Computing Logic Programming Semantics in Linear Algebra

Hien D. Nguyen, University of Information Technology (UIT), VNU-HCM, Vietnam

Chiaki Sakama, Wakayama Univ., Japan

Taisuke Sato, AIST, Japan

Katsumi Inoue, NII, Japan

MIWAI 2018@Hanoi

11/28/2018

Content

- Introduction
- Computing least model of a definite program
- Computing stable model of a normal program
- Experimental results
- Conclusion

Content

- Introduction
- Computing least model of a definite program
- Computing stable model of a normal program
- Experimental results
- Conclusion

Introduction

- **Logic programming** is a type of programming paradigm which is largely based on formal logic.
- Provides languages for declarative problem solving and symbolic reasoning.

- **Linear algebra** is at the core of many applications of scientific computation.
- One of challenging topic in AI is integrating linear algebraic computation and symbolic computation.

Purpose

- ❑ Refine the framework of (Sakama et. al. 2017) and present algorithms for finding the least model of a definite program and stable models of a normal program.
- ❑ Based on the structure of matrices representing logic programs, research some optimization techniques for speeding-up these algorithms.
- ❑ Evaluate the complexity of proposed algorithms.
- ❑ Testing and comparing these methods.

Sakama, C., Inoue, K., Sato, T.: *Linear Algebraic Characterization of Logic Programs*, In: Proc. of KSEM 2017, LNAI 10412, pp.530-533, Springer, Melbourne, Australia (2017)

Content

- Introduction
- Computing least model of a definite program
- Computing stable model of a normal program
- Experimental results
- Conclusion

Vector Representation of Interpretations

- Given the Herbrand base $B_P = \{ p, q, r, s \}$, an interpretation $I = \{ p, r \}$ is represented by the vector:

$$\boldsymbol{v} = \begin{array}{cc} \left(\begin{matrix}\mathbf{1}\\ \mathbf{0}\\ \mathbf{1}\\ \mathbf{0}\end{matrix}\right) & \begin{matrix} \boldsymbol{p}\\ \boldsymbol{q}\\ \boldsymbol{r}\\ \boldsymbol{s}\end{matrix}\end{array}$$

- The i-th element of $\boldsymbol{v}$ represents the truth value of p_i (written $row_1(\mathbf{v})= p_,$ $row_2(\mathbf{v})= q_,$ $row_3(\mathbf{v})= r_,$ etc).

- Given $\boldsymbol{v} =(a_1 ,\ldots,a_n)^{\mathrm{T}} \in \mathbf{R}^{\mathrm{n}}$, $\boldsymbol{v}[a_1\ldots a_k]$ represents a (sub)vector $(a_1 ,\ldots,a_{\mathrm{k}})^{\mathrm{T}} \in \mathbf{R}^{\mathrm{k}}$ $(\mathrm{k} \le \mathrm{n})$.

Matrix Representation of Definite Programs

- $P = \{\ p \leftarrow q, \quad q \leftarrow p \wedge r, \quad r \leftarrow s, \quad s \leftarrow\ \}$ is represented by $\boldsymbol{M}_{\mathrm{P}} \in \mathbf{R}^{4\times 4}$:

$$\begin{array}{ccccccl}
 & & \multicolumn{4}{c}{\textbf{body}} & \\
 & & p & q & r & s & \\
 & p & 0 & 1 & 0 & 0 & \rightarrow p \leftarrow q \\
\textbf{head} & q & \frac{1}{2} & 0 & \frac{1}{2} & 0 & \rightarrow q \leftarrow p \wedge r \\
 & r & 0 & 0 & 0 & 1 & \rightarrow r \leftarrow s \\
 & s & 0 & 0 & 0 & 1 & \rightarrow s \leftarrow
\end{array}$$

! Fact $(s \leftarrow)$ is encoded as $(s \leftarrow s)$.

- The i-th row represents the atom p_i in the head, and the j-th column represents the atom p_j in the body of a rule (written: $row_1(\boldsymbol{M}_{\mathrm{P}}) = p$, $col_2(\boldsymbol{M}_{\mathrm{P}}) = q$, … etc)

Matrix Representation of Rules with the Same Head

- $P = \{\, p \leftarrow q,\ \ q \leftarrow p \wedge r,\ \ q \leftarrow s,\ \ s \leftarrow \,\}$ is transformed to the program $P^{\delta} = Q \cup D$ where:

  $Q = \{\, p \leftarrow q,\ \ t \leftarrow p \wedge r,\ \ u \leftarrow s,\ \ s \leftarrow \,\}$ and $D = \{\, q \leftarrow t \vee u \,\}$.

- P^{δ} is represented by $\boldsymbol{M}_{\mathrm{P}^{\delta}} \in \mathbf{R}^{6\times 6}$:

$$\begin{array}{c|cccccc} & p & q & r & s & t & u \\ \hline p & 0 & 1 & 0 & 0 & 0 & 0 \\ q & 0 & 0 & 0 & 0 & 1 & 1 \\ r & 0 & 0 & 0 & 0 & 0 & 0 \\ s & 0 & 0 & 0 & 1 & 0 & 0 \\ t & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 \\ u & 0 & 0 & 0 & 1 & 0 & 0 \end{array}$$

- Rules in D are called **d-rules**.

