Passing Sub-Query Bindings with MS

\[ p(X,Y) \leftarrow q(X,Z), r(Y,Z), s(X,W), t(W,Y). \]

\{ (X) : p(X, 1) \} ?

\{ (Y) : p(2, Y) \} ?

\{ (X,Y) : p(X, Y) \} ?

\{ ( ) : p(2, 1) \} ?
For a given adornment there usually is at least one "good" left-right ordering of the body literals in the corresponding rule.

Good means that the input bindings of the query adornments can be propagated in a favorable manner (wrt efficiency, i.e. from the perspective of logical optimization) to the switch literals. This propagation happens sideways to subsequent literals in the left-right sequence of body literals. Thus, we speak of a "sideways information passing strategy" (short: SIPS).

The reordering of literals needed for achieving a "favorable" order of evaluation (wrt the overall efficiency of answer computation) corresponds to the determination of a "good" join order during logical optimization in SQL systems.

Efficient SIPS can be found by analyzing the data flow in a rule body, e.g.:
When choosing a „good“ SIPS, the following heuristics turns out to be helpful: **Always try to bind as many parameters as possible!**

The number of parameters bound when choosing a particular ordering of literals can be represented by using *adornments*, too – in the above example:

- \( p^{bf} \leftarrow q^{bf}, r^{fb}, t^{fb}, s^{bb} \)
- \( p^{bf} \leftarrow q^{bf}, r^{fb}, s^{bf}, t^{bb} \)
- \( p^{bf} \leftarrow q^{bf}, s^{fb}, r^{fb}, t^{bb} \)
- \( p^{bf} \leftarrow q^{bf}, s^{fb}, t^{bf}, r^{bb} \)
- \( p^{bf} \leftarrow s^{bf}, t^{bf}, q^{bf}, r^{bb} \)
- \( p^{bf} \leftarrow s^{bf}, q^{bf}, r^{fb}, t^{bb} \)
- \( p^{bf} \leftarrow s^{bf}, q^{bf}, t^{bf}, r^{bb} \)
- \( p^{bf} \leftarrow s^{bf}, t^{bf}, q^{bf}, r^{bb} \)

Each of these 8 SIPS is **optimal wrt to dataflow**, because each time the newly occurring variable bindings are transferred to follow-up literals in an optimal way, reusing them for further restrictions during evaluation immediately.
If considering additional criteria (apart from just dataflow), an even better efficiency may be possibly achieved
(Similar idea: Cost models in "classical" relational DB-techniques).

Such additional criteria could be: Size of the resp. relations, existence of indexes (in case of base relations, i.e., data structures speeding up access), selectivity of the resp. join attributes etc.
The Magic Sets Transformation (rule compilation phase) proceeds in the general case as follows:

Given a rule in the format $L \leftarrow L_1, \ldots, L_n$ defining a relation $p$ and an adornment $\alpha$ for $p$.

1. Determine a SIP-strategy for $\alpha$ and reorder the rule body according to this strategy:

   $L \leftarrow L'_1, \ldots, L'_n$.

2. Generate an answer rule for $\alpha$ of the following form:

   \[ \text{answer}_L \leftarrow \text{query}^\alpha \_ L_0, L^*_1, \ldots, L^*_n. \]

   - Here, $L_0$ is obtained from $L$ by omitting all parameters not occurring in positions coded as bound in $\alpha$.
   - The body literals $L^*_i$ ($1 \leq i \leq n$) of the answer rule are defined as follows:

     \[
     L^*_i = \begin{cases} 
     L'_i & \text{if } L'_i \text{ references a base relation} \\
     \text{answer}_L'_i & \text{else}
     \end{cases}
     \]
3. For each rule-defined literal $L'_i$, which has received an adornment $\beta$ in the SIPS determined in step 1., generate a corresponding query rule as follows:

$$\text{query}^\beta_{L''_i} \leftarrow \text{query}^\alpha_{L_0, L_1^*, \ldots, L_{i-1}^*}.$$ 

$L''_i$ is obtained from $L'_i$ by omitting all parameters not bound in $\beta$.

4. Perform these steps for each rule and each (relevant) adornment.
Query evaluation according to the Magic Sets method (fixpoint iteration phase) over a deducitive database $D = (R, F)$ is performed as follows:

Given a magic sets-transformed rule set $R_{\text{magic}}$ and a query $\{ (t_1, \ldots, t_n) : Q \} ?$, the qualification part of which consists of a single literal $Q$ only. Let in addition $\alpha$ denote the adornment of $Q$.

1. Generate a seed fact query $\alpha_Q'$, where $Q'$ is obtained from $Q$ again by omitting all parameters not bound in $\alpha$.

2. Determine via FPI the least fixpoint $F^*$ of the $T_{\text{pos}}$-operator for $R_{\text{magic}}$, properly containing the fact set $F \cup \{ \text{query}^{\alpha}_Q' \}$.

