Probabilistic Graphical Models Specified by Probabilistic Logic Programs: Semantics and Complexity

Fabio G. Cozman and Denis D. Mauá

Universidade de São Paulo, Brazil

September 8, 2016

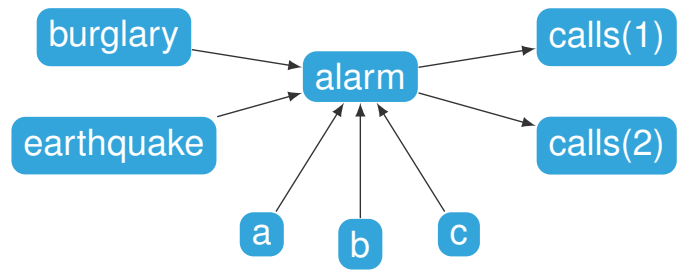
Many languages to specify complex PGMs

- Tabular with pre-defined functions (e.g. noisy-or)
- Markov Logic
- BUGS
- Church
- Factorie
- PRMs
- RBNs
- . . .
- Probabilistic Logic Programming: e.g. ProbLog
  - Intuitive semantics (distributional semantics)
  - Declarative approach
  - Efficient inference (by model counting)

ProbLog

Logic program with some facts annotated with probabilities

0.7 :: burglary . 0.2 :: earthquake .
0.9 :: a . 0.8 :: b . 0.1 :: c .
neighbor(1) . neighbor(2) .
alarm $\leftarrow$ burglary, earthquake, a .
alarm $\leftarrow$ burglary, **not** earthquake, b .
alarm $\leftarrow$ **not** burglary, earthquake, c .
calls(X) $\leftarrow$ alarm, neighbor(X) .



This work

What is the complexity of marginal inference in PGMs specified by logic programs:

- Acyclic programs specify Bayesian networks
- Stratified Cyclic programs specify chain graphs

Normal Logic Programs

Finite set of normal rules:

$$A_0 \leftarrow A_1, \ldots, A_m, \text{not } A_{m+1}, \ldots, \text{not } A_n$$

- each A_i is an atom $r(X_1, \ldots, X_{|r|})$
- A_0 is the head
- $A_1, \ldots, A_n$ is the body

- Propositional program contains no logical variables
- Definite program contains no negated literals (not)
- No functions here

HERBRAND BASE

Set of all propositional atoms obtained by substituting, for every relation in the program, variables with constants

E.g.: $\{\mathsf{alarm}, \mathsf{neighbor}(1), \mathsf{neighbor}(2), \mathsf{earthquake}, \dots\}$

INTERPRETATION

Truth-value assignment to ground atoms; represented as a subset of the Herbrand base (understood as the set of true atoms)

E.g.: $\mathsf{alarm} \mapsto \mathsf{true}, \mathsf{neighbor}(1) \mapsto \mathsf{fale}, \mathsf{neighbor}(2) \mapsto \mathsf{true}, \dots$

PROBABILISTIC LOGIC PROGRAM (PLP)

- normal logic program
- probabilistic facts: $p :: A$, rational $p \in [0, 1]$
- probabilistic facts do not unify with heads of rules

$$0.7 :: \mathsf{burglary}\,. \quad 0.2 :: \mathsf{earthquake}\,.$$
$$0.9 :: \mathsf{a}\,. \quad 0.8 :: \mathsf{b}\,. \quad 0.1 :: \mathsf{c}\,.$$
$$\mathsf{neighbor}(1)\,. \quad \mathsf{neighbor}(2)\,.$$
$$\mathsf{alarm} \leftarrow \mathsf{burglary}, \mathsf{earthquake}, \mathsf{a}\,.$$
$$\mathsf{alarm} \leftarrow \mathsf{burglary}, \mathbf{not}\ \mathsf{earthquake}, \mathsf{b}\,.$$
$$\mathsf{alarm} \leftarrow \mathbf{not}\ \mathsf{burglary}, \mathsf{earthquake}, \mathsf{c}\,.$$
$$\mathsf{calls}(X) \leftarrow \mathsf{alarm}, \mathsf{neighbor}(X)\,.$$

SEMANTICS

- Total choice: truth-value assignment to probabilistic facts
- Probabilistic facts assumed stochastically independent
- Probability of total choice C (set of true atoms):

$$\Pr(\mathbf{C}) = \prod_{A \in \mathbf{C}} \Pr(A) \prod_{A \in \mathbf{PF} \setminus \mathbf{C}} [1 - \Pr(A)]$$

  where $\Pr(A) = p$ such that $p :: A$.