- Note: $q \leftarrow t \vee u$ is a shorthand of $q \leftarrow t$ and $q \leftarrow u$,

so P^{δ} is considered a definite program.

Computing Least Models

- Given $P = \{\, p \leftarrow q,\ \ q \leftarrow p \wedge r,\ \ r \leftarrow s,\ \ s \leftarrow \,\}$, the **initial vector** $\boldsymbol{v}_0 = (0,0,0,1)^{\mathrm{T}}$ represents facts in *P*. Then,

$$\boldsymbol{M}_{\mathrm{P}}\boldsymbol{v}_0 = \begin{array}{c|cccc} & p & q & r & s \\ \hline p & 0 & 1 & 0 & 0 \\ q & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ r & 0 & 0 & 0 & 1 \\ s & 0 & 0 & 0 & 1 \end{array} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \qquad \boldsymbol{v}_1 = \boldsymbol{\theta}(\boldsymbol{M}_{\mathrm{P}}\boldsymbol{v}_0)$$

$$\boldsymbol{M}_{\mathrm{P}}\boldsymbol{v}_1 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{1}{2} \\ 1 \\ 1 \end{pmatrix} \qquad \boldsymbol{v}_2 = \boldsymbol{\theta}(\boldsymbol{M}_{\mathrm{P}}\boldsymbol{v}_1) = \boldsymbol{v}_1$$

- $\boldsymbol{v}_1$ is a fixpoint of $\boldsymbol{v}_k = \boldsymbol{\theta}(\boldsymbol{M}_{\mathrm{P}}\boldsymbol{v}_{k-1})$ $(k \geq 1)$.
- $\boldsymbol{v}_1 = (0,0,1,1)^{\mathrm{T}}$ represents the least model $\{\, r, s \,\}$ of *P*.

Column Reduction

- Consider $P^{\delta} = Q \cup D$ where $Q = \{\, p \leftarrow q,\ \ t \leftarrow p \wedge r,\ \ u \leftarrow s,\ \ s \leftarrow \,\}$ and $D = \{\, q \leftarrow t \vee u \,\}$.
- Reduce columns for newly introduced atoms and produce $\boldsymbol{N}_{\mathrm{P}^{\delta}} \in \mathbf{R}^{6\times 4}$:

$$\boldsymbol{M}_{\mathrm{P}^{\delta}} = \begin{array}{c|cccccc} & p & q & r & s & t & u \\ \hline p & 0 & 1 & 0 & 0 & 0 & 0 \\ q & 0 & 0 & 0 & 0 & 1 & 1 \\ r & 0 & 0 & 0 & 0 & 0 & 0 \\ s & 0 & 0 & 0 & 1 & 0 & 0 \\ t & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 \\ u & 0 & 0 & 0 & 1 & 0 & 0 \end{array}$$

➡

$$\boldsymbol{N}_{\mathrm{P}^{\delta}} = \begin{array}{c|cccccc} & p & q & r & s & t & u \\ \hline p & 0 & 1 & 0 & 0 & 0 & 0 \\ q & 0 & 0 & 0 & 0 & 1 & 1 \\ r & 0 & 0 & 0 & 0 & 0 & 0 \\ s & 0 & 0 & 0 & 1 & 0 & 0 \\ t & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 \\ u & 0 & 0 & 0 & 1 & 0 & 0 \end{array}$$

Example (cont.)

- $P^{\delta} = \{\ p \leftarrow q,\quad t \leftarrow p \wedge r,\quad u \leftarrow s,\quad s \leftarrow,\quad q \leftarrow t \vee u\ \}.$
- Given $\boldsymbol{v} = (0,0,0,1)^{\mathrm{T}}$, it becomes $\boldsymbol{w} = \boldsymbol{N}_{\mathrm{P}^{\delta}}\,\boldsymbol{v} = (0,0,0,1,0,1)^{\mathrm{T}}$.
- Introduce the rule: *if an element in the body of a d-rule is 1, then the element in the head of the d-rule is set to 1.*

$$\boldsymbol{N}_{\mathrm{P}^{\delta}} = \begin{array}{cccc|c} p & q & r & s & \\ \hline 0 & 1 & 0 & 0 & p \\ 0 & 0 & 0 & 0 & q \\ 0 & 0 & 0 & 0 & r \\ 0 & 0 & 0 & 1 & s \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & t \\ 0 & 0 & 0 & 1 & u \end{array}$$

Add this rule to the θ-theresholding (written θ_{D}).