3. Answer the query $\{ (t_1, \ldots, t_n) : \text{answer}_Q \} ?$ over this $F^*$.

4. All answers thus obtained are answers to the given query, too.

(The violation of the standardization assumption by adding the seed fact to the resp. query rules can be „remedied“ by means of those techniques we introduced in Chapter 1.)
Admissible SIPS

- Negative literals 'not L' are evaluated in Datalog according to the "negation as failure" principle, i.e., we try to derive L and "invert" the resulting truth value.

- Negative literals must be variable free (ground literals) and are only used for testing.

- Thus, we have to restrict the choice of a SIPS to such strategies that take care of delaying the evaluation of negative literals at least as long as necessary for making them ground at the time of evaluation: We call such orderings of body literals admissible SIPS!

- As we deal with safe rules only, at least one admissible SIPS exists for each rule.

\[
p(X) \leftarrow s(X,Y), \text{not } r(Y), q(Y).
\]

**safe SIPS:**
- \(p^b \leftarrow s^{bf}, \text{not } r^b, q^b.\)
- \(p^b \leftarrow s^{bf}, q^b, \text{not } r^b.\)
- \(p^b \leftarrow q, s^{bb}, \text{not } r^b.\)
- \(p^b \leftarrow q, \text{not } r^b, s^{bb}.\)

**unsafe SIPS:**
- \(p^b \leftarrow \text{not } r, q, s^{bb}.\)
Magic Sets and Safe SIPS: Example

\[ p(X) \leftarrow s(X,Y), \textbf{not} r(Y), q(Y). \]
\[ r(Y) \leftarrow t_1(X,Y). \]
\[ q(Y) \leftarrow t_2(Y,X). \]

\[ p(a) ? \]
\[ \text{query}_p^b(a). \]

\[ \text{query}_r^b(Y) \leftarrow \text{query}_p^b(X), s(X,Y). \]
\[ \text{query}_q^b(Y) \leftarrow \text{query}_p^b(X), s(X,Y), \textbf{not} \text{answer}_r(Y). \]

**Magic Sets transformation (with admissible SIPS)**

\[ \text{answer}_p(X) \leftarrow \text{query}_p^b(X), s(X,Y), \textbf{not} \text{answer}_r(Y), \text{answer}_q(Y). \]
\[ \text{answer}_r(Y) \leftarrow \text{query}_r^b(Y), t_1(X,Y). \]
\[ \text{answer}_q(Y) \leftarrow \text{query}_q^b(Y), t_2(Y,X). \]

**stratifiable**

\[ p \]
\[ \textbf{not} r \]
\[ q \]
Magic Sets and Negation

- The Magic Sets Transformation (rule compilation phase) can be applied to stratifiable rules with negation without any change, as long as they are based on a safe SIPS.

- For the FPI-Phase of a Magic Sets evaluation one has to employ at least iterated FPI (based on a stratification of $R^{\text{magic}}$), of course.

- For negative literals, Magic Sets generates positive query-literals ("side evaluation" of the "negation as failure" idea), whereas answer-literals remain negated ("as failure"):

\[
p(X) \leftarrow s(X,Y), \textbf{not} r(Y), q(Y).\]

\[
\text{query}_{r^b}(Y) \leftarrow \ldots .
\]

\[
\text{answer}_p(X) \leftarrow \ldots , \textbf{not} \text{answer}_r(Y), \ldots .
\]
In this example, $R_{\text{magic}}$ is stratifiable, too!

$$
\begin{align*}
\text{answer}_p(X) & \leftarrow \text{query}_{p^b}(X), s(X,Y), \text{not} \, \text{answer}_r(Y), \text{answer}_q(Y). \\
\text{answer}_r(Y) & \leftarrow \text{query}_{r^b}(Y), t_1(X,Y). \\
\text{answer}_q(Y) & \leftarrow \text{query}_{q^b}(Y), t_2(Y,X).
\end{align*}
$$

$$
\begin{align*}
\text{query}_{r^b}(Y) & \leftarrow \text{query}_{p^b}(X), s(X,Y). \\
\text{query}_{q^b}(Y) & \leftarrow \text{query}_{p^b}(X), s(X,Y), \text{not} \, \text{answer}_r(Y).
\end{align*}
$$
A stratification is not easy to detect, however!

(or alternatively the canonical stratification of this dependency graph)
The "subordination" of answer\(_r\) and query\(_r\) to \(p\) and \(q\) cannot be derived from just inspecting the dependency graph of the original rule set \(R\).

It is caused by the fact that the r-Literal in the body of the \(p\)-rule stands left of (behind) the q-literal:

\[
p(X) \leftarrow s(X,Y), \text{not} r(Y), q(Y).\]
MS with Negation: Recursive Example (1)

The same example, just modified a little, now with recursive p:

R:

\[ p(X) \leftarrow s(X,Y), \text{not} \ r(Y), p(Y). \]
\[ p(X) \leftarrow q(X). \]
\[ r(Y) \leftarrow t_1(X,Y). \]
\[ q(X) \leftarrow t_2(X,Y). \]

\( p(a) \) ? \( \text{query}_{p^1}(a) \).