- For each total choice, the resulting (non-probabilistic) logic program has a unique well-founded model (which is also the stable model, minimal model)
  - Thus probability distribution over models is well-defined

0.7 :: a.

c $\leftarrow$ a.

0.4 :: b.

d $\leftarrow$ **not** b.

a	b	model	$\Pr(\text{model})$
true	true	$\{\text{c}\}$	$0.7 \cdot 0.4 = 0.28$
true	false	$\{\text{c}, \text{d}\}$	$0.7(1 - 0.4) = 0.42$
false	true	$\{\}$	$(1 - 0.7)0.4 = 0.12$
false	false	$\{\text{d}\}$	$(1 - 0.7)(1 - 0.4) = 0.18$

$$\Pr(c = \text{true}) = \sum_{\text{model} \models \text{c}} \Pr(\text{model}) = 0.28 + 0.42 = 0.7$$

Dependency

A depends on B if there is a rule where A is in the head and B in the body; if B appears negated (in at least one such rule) then dependency is negative (otherwise is positive)

Dependency graph

Directed graph where each relation is a node; each edge is a dependency relation annotated with its sign

Acyclic program

Dependency graph is acyclic

Stratified program

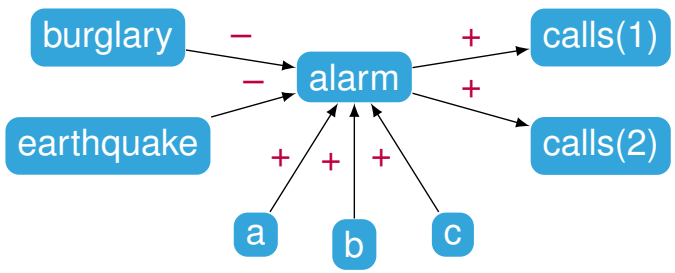
Dependency graph does not contain negative cycles

SOURCES OF COMPLEXITY

- Inferential/combined complexity: input is a PLP and truth-assignment $\mathbf{Q}$ to some of the ground atoms of the Herbrand base, output is $\Pr(\mathbf{Q})$
  - useful when program is large
- Query complexity: fix a PLP; input is truth-assignment $\mathbf{Q}$ to some of the ground atoms of the Herbrand base, output is $\Pr(\mathbf{Q})$
  - useful when program is small, query is large
- Domain complexity: fix a PLP and a truth-assignment $\mathbf{Q}$, input is a domain size (either in unary or in binary), output is $\Pr(\mathbf{Q})$
  - useful when investigating sensitivity of query to domain size

DEPENDENCY GRAPH OF ACYCLIC PROGRAM

0.7 :: burglary . 0.2 :: earthquake .
0.9 :: a . 0.8 :: b . 0.1 :: c .
neighbor(1) . neighbor(2) .
alarm $\leftarrow$ burglary, earthquake, a .
alarm $\leftarrow$ burglary, **not** earthquake, b .
alarm $\leftarrow$ **not** burglary, earthquake, c .
calls(X) $\leftarrow$ alarm, neighbor(X) .



(non probabilistic) facts were omitted from the graph

Acyclic Programs

- Semantics is given by Clark Completion (substitute $\leftarrow$ with $\leftrightarrow$, not with $\neg$)
- Acyclic programs can be efficiently translated into Bayesian networks and vice-versa
  - $\therefore$ inference complexity in propositional acyclic programs is #P-equivalent

Theorem

Inferential complexity of acyclic propositional PLP*s such that each atom is the head of at most one rule is #P-equivalent even if*

1. $\mathbf{Q}$ *contains only* true *(and the program may not be definite)*
2. *the program is definite (and* $\mathbf{Q}$ *may contain* false*)*