- Put $d = (q \leftarrow t \vee u)$. Since $row_6(\boldsymbol{w}) = u \in body(d)$ and $head(d) = q$, applying θ_{D} to $\boldsymbol{N}_{\mathrm{P}^{\delta}}\,\boldsymbol{v}$ produces $\theta_{\mathrm{D}}\,(\boldsymbol{N}_{\mathrm{P}^{\delta}}\,\boldsymbol{v}) = (0,1,0,1,0,1)^{\mathrm{T}}$.

Computing Least Models

- $P^{\delta} = \{\ p \leftarrow q,\quad t \leftarrow p \wedge r,\quad u \leftarrow s,\quad s \leftarrow,\quad q \leftarrow t \vee u\ \}.$
- Given $\boldsymbol{v}_0 = (0,0,0,1,0,0)^{\mathrm{T}}$, $\boldsymbol{v}_0[1\ldots4] = (0,0,0,1)^{\mathrm{T}}$:

$$N_{\mathrm{P}^{\delta}} = \begin{array}{c|cccc} & p & q & r & s \\ \hline p & 0 & 1 & 0 & 0 \\ q & 0 & 0 & 0 & 0 \\ r & 0 & 0 & 0 & 0 \\ s & 0 & 0 & 0 & 1 \\ t & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ u & 0 & 0 & 0 & 1 \end{array}$$

$\boldsymbol{v}_1 = \theta_{\mathrm{D}}\,(\boldsymbol{N}_{\mathrm{P}^{\delta}}\,\boldsymbol{v}_0[1\ldots4]) = (0,1,0,1,0,1)^{\mathrm{T}}$

$\boldsymbol{v}_2 = \theta_{\mathrm{D}}\,(\boldsymbol{N}_{\mathrm{P}^{\delta}}\,\boldsymbol{v}_1[1\ldots4]) = (1,1,0,1,0,1)^{\mathrm{T}}$

$\boldsymbol{v}_3 = \theta_{\mathrm{D}}\,(\boldsymbol{N}_{\mathrm{P}^{\delta}}\,\boldsymbol{v}_2[1\ldots4]) = (1,1,0,1,0,1)^{\mathrm{T}} = \boldsymbol{v}_2$

- Then $\boldsymbol{v}_2$ represents the least model of P^{δ} and $\boldsymbol{v}_2[1\ldots4] = (1,1,0,1)$ is a vector representing the least model $\{\ p, q, s\ \}$ of P.

Theorem 2.3: Let P be a definite program with $B_P = \{p_1,\ldots,p_n\}$, and P^δ a transformed d-program with $B_{P^\delta} = \{p_1,\ldots,p_n, p_{n+1},\ldots,p_m\}$.

Let $\boldsymbol{N}_{\mathrm{P}^\delta} \in \mathbf{R}^{m\times n}$ be a submatrix of P^δ. Given a vector $v \in \mathbf{R}^n$ representing an interpretation I of P, let $u = \theta_D(\boldsymbol{N}_{\mathrm{P}^\delta} v) \in \mathbf{R}^m$.

Then u is a vector representing an interpretation J of P^δ such that:

$$J \cap B_{P^\delta} = T_P(I).$$

Complexities

- In matrix computation, complexity of computing $\boldsymbol{M}_{P^\delta}\,\boldsymbol{v}$ is $O(m^2)$ and computing θ(.) is $O(m)$. The number of times for iterating $\boldsymbol{M}_{P^\delta}\,\boldsymbol{v}$ is at most $(m+1)$ times. So the complexity of fixpoint computation is $O((m+1)\times(m+m^2))= O(m^3)$.
- In column reduction, the complexity of computing $\boldsymbol{N}_{P^\delta}\,\boldsymbol{v}$ is $O(m\times n)$ and computing $\theta_D(.)$ is $O(m\times n)$. The number of times for iterating $\boldsymbol{N}_{P^\delta}\,\boldsymbol{v}$ is at most $(m+1)$ times. So the complexity of fixpoint computation is:

$$O((m+1)\times(m\times n+m\times n))= O(m^2\times n).$$

- Column reduction reduces complexity as $m \geqslant n$ in general.

Content

- Introduction
- Computing least model of a definite program
- Computing stable model of a normal program
- Experimental results
- Conclusion

Matrix Representation of Normal Program

- $P = \{\, p \leftarrow q \wedge \neg r \wedge s,\ \ q \leftarrow \neg t \wedge q,\ \ q \leftarrow s,\ \ r \leftarrow \neg t,\ \ s \leftarrow,\ \ t \leftarrow \}$

- $P^{+} = \{p \leftarrow q \wedge \overline{r} \wedge s,\ q \leftarrow \overline{t} \wedge q,\ q \leftarrow s,\ r \leftarrow \overline{t},\ s \leftarrow,\ t \leftarrow\}$
- $P^{\delta} = Q \cup D$

where $Q = \{p \leftarrow q \wedge \overline{r} \wedge s,\ q_1 \leftarrow \overline{t} \wedge q,$