R\text{magic}:

\[ \text{answer}_{p}(X) \leftarrow \text{query}_{p^b}(X), s(X,Y), \text{not} \ \text{answer}_{r}(Y), \text{answer}_{p}(Y). \]
\[ \text{answer}_{p}(X) \leftarrow \text{query}_{p^b}(X), \text{answer}_{q}(X). \]
\[ \text{answer}_{r}(Y) \leftarrow \text{query}_{r^b}(Y), t_1(X,Y). \]
\[ \text{answer}_{q}(X) \leftarrow \text{query}_{q^b}(X), t_2(X,Y). \]

\[ \text{query}_{r^b}(Y) \leftarrow \text{query}_{p^b}(X), s(X,Y). \]
\[ \text{query}_{p^b}(Y) \leftarrow \text{query}_{p^b}(X), s(X,Y), \text{not} \ \text{answer}_{r}(Y). \]
\[ \text{query}_{q^b}(X) \leftarrow \text{query}_{p^b}(X), \text{not} \]
MS with Negation: Recursive Example (1)

\[ R^{\text{magic}}: \]

\[
\begin{align*}
\text{answer}_p(X) & \leftarrow \text{query}_{p^b}(X), s(X,Y), \textbf{not} \ \text{answer}_r(Y), \ \text{answer}_p(Y). \\
\text{answer}_p(X) & \leftarrow \text{query}_{p^b}(X), \ \text{answer}_q(X). \\
\text{answer}_r(Y) & \leftarrow \text{query}_{r^b}(Y), \ t_1(X,Y). \\
\text{answer}_q(X) & \leftarrow \text{query}_{q^b}(X), \ t_2(X,Y). \\
\text{query}_{r^b}(Y) & \leftarrow \text{query}_{p^b}(X), s(X,Y). \\
\text{query}_{p^b}(Y) & \leftarrow \text{query}_{p^b}(X), s(X,Y), \textbf{not} \ \text{answer}_r(Y). \\
\text{query}_{q^b}(X) & \leftarrow \text{query}_{p^b}(X),
\end{align*}
\]
The Magic Sets Transformation "destroys" stratifiability in this case! $R^{\text{magic}}$ is unstratifiable (even though $R$ was stratifiable)!

**Reason:** The recursive $p$-literal is placed in the SIPS behind the negative $r$-literal and thus depends on $\text{answer}_r$ negatively within the query-rule.

$$p(X) \leftarrow s(X,Y), \text{not} r(Y), p(Y).$$

$$\text{query}_p^b(Y) \leftarrow \text{query}_p^b(X), s(X,Y), \text{not} \text{answer}_r(Y).$$
Recursive Example Revised (1)

Maybe exchanging both literals helps?

R:
\[
\begin{align*}
p(X) & \leftarrow s(X,Y), \ p(Y), \ \textbf{not} \ r(Y). \\
p(X) & \leftarrow q(X). \\
r(Y) & \leftarrow t_1(X,Y). \\
q(X) & \leftarrow t_2(X,Y). 
\end{align*}
\]

Still stratifiable

Magic Sets transformation (still with admissible SIPS)

R\text{magic}:
\[
\begin{align*}
\text{answer}_p(X) & \leftarrow \text{query}_p^b(X), \ s(X,Y), \ \text{answer}_p(Y), \ \textbf{not} \ \text{answer}_r(Y). \\
\text{answer}_p(X) & \leftarrow \text{query}_p^b(X), \ \text{answer}_q(X). \\
\text{answer}_r(Y) & \leftarrow \text{query}_r^b(Y), \ t_1(X,Y). \\
\text{answer}_q(X) & \leftarrow \text{query}_q^b(X), \ t_2(X,Y). \\
\text{query}_r^b(Y) & \leftarrow \text{query}_p^b(X), \ s(X,Y), \ \text{answer}_p(Y). \\
\text{query}_p^b(Y) & \leftarrow \text{query}_p^b(X), \ s(X,Y), \ \textbf{not} \ \text{answer}_r(Y). \\
\text{query}_q^b(X) & \leftarrow \text{query}_p^b(X), \\
\end{align*}
\]

Maybe exchanging both literals helps?
Recursive Example Revised (2)

\begin{align*}
\text{answer}_p(X) & \leftarrow \text{query}_p^b(X), s(X,Y), \text{answer}_p(Y), \textbf{not} \text{answer}_r(Y). \\
\text{answer}_p(X) & \leftarrow \text{query}_p^b(X), \text{answer}_q(X). \\
\text{answer}_r(Y) & \leftarrow \text{query}_r^b(Y), t_1(X,Y). \\
\text{answer}_q(X) & \leftarrow \text{query}_q^b(X), t_2(X,Y). \\
\end{align*}
Even though one of the not-edges disappears and the old cycle is "broken, still . . .

