximation

- $< \alpha^{bl}, \gamma^{bl} >$ is a reversed Galois insertion, i.e.,

\[
\alpha^{bl}(\cap X_i) = \cap (\alpha^{bl}(X_i)).
\]

- We can define the optimal abstract fixpoint operator $T^{\text{frbl}}_p$ on $D^{bl}$.

Example

\[
p
\]
\[
q(a) : \neg p(X)
\]
\[
p(f(X)) : \neg p(X)
\]

for $k = 3$,

\[
lfp(T^{\text{frbl}}_p) = \{q(f(X)), q(f(a)), q(f(X)), q(f(a)), p(a), p(f(a)), p(f(f(a))))\}
\]

\[
\gamma^{bl}(lfp(T^{\text{frbl}}_p)) \subseteq lfp(T^{\text{fr}}_p)
\]
The upward approximation

- $< \alpha_{up}, \gamma_{up} >$ is a Galois insertion.

- We can define the optimal abstract fixpoint operator $T_p^{ff_{up}}$ on $D_{up}$.

**Example**

$$
p
q(a) : -p(X)
p(f(X)) : -p(X)
$$

for $k = 3$,

$$
\text{lfp}(T_p^{ff_{up}}) = \{ q(f(f(K))), q(f(f(X)))[X/f(X)], q(f(f(a))), q(f(X)), q(f(a)),
p(a), p(f(a)), p(f(f(a))), p(f(f(K))) \}
$$

$$
\text{lfp}(T_p^{ff}) \subseteq \gamma_{up}(\text{lfp}(T_p^{ff_{up}}))
$$
depth – k correctness and sufficient conditions

• the two abstractions are used to get finite approximations of the Non-Ground Finite Failure set and of the complement of the success set.

• the specifications

  – $S_{\alpha_{up}}$ is the $\alpha_{up}$ abstraction of the intended Non-Ground Finite Failure set.

  – $S'_{\alpha_{bl}}$ is the $\alpha_{bl}$ abstraction of the intended set of atoms which either finitely fail or (universally) do not terminate.

    * the complement of the set of atoms (of depth $\leq k$) which have a successful derivation.

• a program $P$ is **depth – k correct** if

  \[ c_1 \alpha_{up}(lfp(T_{P}^{ff})) \subseteq S_{\alpha_{up}}. \]

  \[ c_2 S'_{\alpha_{bl}} \subseteq \alpha_{bl}(T_{P}^{ff} \downarrow \omega). \]

• sufficient conditions for the depth – k correctness

  \[ sc_1 T_{P}^{ff}_{up}(S_{\alpha_{up}}) \subseteq S_{\alpha_{up}}. \]

  \[ sc_2 S'_{\alpha_{bl}} \subseteq T_{P}^{ff}_{bl}(S'_{\alpha_{bl}}). \]
Examples I

• Example 1

\( P_1 : \) append([], X, X) : \text{not } \text{list}([X]) \quad \text{instead of } \text{append}([], X, X) : \text{not } \text{list}(X)

append([X|Y], Z, T) : \text{not } \text{append}(Y, Z, [X|T]).

\text{list}([ ]).

\text{list}([X|Y]) : \text{not } \text{list}(Y).

– The program is not correct w.r.t. the intended depth-k success set.

We can detect this error.

\text{append}([], a, a) \in S'_{\alpha^u} \quad \text{yet} \quad \text{append}([], a, a) \notin T^{\text{finf}}_{P_1} (S'_{\alpha^u}).

Therefore \textbf{sC}2 does not hold.

• Example 2

\( P_2 : \) append([X|Y], Z, T) : \text{not } \text{append}(Y, Z, [X|T]).

\text{list}([ ]).

\text{list}([X|Y]) : \text{not } \text{list}(Y).

– The program is not correct w.r.t. the intended depth-k finite failure set.

We can detect this error.

\text{append}([], [a], [a]) \in T^{\text{finf}}_{P_2} (S_{\alpha^u}), \quad \text{yet}

\text{append}([], [a], [a]) \notin S_{\alpha^u}.

Therefore \textbf{sC}1 does not hold.
Examples II

• Example 3

\[ P_3: \ append([], X, X) : \neg \text{list}(X). \]
\[ \quad append([X|Y], Z, T) : \neg append(Y, Z, [X|T]). \]
\[ \quad \text{list}([]). \]
\[ \quad \text{list}([X|Y]) : \neg \text{list}(Y). \]

- **sc₁** holds.
  \[ \Downarrow \]
  The program is correct w.r.t. the intended depth-\(k\) finite failure set.

- **sc₂** holds.
  \[ \Downarrow \]
  The program is correct w.r.t. the intended depth-\(k\) successful set.
Future Work

- how to extend the approach to other abstract domains which might be useful for reasoning about finite failure (e.g. assertions).