$q_2 \leftarrow s,\ r \leftarrow \overline{t},\ s \leftarrow,\ t \leftarrow\}$

and $D = \{q \leftarrow q_1 \vee q_2\}$

$$\boldsymbol{M}_{P^{\delta}} \in \mathbf{R}^{9\times 9}$$

$$M_{P'} = \begin{array}{c} \begin{array}{ccccccccc} p & q & r & s & t & q_1 & q_2 & \overline{r} & \overline{t} \end{array} \\ \left(\begin{array}{ccccccccc} 0 & 1/3 & 0 & 1/3 & 0 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & ½ & 0 & 0 & 0 & 0 & 0 & 0 & ½ \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right) \end{array} \begin{array}{c} \\ p \\ q \\ r \\ s \\ t \\ q_1 \\ q_2 \\ \overline{r} \\ \overline{t} \end{array}$$

Initial matrix

Initial matrix $M_o \in \mathbf{R}^{m\times h}$ $(1 \le h \le 2^{m-n})$:

- Each row of M_o corresponds to each element of $B_{\underline{P^\delta}}$ in a way that row$_i(M_o) = p_i$ for $1 \le i \le n$ and row$_i(M_o) = q_i$ for $n+1 \le i \le m$.
- $a_{ij} = 1$ $(1 \le i \le n,\ 1 \le j \le h)$ iff a fact $p_i \leftarrow$ is in P; otherwise, $a_{ij} = 0$.
- $a_{ij} = 0$ $(n+1 \le i \le m,\ 1 \le j \le h)$ iff a fact $q_i \leftarrow$ is in P; otherwise, there are two possibilities 0 and 1 for a_{ij}, so it is either 0 or 1.
- $P^\delta = Q \cup D$

where $Q = \{p \leftarrow q \wedge \bar{r} \wedge s,\ q_1 \leftarrow \bar{t} \wedge q,$

$q_2 \leftarrow s,\ r \leftarrow \bar{t},\ s \leftarrow,\ t \leftarrow\}$

and $D = \{q \leftarrow q_1 \vee q_2\}$

$$M_0 = \begin{array}{c} p \\ q \\ r \\ s \\ t \\ q_1 \\ q_2 \\ \bar{r} \\ \bar{t} \end{array} \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$$

$\boldsymbol{M_0 \in \mathbf{R}^{9\times 2}}$

Computing stable models

- $P = \{\, p \leftarrow q \wedge \neg r \wedge s,\ \ q \leftarrow \neg t \wedge q,\ \ q \leftarrow s,\ \ r \leftarrow \neg t,\ \ s \leftarrow,\ t \leftarrow \}$.
  Then,

$$M_P\delta = \begin{array}{c}
\begin{array}{ccccccccc} p & q & r & s & t & q_1 & q_2 & \overline{r} & \overline{t} \end{array} \\
\left(\begin{array}{ccccccccc}
0 & 1/3 & 0 & 1/3 & 0 & 0 & 0 & 1/3 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & ½ & 0 & 0 & 0 & 0 & 0 & 0 & ½ \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}\right)
\end{array}
\begin{array}{c} \\ p \\ q \\ r \\ s \\ t \\ q_1 \\ q_2 \\ \overline{r} \\ \overline{t} \end{array}
\qquad
M_0 = \begin{array}{c} p \\ q \\ r \\ s \\ t \\ q_1 \\ q_2 \\ \overline{r} \\ \overline{t} \end{array}
\begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}
\qquad
M_1 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}
\qquad
M_2 = \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}
\qquad
M_3 = \begin{pmatrix} 0 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$$

$M_1 = \theta(M_P\delta.M_0)$ $\quad M_2 = \theta(M_P\delta.M_1)$ $\quad M_3 = \theta(M_P\delta.M_2) = M_2$

- $\boldsymbol{M}_3$ is a fixpoint of $\boldsymbol{M}_k = \boldsymbol{\theta}(\boldsymbol{M}_P\boldsymbol{\delta}.\boldsymbol{M}_{k-1})$ $(k \geq 1)$.
- $\mathbf{v}_2 = (1,1,0,1,1,0,1,1,0)^T$ represents the set A=$\{p, q, s, t, q_2, \overline{r}\}$ and
  A $\cap$ $B_P = \{p, q, s, t\}$ is the stable model of P.

Column Reduction

- Consider $P^{\delta} = Q \cup D$ with representation matrix $M_{P^{\delta}} \in \mathbf{R}^{9\times 9}$
- Reduce columns for newly introduced atoms and produce $N_{P^{\delta}} \in \mathbf{R}^{9\times 7}$:

$$M_{P^{\delta}} = \begin{array}{ccccccccc|c} p & q & r & s & t & q_1 & q_2 & \bar{r} & \bar{t} & \\ \hline 0 & 1/3 & 0 & 1/3 & 0 & 0 & 0 & 1/3 & 0 & p \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & q \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & r \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & s \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & t \\ 0 & \tfrac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & \tfrac{1}{2} & q_1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & q_2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \bar{r} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \bar{t} \end{array} \Rightarrow N_{P^{\delta}} = \begin{array}{ccccccc|c} p & q & r & s & t & \bar{r} & \bar{t} & \\ \hline 0 & 1/3 & 0 & 1/3 & 0 & 1/3 & 0 & p \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & q \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & r \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & s \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & t \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & \bar{r} \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & \bar{t} \\ 0 & \tfrac{1}{2} & 0 & 0 & 0 & 0 & \tfrac{1}{2} & q_1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & q_2 \end{array}$$

$$P = \{\, p \leftarrow q \wedge \neg r \wedge s,\ \ q \leftarrow \neg t \wedge q,\ \ q \leftarrow s,\ \ r \leftarrow \neg t,\ \ s \leftarrow,\, t \leftarrow \}$$