. . . the new positive edge closes a different not-cycle!

Exchanging does not help: $R^{\text{magic}}$ remains unstratifiable!

\[
\text{query}_r^b(Y) \leftarrow \ldots, \text{answer}_p(Y).
\]

\[
\text{answer}_p(X) \leftarrow \ldots, \text{not} \text{ answer}_r(Y).
\]
Magic Sets and Negation: Summary

- Even in case the input set of rules $R$ is stratifiable it is **not** guaranteed that the Magic Sets Transformation preserves this property: $R_{\text{Magic}}$ may always **become** unstratifiable!

- In such cases, iterated FPI is no longer sufficient, but the (much more demanding) CFP-Method has to be applied in order to materialize query- and answer-facts.

- **But:** If using the "naive" evaluation mode by fully materializing $F^*$ using the original (untransformed) rule set $R$ and selecting answer facts at the end, **no** problems with determining truth values of answers and intermediate results can be observed!

- **Thus:** It can be proved that CFP-materialization of unstratifiable rules obtained via a Magic Sets Transformation from stratifiable rules always produces defined truth values:
  
  **Undefined facts do never occur!**

- **Conclusion:** Violations of stratifiability introduced by the MS-Transformation are always "benign" (if presence of undefined facts is called "malicious")!
is_ok(T) ←
  has Been Tested(T).

is_ok(T) ←
  is_complex_part(T), not has_bad_component(T).

has_bad_component(T) ←
  component_of(T, T'), not is_ok(T').

is_complex_part(T) ←
  component_of(T, _).

component_of:

\[
\begin{array}{c}
p_2 \\
p_5 \\
p_6
\end{array}
\quad\quad
\begin{array}{c}
p_1 \quad p_3 \\
p_4
\end{array}
\quad\quad
\begin{array}{c}
p_7 \\
p_9 \\
p_{10}
\end{array}
\]
Now even the original rule set is unstratifiable, but still its evaluation over the given fact \( F \) remains two-valued (no undefined facts):

\[
F^+ = \{ \text{is\_ok}(p_2), \text{is\_ok}(p_8), \text{has\_bad\_component}(p_1), \text{is\_ok}(p_4), \text{is\_ok}(p_9), \text{has\_bad\_component}(p_2), \text{is\_ok}(p_7), \text{is\_ok}(p_{10}), \text{is\_complex\_part}(p_1), \text{is\_complex\_part}(p_2), \text{is\_complex\_part}(p_4), \text{is\_complex\_part}(p_8) \}
\]

\[
F^- = H_D - F^+
\]

Component of:

\( F \)

\( p_2 \)

\( p_5 \) \hspace{1cm} \( p_6 \)

\( p_1 \)

\( p_3 \)

\( p_4 \)

\( p_7 \)

\( p_8 \)

\( p_9 \)

\( p_{10} \)

\( p_1 \) tested (thus ok)

\( p_2 \) ok recursively

\( p_4 \) bad component

\( p_7 \)

\( p_8 \)

\( p_9 \)

\( p_{10} \)

\( \text{is\_ok}(p_1) \) ? \hspace{1cm} \text{No !}

(Part \( p_3 \) has not been tested and doesn’t have components, all of which are ok.)
Such an unstratifiable but (at least over "acyclic" facts) "benign" rule set can be treated via Magic Sets, too:

\[
\begin{align*}
is\_ok(T) & \leftarrow \text{has\_been\_tested}(T). \\
is\_ok(T) & \leftarrow \text{is\_complex\_part}(T), \text{not} \ \text{has\_bad\_component}(T). \\
\text{has\_bad\_component}(T) & \leftarrow \text{component\_of}(T, T'), \text{not} \ \text{is\_ok}(T'). \\
is\_complex\_part(T) & \leftarrow \text{component\_of}(T, _).
\end{align*}
\]

Such an unstratifiable but (at least over "acyclic" facts) "benign" rule set can be treated via Magic Sets, too:

\[
\begin{align*}
\text{answer\_ok}(T) & \leftarrow \text{query\_ok}^b(T), \text{tested}(T). \\
\text{answer\_ok}(T) & \leftarrow \text{query\_ok}^b(T), \text{answer\_complex}(T), \text{not} \ \text{answer\_bad\_comp}(T). \\
\text{answer\_bad\_comp}(T) & \leftarrow \text{query\_bad\_comp}^b(T), \text{comp}(T, T'), \text{not} \ \text{answer\_ok}(T'). \\
\text{answer\_complex}(T) & \leftarrow \text{query\_complex}^b(T), \text{comp}(T, _).
\end{align*}
\]

\[
\begin{align*}
\text{query\_complex}^b(T) & \leftarrow \text{query\_ok}^b(T). \\
\text{query\_bad\_comp}^b(T) & \leftarrow \text{query\_ok}^b(T), \text{answer\_complex}(T). \\
\text{query\_ok}^b(T') & \leftarrow \text{query\_bad\_comp}^b(T), \text{comp}(T, T').
\end{align*}
\]
MS for Unstratifiable Rules (4)

answer_ok(T) ← query_okb(T), tested(T).
answer_ok(T) ← query_okb(T), answer_complex(T), not answer_bad_comp(T).
answer_bad_comp(T) ← query_bad_compb(T), comp(T, T'), not answer_ok(T').
answer_complex(T) ← query_complexb(T), comp(T, _).

