Beyond Stratification

CS240B Notes

Notes based on Section 10.4 of *Advanced Database Systems*—Morgan Kaufmann, 1997

C. Zaniolo, April 2002

Introduction

- Many proposals—e.g., **locally stratified programs** and **well founded models**–on how to go beyond stratified negation
- Objective is to approach the power of stable model, without its exponential complexity.
- also the semantic well-formedness of the program can be determined from the rules (independent of the DB) as for stratified programs
- XY-stratification: is a particular class of locally stratified programs for which we also have a simple compile-time check, and an efficient implementation
- In fact, XY-stratified programs are particular Datalog$_{\mathbf{1s}}$ programs.

Stratification by the Temporal Argument

Ancestors of marc and their generation gap expressed using the differential fixpoint:

$$
\begin{array}{ll}
r_1 : \mathtt{delta\_anc}(0, \mathtt{marc}). & \\
r_2 : \mathtt{delta\_anc}(\mathtt{J}+1, \mathtt{Y}) \leftarrow & \mathtt{delta\_anc}(\mathtt{J}, \mathtt{X}), \mathtt{parent}(\mathtt{Y}, \mathtt{X}), \\
 & \neg \mathtt{all\_anc}(\mathtt{J}, \mathtt{Y}). \\
r_3 : \mathtt{all\_anc}(\mathtt{J}+1, \mathtt{X}) \leftarrow & \mathtt{all\_anc}(\mathtt{J}, \mathtt{X}). \\
r_4 : \mathtt{all\_anc}(\mathtt{J}, \mathtt{X}) \leftarrow & \mathtt{delta\_anc}(\mathtt{J}, \mathtt{X}).
\end{array}
$$

Stratification by the temporal Argument: cont.

1. *This program is locally stratified by the first argument in* `anc` *that serves as temporal argument.*
2. *The zero stratum consists of atoms of nonrecursive predicates such as* `parent` *and of atoms that unify with* $\mathtt{all\_anc}(\mathtt{0}, \mathtt{X})$ *or* $\mathtt{delta\_anc}(\mathtt{0}, \mathtt{X})$*, where* $\mathtt{X}$ *can be any constant in the universe*
3. *The* k^{th} *stratum contains atoms* $\mathtt{all\_anc}(\mathtt{k}, \mathtt{X})$ *and* $\mathtt{delta\_anc}(\mathtt{k}, \mathtt{X})$*.*
   Thus, this program is locally stratified, since the heads of recursive rules belong to strata that are one above those of their goals.

X-rules and Y-rules

- r is an X-rule when the temporal argument (TA) in every recursive predicate in r is the same variable (e.g., J),
- r is a Y-rule when for some variable J
  1. the head of r has $J+1$ as its TA
  2. some goal of r has as TA J, and
  3. the remaining recursive goals have either J or $J+1$ as their TAs.

XY-programs: Let P be a set of rules defining mutually recursive predicates. Then we say that P is an XY-program when:

1. Every recursive predicate of P has a distinguished temporal argument.
2. Every recursive rule r is either an X-rule or a Y-rule.

The Old and the New

Given an XY -program P, its *bi-state program*, P_{bis}, is computed as follows: For each $r \in P$,

1. *Rename all the recursive predicates in* `r` *that have the same temporal argument as the head of* r *with the distinguished prefix* `new_`*.*
2. *Rename all other occurrences of recursive predicates in* `r` *with the distinguished prefix* `old_`*.*
3. *Drop the temporal arguments from the recursive predicates.*

$$
\begin{array}{lll}
\texttt{new\_delta\_anc}(\texttt{marc}). & & \\
\texttt{new\_delta\_anc}(\texttt{Y}) \leftarrow & & \texttt{old\_delta\_anc}(\texttt{X}), \texttt{parent}(\texttt{Y}, \texttt{X}), \\
& & \neg \texttt{old\_all\_anc}(\texttt{Y}). \\
\texttt{new\_all\_anc}(\texttt{X}) \leftarrow & & \texttt{new\_delta\_anc}(\texttt{X}). \\
\texttt{new\_all\_anc}(\texttt{X}) \leftarrow & & \texttt{old\_all\_anc}(\texttt{X}).
\end{array}
$$

XY-stratificatied Programs

Definition: Let P be an XY-program. P is said to be XY-stratified when P_{bis} is a stratified program.

Our previous program is stratified with the following strata:

$$\begin{aligned} \mathrm{S}_0 &= \{\texttt{parent}, \texttt{old\_all\_anc}, \texttt{old\_delta\_anc}\}, \\ \mathrm{S}_1 &= \{\texttt{new\_delta\_anc}\} \\ \mathrm{S}_2 &= \{\texttt{new\_all\_anc}\} \end{aligned}$$

Theorem: Let P be an XY-stratified program. Then P has a unique stable model.

Computing the stable model of an XY-stratified program P

Inititialize: *Set* $T = 0$ *and insert the fact* $\mathtt{counter}(\mathtt{T})$.