- $v_1 \in \mathbf{R}^5$ represents the facts in P, $v_1 = (0\ 0\ 0\ 1\ 1)^T$
- $A = \{(0\ \ 0)^T, (1\ \ 0)^T, (0\ \ 1)^T, (1\ \ 1)^T\}$ with card(A) = $2^2 = 4$
- $B = \{(0\ \ 1)^T, (1\ \ 1)^T\}$

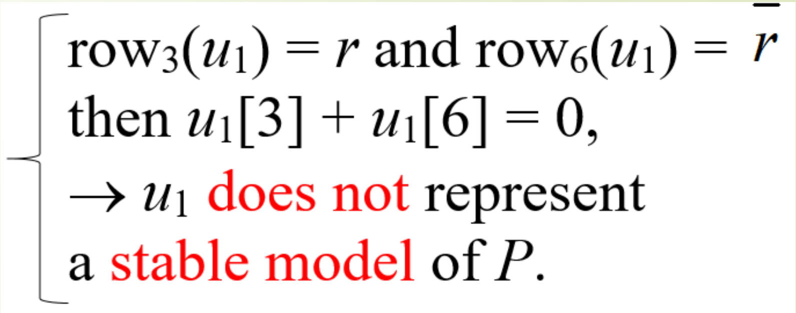
$v_2 \in A$ ¥ $B = \{(0\ \ 0)^T, (1\ \ 0)^T\}$

$V = \{(v_1\ \ v_2)^T | \ v_2 \in A$ ¥ $B\} = \{(0\ 0\ 0\ 1\ 1\ 0\ 0)^T, (0\ 0\ 0\ 1\ 1\ 1\ 0)^T\}$

(i) For $u_o = (0\ 0\ 0\ 1\ 1\ 0\ 0)^T$:

$$u_1 = \theta_D(N_P, u_o) = (0\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 1)^T$$

$$u_2 = \theta_D(N_P, u_1[1 \ldots 7]) = (0\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 1)^T = u_1.$$



$\text{row}_3(u_1) = r$ and $\text{row}_6(u_1) = \bar{r}$
then $u_1[3] + u_1[6] = 0$,
$\rightarrow u_1$ does not represent
a stable model of P.

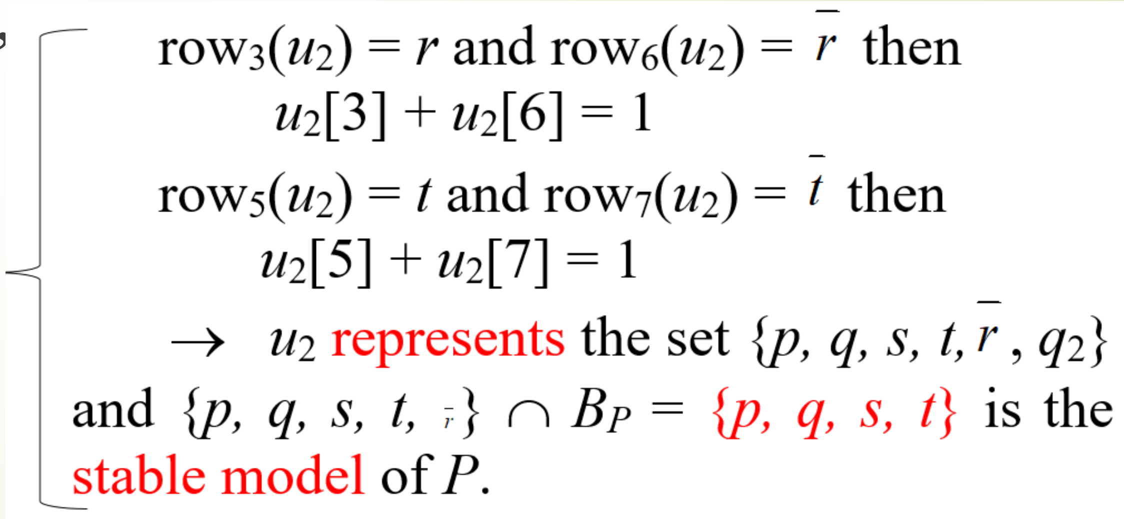
(ii) For $u_o = (0\ 0\ 0\ 1\ 1\ 1\ 0)^T$:

$u_1 = \theta_D(N_P, u_o) = (0\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1)^T$,