query_complexb(T) ← query_okb(T).
query_bad_compb(T) ← query_okb(T), answer_complex(T).
query_okb(T') ← query_bad_compb(T), comp(T, T').
In this example, the "uncritical" lower part of the dependency graph can be evaluated separately via positive FPI. On top of the fact set derived this way, the non-stratifiable upper part can be treated by CFP afterwards.

Such a mixture of FPIs might be a good idea in general, not just in context of MS!
MS for Unstratifiable Rules (6)

Stratifiable lower part:

\[
\text{answer\_complex}(T) \leftarrow \text{query\_complex}^b(T), \text{comp}(T, \_).
\]

\[
\text{query\_complex}^b(T) \leftarrow \text{query\_ok}^b(T).
\]

\[
\text{query\_bad\_comp}^b(T) \leftarrow \text{query\_ok}^b(T), \text{answer\_complex}(T).
\]

\[
\text{query\_ok}^b(T') \leftarrow \text{query\_bad\_comp}^b(T), \text{comp}(T, T').
\]

\[
\begin{align*}
\text{comp}(p_1, p_2). & \quad \text{tested}(p_2). \\
\text{comp}(p_1, p_3). & \quad \text{tested}(p_7). \\
\text{comp}(p_1, p_4). & \quad \text{tested}(p_9). \\
\text{comp}(p_2, p_5). & \quad \text{tested}(p_{10}). \\
\text{comp}(p_2, p_6). & \quad \text{query\_ok}^b(p_1). \\
\text{comp}(p_4, p_7). & \quad \text{query\_ok}^b(p_1). \\
\text{comp}(p_4, p_8). & \quad \text{query\_ok}^b(p_2). \\
\text{comp}(p_8, p_9). & \quad \text{query\_ok}^b(p_3). \\
\text{comp}(p_8, p_{10}). & \quad \text{query\_ok}^b(p_4).
\end{align*}
\]
### MS for Unstratifiable Rules (7)

| comp(p₁, p₂).  | tested(p₂).     |
| comp(p₁, p₃).  | tested(p₇).     |
| comp(p₁, p₄).  | tested(p₉).     |
| comp(p₂, p₅).  | tested(p₁₀).    |
| comp(p₂, p₆).  |                |
| comp(p₄, p₇).  | query_okᵇ(p₁). |
| comp(p₄, p₈).  |                |
| comp(p₈, p₉).  |                |
| comp(p₈, p₁₀). |                |

```plaintext
query_complexᵇ(p₁).
answer_complex(p₁)
query_bad_compᵇ(p₁)
query_okᵇ(p₂)
query_okᵇ(p₃)
query_окᵇ(p₄)
query_bad_compᵇ(p₂)
query_bad_compᵇ(p₄)
answer_complex(p₂)
answer_complex(p₄)
```

---

Recursive descent through the hierarchy of parts in search for "bad components"!
Fixpoint of the bottom layer:

\[
\begin{align*}
\text{comp}(p_1, p_2). & \quad \text{tested}(p_2). \\
\text{comp}(p_1, p_3). & \quad \text{tested}(p_7). \\
\text{comp}(p_1, p_4). & \quad \text{tested}(p_9). \\
\text{comp}(p_2, p_5). & \quad \text{tested}(p_{10}). \\
\text{comp}(p_2, p_6). & \\
\text{comp}(p_4, p_7). & \\
\text{comp}(p_4, p_8). & \\
\text{comp}(p_8, p_9). & \\
\text{comp}(p_8, p_{10}). & 
\end{align*}
\]

Unstratifiable rules from the top layer:

\[
\begin{align*}
\text{answer\_complex}(p_1) & \\
\text{answer\_complex}(p_2) & \\
\text{answer\_complex}(p_4) & \\
\text{query\_bad\_comp}(p_8) & \\
\text{answer\_complex}(p_8) & \\
\text{query\_bad\_comp}(p_1) & \\
\text{query\_bad\_comp}(p_2) & \\
\text{query\_bad\_comp}(p_4) & \\
\text{query\_bad\_comp}(p_8) & \\
\text{query\_ok}(p_1) & \\
\text{query\_ok}(p_2) & \\
\text{query\_ok}(p_3) & \\
\text{query\_ok}(p_4) & \\
\text{query\_ok}(p_5) & \\
\text{query\_ok}(p_6) & \\
\text{query\_ok}(p_7) & \\
\text{query\_ok}(p_8) & \\
\text{query\_ok}(p_9) & \\
\text{query\_ok}(p_{10}) & 
\end{align*}
\]