Forever repeat the following two steps:

1. *Apply the iterated fixpoint computation to the synchronized program* P_{bis}, *and for each recursive predicate* $\mathtt{q}$, *compute* $\mathtt{new\_q}$. *Return the* $\mathtt{new\_q}$ *atoms so computed, after adding a temporal argument* $\mathtt{T}$ *to these atoms; the value of* $\mathtt{T}$ *is taken from* $\mathtt{counter}(\mathtt{T})$.
2. *For each recursive predicate* $\mathtt{q}$, *replace* $\mathtt{old\_q}$ *with* $\mathtt{new\_q}$, *computed in the previous step. Then, replace* $\mathtt{counter}(\mathtt{T})$ *with* $\mathtt{counter}(\mathtt{T}+1)$.

- Copy rules
- When does the computation stop?

Classical Algorithms can be Expressed as XY-stratified programs

A simple example: Coalescing after Temporal Projection.

$$\texttt{emp\_dep\_sal}(1001, \texttt{shoe}, 35000, 19920101, 19940101).$$
$$\texttt{emp\_dep\_sal}(1001, \texttt{shoe}, 36500, 19940101, 19960101).$$

These two tuples must be merged.

Merging overlapping periods after a temporal projection

$$\mathtt{e\_hist}(0, \mathtt{Eno}, \mathtt{Frm}, \mathtt{To}) \leftarrow \quad \mathtt{emp\_dep\_sal}(0, \mathtt{Eno}, \mathtt{D}, \mathtt{S}, \mathtt{Frm}, \mathtt{To}).$$

$$\begin{aligned} \mathtt{overlap}(\mathtt{J}+1, \mathtt{Eno}, \mathtt{Frm1}, \mathtt{To1}, \mathtt{Frm2}, \mathtt{To2}) \leftarrow & \\ & \mathtt{e\_hist}(\mathtt{J}, \mathtt{Eno}, \mathtt{Frm1}, \mathtt{To1}), \\ & \mathtt{e\_hist}(\mathtt{J}, \mathtt{Eno}, \mathtt{Frm2}, \mathtt{To2}), \\ & \mathtt{Frm1} \leq \mathtt{Frm2}, \mathtt{Frm2} \leq \mathtt{To1}, \\ & \mathtt{distinct}(\mathtt{Frm1}, \mathtt{To1}, \mathtt{Frm2}, \mathtt{To2}). \end{aligned}$$

$$\begin{aligned} \mathtt{e\_hist}(\mathtt{J}, \mathtt{Eno}, \mathtt{Frm1}, \mathtt{To}) \leftarrow \quad & \mathtt{overlap}(\mathtt{J}, \mathtt{Eno}, \mathtt{Frm1}, \mathtt{To1}, \mathtt{Frm2}, \mathtt{To2} \\ & \mathtt{select\_larger}(\mathtt{To1}, \mathtt{To2}, \mathtt{To}). \end{aligned}$$

$$\begin{aligned} \mathtt{e\_hist}(\mathtt{J}+1, \mathtt{Eno}, \mathtt{Frm}, \mathtt{To}) \leftarrow \quad & \mathtt{e\_hist}(\mathtt{J}, \mathtt{Eno}, \mathtt{Frm}, \mathtt{To}), \\ & \mathtt{overlap}(\mathtt{J}+1, \_, \_, \_, \_, \_), \\ & \neg\mathtt{overlap}(\mathtt{J}+1, \mathtt{Eno}, \mathtt{Frm}, \mathtt{To}, \_, \_), \\ & \neg\mathtt{overlap}(\mathtt{J}+1, \mathtt{Eno}, \_, \_, \mathtt{Frm}, \mathtt{To}). \end{aligned}$$

$$\begin{aligned} \mathtt{final\_e\_hist}(\mathtt{Eno}, \mathtt{Frm}, \mathtt{To}) \leftarrow \quad & \mathtt{e\_hist}(\mathtt{J}, \mathtt{Eno}, \mathtt{Frm}, \mathtt{To}), \\ & \neg\mathtt{e\_hist}(\mathtt{J}+1, \_, \_, \_). \end{aligned}$$

Temporal Projection–auxiliary predicates

$$
\begin{aligned}
\mathtt{distinct}(\mathtt{Frm1}, \mathtt{To1}, \mathtt{Frm2}, \mathtt{To2}) &\leftarrow \quad \mathtt{To1} \neq \mathtt{To2}. \\
\mathtt{distinct}(\mathtt{Frm1}, \mathtt{To1}, \mathtt{Frm2}, \mathtt{To2}) &\leftarrow \quad \mathtt{Frm1} \neq \mathtt{Frm2}. \\
\mathtt{select\_larger}(\mathtt{X}, \mathtt{Y}, \mathtt{X}) &\leftarrow \quad \mathtt{X} \geq \mathtt{Y}. \\
\mathtt{select\_larger}(\mathtt{X}, \mathtt{Y}, \mathtt{Y}) &\leftarrow \quad \mathtt{Y} > \mathtt{X}.
\end{aligned}
$$