$u_2 = \theta_D(N_P, u_1[1\ldots7]) = (1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1)^T$,

$u_3 = \theta_D(N_P, u_2[1\ldots7]) = (1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 1)^T$

$= u_2$



$$\left\{\begin{array}{l} \text{row}_3(u_2) = r \text{ and } \text{row}_6(u_2) = \bar{r} \text{ then} \\ \quad u_2[3] + u_2[6] = 1 \\ \text{row}_5(u_2) = t \text{ and } \text{row}_7(u_2) = \bar{t} \text{ then} \\ \quad u_2[5] + u_2[7] = 1 \end{array}\right.$$

→ u_2 represents the set $\{p, q, s, t, \bar{r}, q_2\}$ and $\{p, q, s, t, \bar{r}\} \cap B_P = \{p, q, s, t\}$ is the stable model of P.

Complexities

- The complexity of $M_{P'}M$ is $O(m^2 \times h)$. The number of times for iterating $M_{P^\delta}.M$ is at most $(\boldsymbol{m} + 1)$ times. Thus, the complexity of computing stable models is $O((m + 1) \times m^2 \times h) = \mathbf{O(m^3 \times h)}$.
- In column reduction, the complexity of computing $N_{P^\delta}.u_o[1 \ldots n']$ is $O(m \times r)$ and computing $\theta_D(.)$ is $O(m \times r)$. Since the number of times for iterating $N_{P'}u_o[1 \ldots r]$ is at most $(m + 1)$ times and $|V| = h$, the complexity of computing stable models is:

$$O((m + 1) \times (m \times r + m \times r) \times h) = \mathbf{O(m^2 \times r \times h)}:$$

- Column reduction reduces complexity as $m \geqslant r$ in general.

Content

- Introduction
- Computing least model of a definite program
- Computing stable model of a normal program
- Experimental results
- Conclusion

Experiments

- Compare 3 algorithms for computing:
  - fixpoint by the T_P-operator (van Emden & Kowalski, 1976)
  - matrix computation
  - column reduction
- Testing is done on a machine with the configuration:
  - OS: Linux Ubuntu 16.04 LTS 64bit
  - CPU: Intel Core i7-6800K (3.4GHz/14nm/Cores=6/ Threads=12 /Cache15MB), Memory 32GB, DDR-2400
  - GPU: GeForce GTX1070TI GDDR5 8GB
  - Implementation Language: Maple 2017, 64bit

Parameters

- Runtime is measured by changing the parameters:
  - n: size of the Herbrand base B_P
  - m: number of rules in P
- Based on (n,m), randomly generate a program having rules as follows:

N	0	1	2	3	4	5	6	7	8
rate	< n/3	4%	4%	10%	40%	35%	4%	2%	∼1%

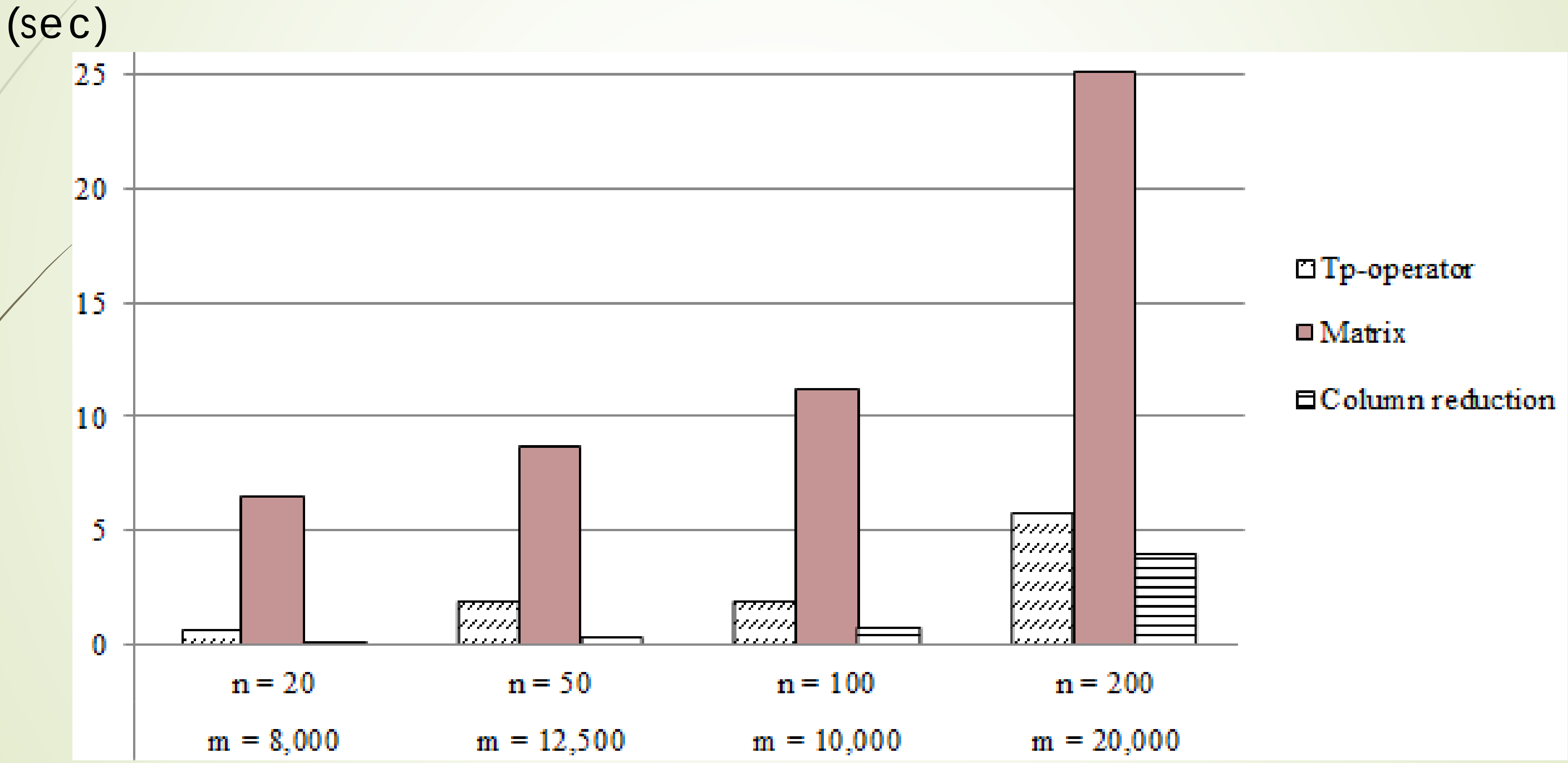
+ N is the number of atoms in the body of a rule
+ Every program has > 95% rules with $|\text{body}(r)| > 1$