Unstratifiable rules from the top layer:

\[
\begin{align*}
\text{answer\_ok}(T) & \leftarrow \text{query\_ok}(T), \text{tested}(T). \\
\text{answer\_ok}(T) & \leftarrow \text{query\_ok}(T), \text{answer\_complex}(T), \underline{\text{not} \text{ answer\_bad\_comp}(T)}. \\
\text{answer\_bad\_comp}(T) & \leftarrow \text{query\_bad\_comp}(T), \text{comp}(T, T'), \underline{\text{not} \text{ answer\_ok}(T')}.
\end{align*}
\]

Redundant?!
MS for Unstratifiable Rules (9)

Fixpoint of the lower layer: $F^u$

- $\text{tested}(p_2)$
- $\text{tested}(p_7)$
- $\text{tested}(p_9)$
- $\text{tested}(p_{10})$

Unstratifiable rules of the upper layer:

- $\text{query}_{\text{ok}}(T) \leftarrow \text{query}_{\text{ok}}(T), \text{tested}(T)$.
- $\text{query}_{\text{ok}}(T) \leftarrow \text{query}_{\text{ok}}(T), \text{answer}_{\text{complex}}(T), \text{not} \ \text{answer}_{\text{bad comp}}(T)$.

... 

CFP-Iteration:

- $\text{answer}_{\text{ok}}(p_2)$.
- $\text{answer}_{\text{ok}}(p_7)$.
- $\text{answer}_{\text{ok}}(p_9)$.
- $\text{answer}_{\text{ok}}(p_{10})$.
MS for Unstratifiable Rules (10)

Fixpoint of the lower layer: \( F^u \)

\[
\begin{align*}
\text{comp}(p_1, p_2) \cdot \text{comp}(p_2, p_6) \cdot \\
\text{comp}(p_1, p_3) \cdot \text{comp}(p_4, p_7) \cdot \\
\text{comp}(p_1, p_4) \cdot \text{comp}(p_4, p_8) \cdot \\
\text{comp}(p_2, p_5) \cdot \text{comp}(p_8, p_9) \cdot \\
\text{comp}(p_8, p_{10}).
\end{align*}
\]

query\_bad\_comp^b(p_1)

query\_bad\_comp^b(p_2)

query\_bad\_comp^b(p_4)

query\_bad\_comp^b(p_8)

\[
\begin{align*}
\text{answer\_bad\_comp}(T) & \leftarrow \text{query\_bad\_comp}^b(T), \text{comp}(T, T'), \textbf{not} \text{ answer\_ok}(T').
\end{align*}
\]

CFP-Iteration (cont.):

\[
\begin{align*}
\text{answer\_bad\_comp}(p_1) & \leftarrow \textbf{not} \text{ answer\_ok}(p_2). \\
\text{answer\_bad\_comp}(p_1) & \leftarrow \textbf{not} \text{ answer\_ok}(p_3). \\
\text{answer\_bad\_comp}(p_1) & \leftarrow \textbf{not} \text{ answer\_ok}(p_4). \\
\text{answer\_bad\_comp}(p_2) & \leftarrow \textbf{not} \text{ answer\_ok}(p_5). \\
\text{answer\_bad\_comp}(p_2) & \leftarrow \textbf{not} \text{ answer\_ok}(p_6). \\
\text{answer\_bad\_comp}(p_4) & \leftarrow \textbf{not} \text{ answer\_ok}(p_7). \\
\text{answer\_bad\_comp}(p_4) & \leftarrow \textbf{not} \text{ answer\_ok}(p_8). \\
\text{answer\_bad\_comp}(p_8) & \leftarrow \textbf{not} \text{ answer\_ok}(p_9). \\
\text{answer\_bad\_comp}(p_8) & \leftarrow \textbf{not} \text{ answer\_ok}(p_{10}).
\end{align*}
\]

Fixpoint of the expansion phase
MS for Unstratifiable Rules (11)

CFP-Iteration: Reduction phase

\[
\begin{align*}
\text{answer}_\text{ok}(p_2) &. \\
\text{answer}_\text{ok}(p_7) &. \\
\text{answer}_\text{ok}(p_9) &. \\
\text{answer}_\text{ok}(p_{10}) &. \\
\end{align*}
\]