Note: Inference in definite programs where no two heads unify is tractable if $\mathbf{Q}$ contains only true

Acyclic Programs

With a bound on arity, there at most polynomially many ground atoms; however

Theorem

The inferential complexity of inference in acyclic PLP*s with bounded predicate arity is* #NP*-equivalent*

#NP is the class of functions computed by a counting Turing machine with a NP oracle in polynomial time

- Grounding is intractable under common assumptions

Acyclic Programs

Without bound on arity:

Theorem

The inferential complexity of inference in acyclic PLP*s is* #EXP*-equivalent*

We can specify an exponentially large propositional relevant network

Acyclic Programs

Theorem

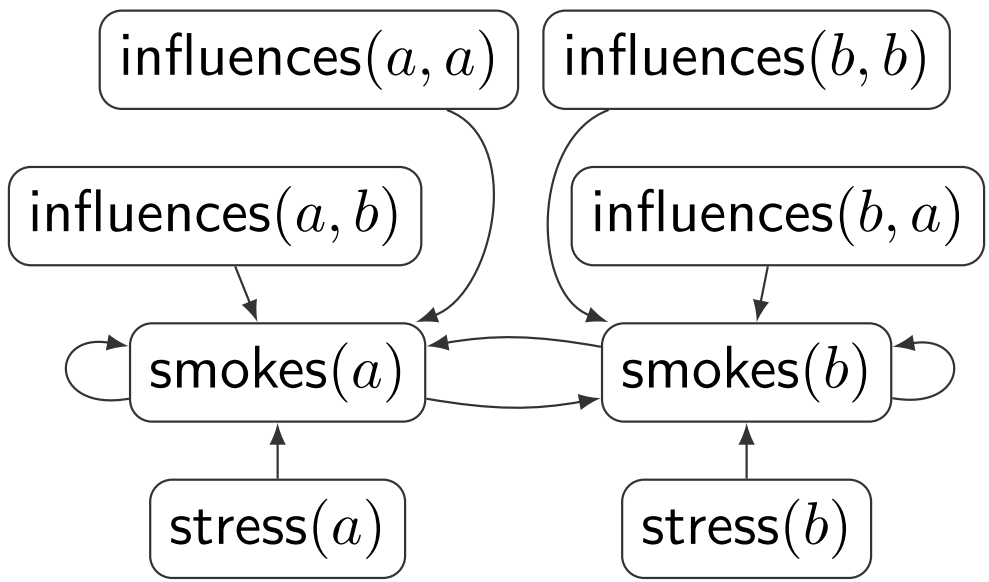
The query complexity of inference for acyclic PLP*s is #*P*-equivalent*

Stratified (Cyclic) Programs

$0.3 :: \mathsf{influences}(a, b)$. $0.3 :: \mathsf{influences}(b, a)$. $0.8 :: \mathsf{stress}(b)$.

$\mathsf{smokes}(X) \leftarrow \mathsf{stress}(X)$.

$\mathsf{smokes}(X) \leftarrow \mathsf{influences}(Y, X), \mathsf{smokes}(Y)$.



Stratified (Cyclic) Programs

- Clark completion encodes multiple models (one of which is the well-founded)

$$\mathsf{sm}(a) \Leftrightarrow \mathsf{in}(b, a) \wedge \mathsf{sm}(b).$$
$$\mathsf{sm}(b) \Leftrightarrow \mathsf{st}(b) \vee (\mathsf{in}(a, b) \wedge \mathsf{sm}(a)).$$

- Need to use loop formulas

$$\neg(\mathsf{st}(a) \vee \mathsf{st}(b)) \Rightarrow \neg\mathsf{sm}(a) \wedge \neg\mathsf{sm}(b).$$

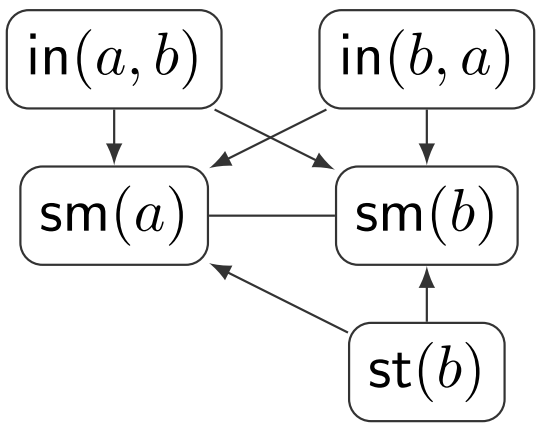
- Clark completion and loop formulas specify (very dense) chain graph (LWF semantics)

Stratified (Cyclic) Programs

$0.3 :: \mathsf{influences}(a, b)\,.\quad 0.3 :: \mathsf{influences}(b, a)\,.\quad 0.8 :: \mathsf{stress}(b)\,.$

$\mathsf{smokes}(X) \leftarrow \mathsf{stress}(X)\,.$

$\mathsf{smokes}(X) \leftarrow \mathsf{influences}(Y, X), \mathsf{smokes}(Y)\,.$



Stratified (Cyclic) Programs

$b \leftarrow a, f\,.$

$f \leftarrow b, g\,.$

$g \leftarrow b\,.$

$c \leftarrow \mathbf{not}\ b\,.$

$h \leftarrow c\,.$

$i \leftarrow h\,.$

$j \leftarrow i\,.$

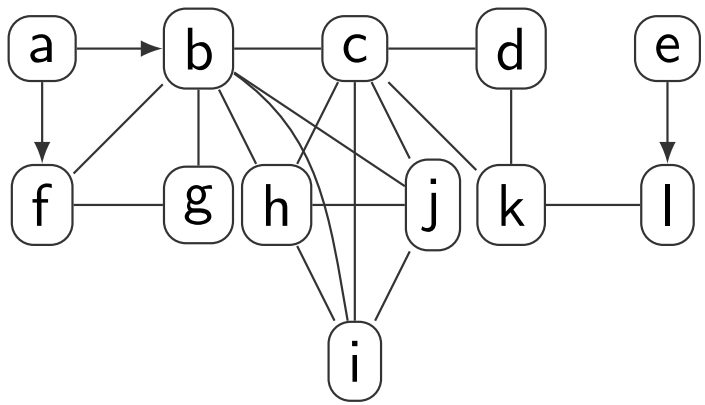
$c \leftarrow j\,.$

$d \leftarrow \mathbf{not}\ c\,.$

$k \leftarrow d\,.$

$d \leftarrow k\,.$

$l \leftarrow k, e\,.$



Stratified (Cyclic) Programs

Loop formulas lead to an exponentially large program, however

Theorem

For locally stratified PLP*s, inferential complexity is*

- *#*P*-equivalent for propositional programs,*
- *#*NP*-equivalent for programs with bounded predicate arity,*
- *#*EXP*-equivalent for general programs.*

Theorem

For locally stratified PLP*s, query complexity is #*P*-equivalent*

THEOREM

Inferential complexity is polynomial for queries containing only positive literals, for PLP*s where each atom is the head of at most one rule, and rules take one of the following forms:*

$$\alpha :: \mathsf{a}(X)\,.$$
$$\beta :: \mathsf{a}(a)\,.$$
$$\gamma :: \mathsf{r}(X, Y)\,.$$
$$\mathsf{a}(a)\,.$$
$$\mathsf{r}(a, b)\,.$$
$$\mathsf{a}(X) \leftarrow \mathsf{a}_1(X), \ldots, \mathsf{a}_k(X)\,.$$
$$\mathsf{a}(X) \leftarrow \mathsf{r}(X, Y)\,.$$
$$\mathsf{a}(X) \leftarrow \mathsf{r}(Y, X)\,.$$

Conclusion

- PLPs provide a flexible and intuitive way to specify complex PGMs
- Acyclic programs are equivalent to Bayesian networks
- Stratified (cyclic) programs can be interpreted as chain graphs
- Inferential complexity ranges form #P to #NP to #EXP
- Query complexity is #P
- A few tractable cases

Open questions

- Equivalence between chain graphs and stratified programs
- Non-stratified programs, tight programs, strict programs, order-consistent, ...
- Allowing functions, disjunctive clauses, arithmetic, ...