Results of testing on definite programs

n	m	T_p	Matrix Fixpoint/All	Column Reduce Fixpoint/All
20	400	0.07	0.225 / 0.238	0.019 / 0.034
20	8,000	0.628	6.491 / 6.709	0.103 / 0.251
50	2,500	0.499	3.797/ 3.925	0.114/ 0.205
50	12,500	1.952	8.709/ 9.023	0.377/ 0.812
100	5,000	2.056	13.23 / 13.326	0.661 / 0.978
100	10,000	1.935	11.166 / 11.479	0.79 / 1.27
200	400	0.037	0.059 / 0.073	0.012 / 0.06
200	20,000	5.846	25.093 / 25.945	3.973 / 6.73

+ “All” means time for creating a program matrix + computing the fixpoint.

Comparison (fixpoint computation)



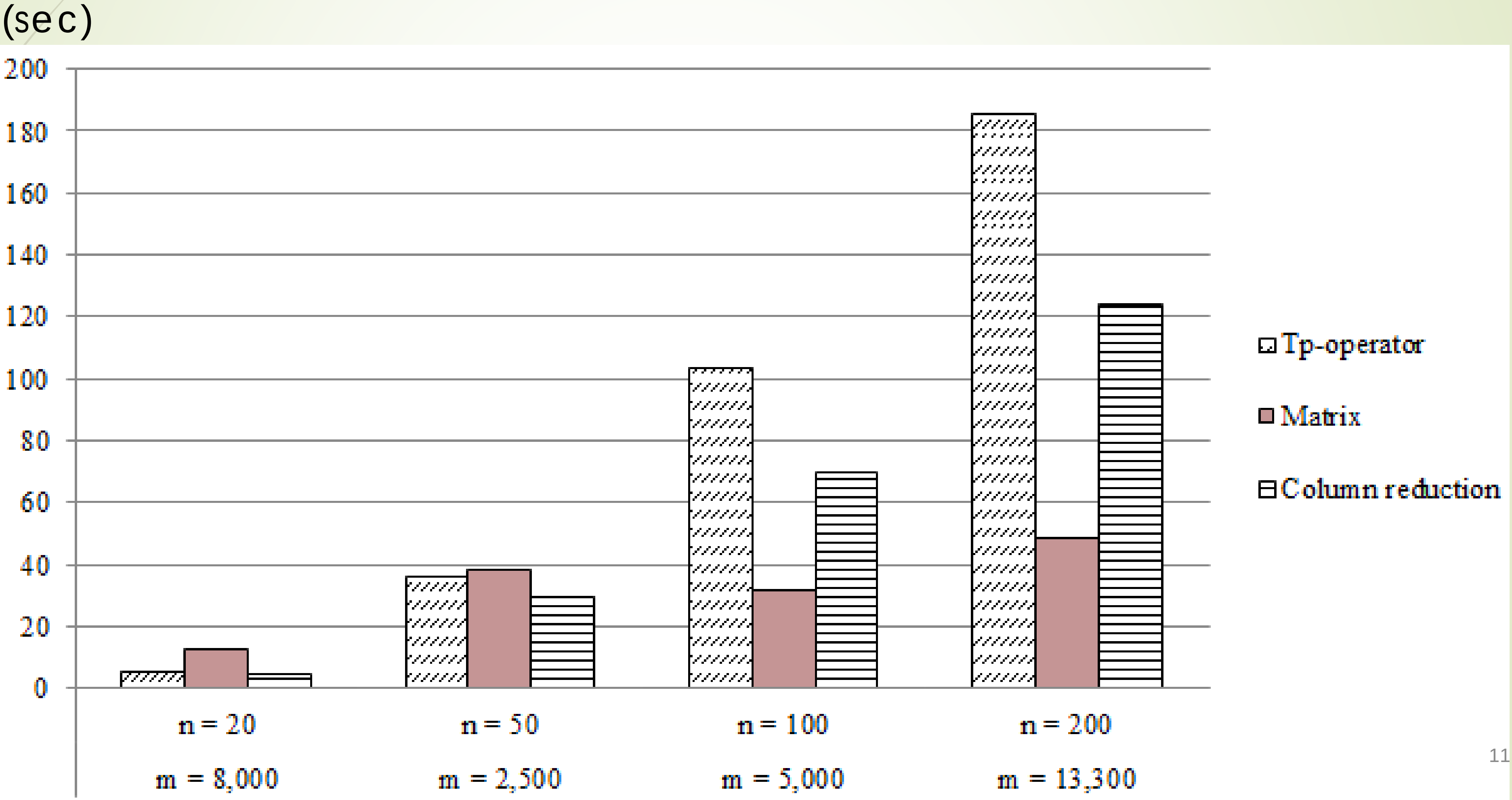
Results of testing on normal programs

n	m	k	T_p	Matrix Fixpoint/All	Column Red Fixpoint/All
20	400	8	2.432	19.603 / 19.714	3.338 / 3.362
20	8,000	6	5.531	12.368 / 12.696	4.502 / 4.603
50	2,500	8	36.574	37.863 / 38.463	29.582 / 29.786
50	12,500	7	49.485	30.819 / 32.00	48.883 / 49.32
100	5,000	8	103.586	31.68 / 32.338	69.579 / 69.851
100	10,000	8	264.547	84.899 / 87.142	192.981 / 194.003
200	400	6	0.429	1.928 / 2.021	1.222 / 1.342
200	13,300	6	185.778	48.185 / 49.185	124.119 / 126.255

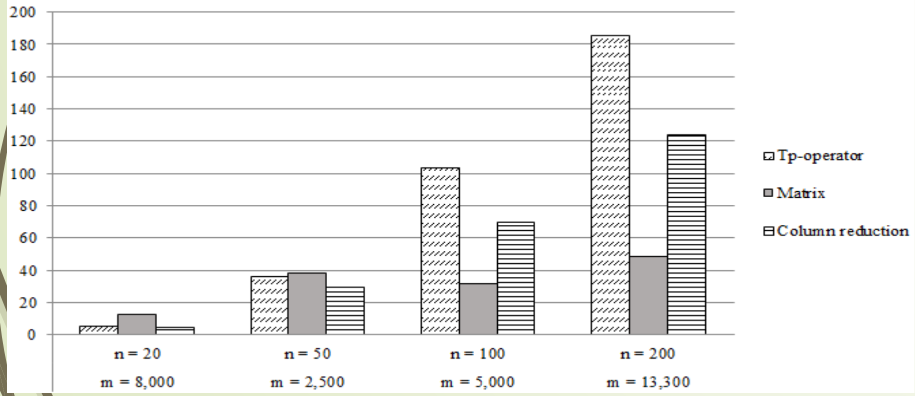
+ *k* is the number of negative literals in a program *P*.

+ “All” means time for creating a program matrix + computing the fixpoint.

Comparison (fixpoint computation)



- Matrix computation is effective when the size of *n* is large (*n* =100 or 200).
- Computation by column reduction is faster than computation by the T_P-operator, while it is slower than the naive method in case of *n* = 100 or 200.
- To see the effect of computation by column reduction, we would need further environment that realizes efficient computation of matrices.



Content

- Introduction
- Computing least model of a definite program
- Computing stable model of a normal program
- Experimental results
- Conclusion

Conclusion

- ❑ Develop new algorithms for computing logic programming semantics in linear algebra and the improvement methods for speeding-up those algorithms.
- ❑ Results of testing show that:
  - ❖ The computation by column reduction is fastest in computing least models.
  - ❖ The naive matrix computation on a d-program is often better than column reduction in computing stable models.

The next work

- **Computating the stable models of a normal program:**
  - Although the size of the program matrix and the initial matrix are large, they have many zero elements (sparse matrix).

→ Improve the method for representing matrices in sparse forms which also brings storage advantages with a good matrix library.

- Combine partial evaluation to reduce runtime (Sakama et. al. 2018).

Chiaki Sakama, Hien D. Nguyen, Taisuke Sato, Katsumi Inoue: *Partial Evaluation of Logic Programs in Vector Space*, 11th Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP 2018), Oxford, UK, July 2018.