Subsumption

\[
\begin{align*}
\text{answer}_\text{ok}(p_1) \leftarrow \text{not} \; \text{answer}_\text{bad}_\text{comp}(p_1). \\
\text{answer}_\text{ok}(p_2) \leftarrow \text{not} \; \text{answer}_\text{bad}_\text{comp}(p_2). \\
\text{answer}_\text{ok}(p_4) \leftarrow \text{not} \; \text{answer}_\text{bad}_\text{comp}(p_4). \\
\text{answer}_\text{ok}(p_8) \leftarrow \text{not} \; \text{answer}_\text{bad}_\text{comp}(p_8). \\
\end{align*}
\]

\[
\begin{align*}
\text{answer}_\text{bad}_\text{comp}(p_2) \leftarrow \text{not} \; \text{answer}_\text{ok}(p_2). \\
\text{answer}_\text{bad}_\text{comp}(p_3) \leftarrow \text{not} \; \text{answer}_\text{ok}(p_3). \\
\text{answer}_\text{bad}_\text{comp}(p_4) \leftarrow \text{not} \; \text{answer}_\text{ok}(p_4). \\
\text{answer}_\text{bad}_\text{comp}(p_5) \leftarrow \text{not} \; \text{answer}_\text{ok}(p_5). \\
\text{answer}_\text{bad}_\text{comp}(p_6) \leftarrow \text{not} \; \text{answer}_\text{ok}(p_6). \\
\text{answer}_\text{bad}_\text{comp}(p_7) \leftarrow \text{not} \; \text{answer}_\text{ok}(p_7). \\
\text{answer}_\text{bad}_\text{comp}(p_8) \leftarrow \text{not} \; \text{answer}_\text{ok}(p_8). \\
\text{answer}_\text{bad}_\text{comp}(p_9) \leftarrow \text{not} \; \text{answer}_\text{ok}(p_9). \\
\text{answer}_\text{bad}_\text{comp}(p_{10}) \leftarrow \text{not} \; \text{answer}_\text{ok}(p_{10}). \\
\end{align*}
\]

\[
\{ \}
\]

\text{answer}_\text{bad}_\text{comp}(p_8) \text{ no longer derivable!}
CFP-Iteration: Red**uction phase**(cont.):

- `answer_ok(p_2)`.
- `answer_ok(p_7)`.
- `answer_ok(p_9)`.
- `answer_ok(p_{10})`.
- `answer_ok(p_1) ← not answer_bad_comp(p_1)`.
- `answer_ok(p_4) ← not answer_bad_comp(p_4)`.
- `answer_ok(p_8)`.

\[\text{answer_bad_comp(p_1) ← not answer_ok(p_3).}\]
\[\text{answer_bad_comp(p_1) ← not answer_ok(p_4).}\]
\[\text{answer_bad_comp(p_2) ← not answer_ok(p_5).}\]
\[\text{answer_bad_comp(p_2) ← not answer_ok(p_6).}\]
\[\text{answer_bad_comp(p_4) ← not answer_ok(p_8).}\]

\[\text{answer_bad_comp(p_4) no longer derivable!}\]
MS for Unstratifiable Rules (13)

CFP-Iteration: **Reduction phase** (cont.):

\[
\begin{align*}
\text{answer}_\text{ok}(p_2). &  \\
\text{answer}_\text{ok}(p_7). &  \\
\text{answer}_\text{ok}(p_9). &  \\
\text{answer}_\text{ok}(p_{10}). &  \\
\text{answer}_\text{ok}(p_1) & \leftarrow \text{not} \text{ answer}_\text{bad\_comp}(p_1). \\
\text{answer}_\text{ok}(p_4). &  \\
\text{answer}_\text{ok}(p_8). &  \\
\text{answer}_\text{bad\_comp}(p_1) & \leftarrow \text{not} \text{ answer}_\text{ok}(p_3). \\
\text{answer}_\text{bad\_comp}(p_1) & \leftarrow \text{not} \text{ answer}_\text{ok}(p_4). \\
\text{answer}_\text{bad\_comp}(p_2) & \leftarrow \text{not} \text{ answer}_\text{ok}(p_5). \\
\text{answer}_\text{bad\_comp}(p_2) & \leftarrow \text{not} \text{ answer}_\text{ok}(p_6). \\
\end{align*}
\]
MS for Unstratifiable Rules (14)

CFP-Iteration: Reduction phase (cont.):

\[
\begin{align*}
\text{answer}_\text{ok}(p_2). \\
\text{answer}_\text{ok}(p_7). \\
\text{answer}_\text{ok}(p_9). \\
\text{answer}_\text{ok}(p_{10}). \\
\text{answer}_\text{bad}_\text{comp}(p_1) \quad &\quad \text{not} \quad \text{answer}_\text{bad}_\text{comp}(p_1) \quad \leftarrow \\
\text{answer}_\text{ok}(p_4). \\
\text{answer}_\text{ok}(p_8). \\
\text{answer}_\text{bad}_\text{comp}(p_1) \quad &\quad \text{not} \quad \text{answer}_\text{ok}(p_3). \\
\text{answer}_\text{bad}_\text{comp}(p_2) \quad &\quad \text{not} \quad \text{answer}_\text{ok}(p_5). \\
\text{answer}_\text{bad}_\text{comp}(p_2) \quad &\quad \text{not} \quad \text{answer}_\text{ok}(p_6). \\
\end{align*}
\]
MS for Unstratifiable Rules (15)

CFP-Iteration: Fixpoint of the reduction phase

No!

\[
\text{ok}(p_1) ?
\]

\[
\begin{align*}
\text{answer}_{\text{ok}}(p_2). \quad \text{answer}_{\text{ok}}(p_8). \\
\text{answer}_{\text{ok}}(p_4). \quad \text{answer}_{\text{ok}}(p_9). \\
\text{answer}_{\text{ok}}(p_7). \quad \text{answer}_{\text{ok}}(p_{10}).
\end{align*}
\]

\[
\begin{align*}
\text{answer}_{\text{bad-comp}}(p_1). \\
\text{answer}_{\text{bad-comp}}(p_2).
\end{align*}
\]

Summary:

- Despite unstratifiability of both, the original as well as the MS-transformed rules a complete and correct classification of the parts in form of a two-valued model has been obtained!
- Magic Sets may even succeed in this kind of "unfavorable" constellation, but may be quite costly (because of CFP being unavoidable).
- If the composition of parts would have been cyclic, however, there would have been quite certainly undefined facts!
- Thus, you cannot rely on MS going well for unstratifiable original rules!

CFP may work even for unstratified query answering!
Here is an example (due to Morishita) showing that MS may go wrong, even though the original rules are unstratifiable and have a two-valued semantics:

R:
\[
\begin{align*}
p(X) & \leftarrow t(X,Y,Z), \text{not } p(Y), \text{not } p(Z). \\
p(X) & \leftarrow p_0(X).
\end{align*}
\]

F:
\[
\begin{align*}
t(a,a,b1). & \\
t(b3,c3,b4). \\
t(b1,c1,b2). & \\
t(b4,c4,c5). \\
t(b2,c2, b3). & p_0(c2)
\end{align*}
\]

Facts F graphically represented:

The rule set is unstratifiable, but over the given F $F^*$ is two-valued again:

$F_{+}^* = p(b4)$, $p(b1)$, $p(c2)$

$F_{-}^* = H_D - F_{+}^*$

$t$: p(X) true
\[f\]: p(X) false
MS for Unstratifiable Rules (17)

Magic Sets rules wrt. query \( p(a) \):

\[
\begin{align*}
\text{answer}_p(X) & \leftarrow \text{query}_p^{b}(X), \text{t}(X,Y,Z), \text{not} \ \text{answer}_p(Y), \text{not} \ \text{answer}_p(Z). \\
\text{answer}_p(X) & \leftarrow \text{query}_p^{b}(X), \text{answer}_p(X).
\end{align*}
\]

\[
\begin{align*}
\text{query}_p^{b}(Y) & \leftarrow \text{query}_p^{b}(X), \text{t}(X,Y,Z). \\
\text{query}_p^{b}(Z) & \leftarrow \text{query}_p^{b}(X), \text{t}(X,Y,Z), \text{not} \ \text{answer}_p(Y).
\end{align*}
\]

The CFP of this rule set is three-valued, where all query-facts (except the 'seed') are undefined:

\[
\begin{align*}
\text{F}_+^*: \quad \text{query}_p^{b}(a) \cup F \\
\text{F}_?^*: \\
\text{query}_p^{b}(b1) & \leftarrow \text{not} \ p(a) \\
\text{query}_p^{b}(b2) & \leftarrow \text{not} \ p(a), \text{not} \ p(c1) \\
\text{query}_p^{b}(b3) & \leftarrow \text{not} \ p(a), \text{not} \ p(c1), \text{not} \ p(c2). \\
\text{query}_p^{b}(c3) & \leftarrow \text{not} \ p(a), \text{not} \ p(c1), \text{not} \ p(c2).
\end{align*}
\]
As all query-facts are undefined, all answer-facts remain undefined, too:

Thus, a wrong answer is produced by the Magic Sets rules as p(a) is now undefined instead of being false!

Conclusion: The Magic Sets Method is not always correct if applied to unstratifiable rules
Magic Sets Method with Operators

R:

\[
\begin{align*}
p(X) & \leftarrow s(X,Y), \ Y > Z+1, \ q(X,Z). \\
q(Y) & \leftarrow t_2(Y,X).
\end{align*}
\]

Test literals containing operators are treated like negative literals when determining SIPS: Evaluable only once all parameters have been bound (admissible SIPS)

\[
\text{query}_p^1(a).
\]

The Magic Sets Transformation leaves test literals unchanged.

\[
\begin{align*}
\text{answer}_p(X) & \leftarrow \text{query}_p^1(X), \ s(X,Y), \ \text{answer}_q(X,Z), \ Y > Z+1. \\
\text{answer}_q(Y) & \leftarrow \text{query}_q^1(X), \ t_2(Y,X).
\end{align*}
\]

\[
\text{query}_q^1(X) \leftarrow \text{query}_p^1(X), \ s(X,Y).
\]

R^\text{magic}: