KRR with Uncertainty

LP^{MLN} Inference Examples

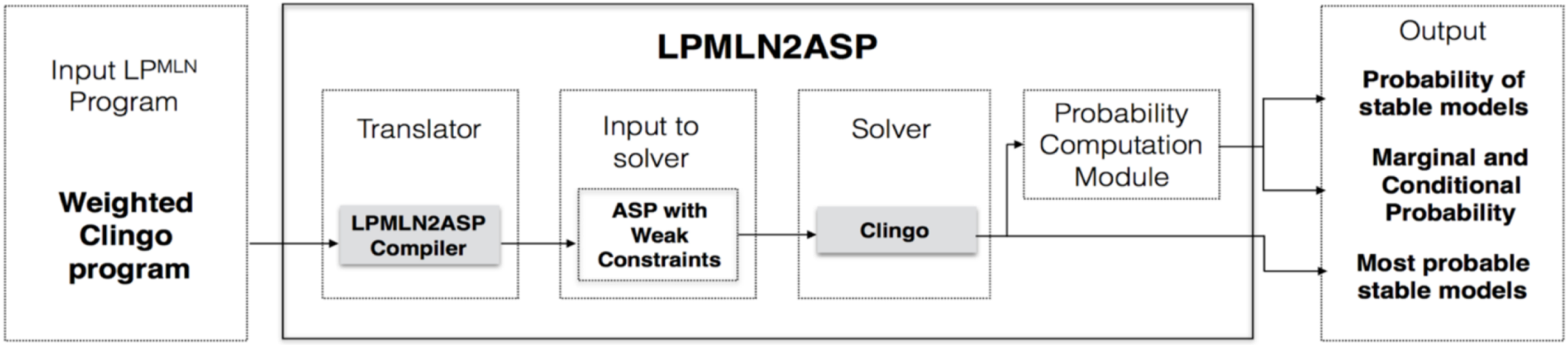
Ira A. Fulton Schools of Engineering
Arizona State University

Objectives

Objective

Apply probabilistic logic programming tool to solving challenging problems

System Architecture: lpmln-infer



- Can compute MAP inference, marginal and conditional probability
- MAP inference is directly computed by clingo
- Probability calculations are computed by a probability computation module

Input Language of lpmln-infer

- The input language resembles the input language of clingo
- Hard rules are encoded exactly the same as clingo rules
- Soft rules are clingo rules with weight prepended

```
% File: bird.lpmln
bird(X) :- residentbird(X).
bird(X) :- migratorybird(X).
:- residentbird(X), migratorybird(X).
2 residentbird(jo).
1 migratorybird(jo).
```

Example: Finding Most Probable Stable Models

```
% bird.lpmln
bird(X) :- residentbird(X).
bird(X) :- migratorybird(X).
:- residentbird(X), migratorybird(X).
2 residentbird(jo).
1 migratorybird(jo).
```

```
$ lpmln-infer bird.lpmln
```

```
Answer: 1
unsat(5,"1") unsat(4,"2")
Optimization: 3000
Answer: 2
unsat(5,"1") residentbird(jo) bird(jo)
Optimization: 1000
OPTIMUM FOUND
```

Example: Probabilities of All Stable Models

```
% bird.lpmln
bird(X) :- residentbird(X).
bird(X) :- migratorybird(X).
:- residentbird(X), migratorybird(X).
2 residentbird(jo).
1 migratorybird(jo).
```

```
$ lpmln-infer bird.lpmln -all
```

[unsat(5,"1"), unsat(4,"2")] : 0.09003057317038046
[residentbird(jo), bird(jo), unsat(5,"1")] : 0.6652409557748219
[bird(jo), migratorybird(jo), unsat(4,"2")] : 0.2447284710547976 7

Example: Marginal Probability of Query

```
% bird.lpmln
bird(X) :- residentbird(X).
bird(X) :- migratorybird(X).
:- residentbird(X), migratorybird(X).
2 residentbird(jo).
1 migratorybird(jo).
```

query atoms

```
$ lpmln-infer bird.lpmln -q residentbird

residentbird(jo) 0.665240955775
```

- **The command is same as**

```
$ lpmln-infer bird.lpmln -q residentbird –exact
```

- **Alternatively one can use sampling-based inference**

```
$ lpmln-infer bird.lpmln -q residentbird –mcasp
```

Example: Conditional Probability of Query

P(residentbird(jo) | bird(jo))

```
% bird.lpmln
bird(X) :- residentbird(X).
bird(X) :- migratorybird(X).
:- residentbird(X), migratorybird(X).
2 residentbird(jo).
1 migratorybird(jo).
```

```
% bird-evid.db
:- not bird(jo).
```

```
$ lpmln-infer bird.lpmln -e bird-evid.db -q residentbird
```

evidence file: set of asp constraints

```
residentbird(jo) : 0.7310585786300049
```

Example: Debugging in ASP

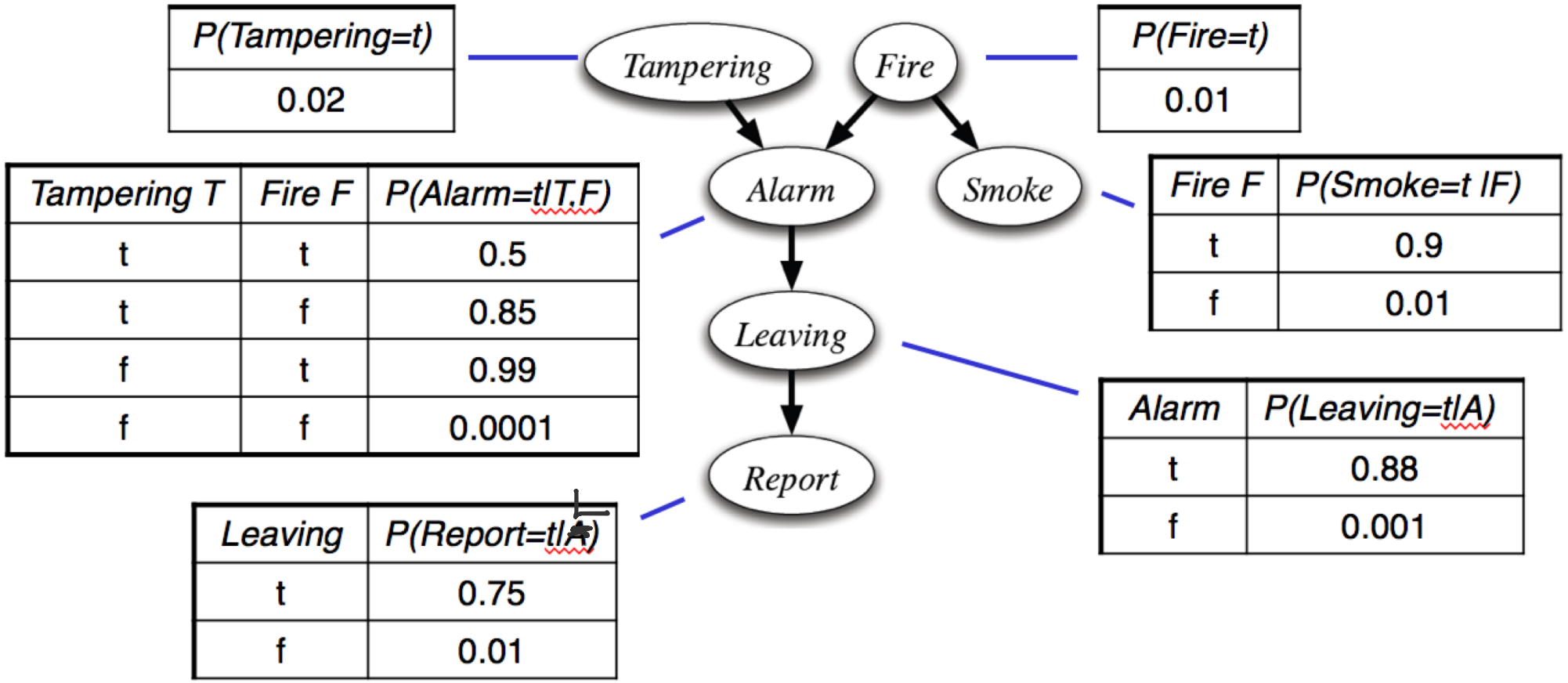
```
% bird1.lpmln
bird(X) :- residentbird(X).
bird(X) :- migratorybird(X).
:- residentbird(X), migratorybird(X).
residentbird(jo).
migratorybird(jo).
```


```
$ lpmln-infer bird1.lpmln -all -hard
```

[bird(jo), migratorybird(jo), unsat(4,"a")] : 0.333333333333333
[residentbird(jo), bird(jo), unsat(3,"a",jo), migratorybird(jo)] : 0.333333333333333
[residentbird(jo), bird(jo), unsat(5,"a")] : 0.333333333333333

Representing Bayesian networks in LP^{MLN}

Recall: Example



P(Tampering=t)
0.02

P(Fire=t)
0.01

Tampering T	Fire F	P(Alarm=t|T,F)
t	t	0.5
t	f	0.85
f	t	0.99
f	f	0.0001

Fire F	P(Smoke=t |F)
t	0.9
f	0.01

Alarm	P(Leaving=t|A)
t	0.88
f	0.001

Leaving	P(Report=t|A)
t	0.75
f	0.01

Representing Bayesian Networks in LP^{MLN}

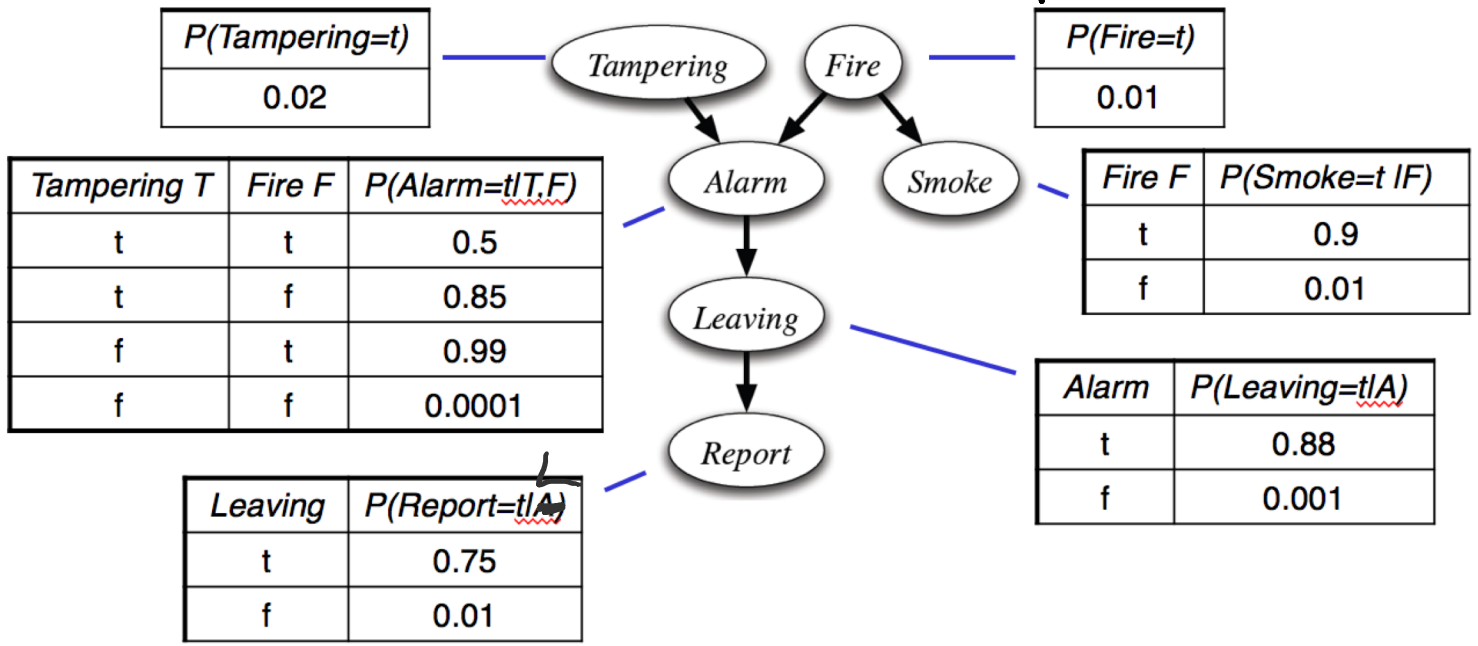
Encode CPT using auxiliary atoms

$\ln\left(\frac{p}{1-p}\right)$: pf

$$P(pf=t) = \frac{e^{\ln\left(\frac{p}{1-p}\right)}}{e^{\ln\left(\frac{p}{1-p}\right)} + e^{0}} = \frac{\frac{p}{1-p}}{\frac{p}{1-p} + 1} = p$$

$$P(pf=f) = \frac{1}{\frac{p}{1-p} + 1} = 1-p$$

```
@log(0.02/0.98) pf(t).
@log(0.01/0.99) pf(f).
@log(0.5/0.5)   pf(a,t1f1).
@log(0.85/0.15)  pf(a,t1f0).
@log(0.99/0.01)  pf(a,t0f1).
@log(0.0001/0.9999) pf(a,t0f0).
@log(0.9/0.1)  pf(s,f1).
@log(0.01/0.99) pf(s,f0).
@log(0.88/0.12)   pf(l,a1).
@log(0.001/0.999) pf(l,a0).
@log(0.75/0.25) pf(r,l1).
@log(0.01/0.99) pf(r,l0).
```



P(Tampering=t)
0.02

P(Fire=t)
0.01

Tampering T	Fire F	P(Alarm=t|T,F)
t	t	0.5
t	f	0.85
f	t	0.99
f	f	0.0001

Fire F	P(Smoke=t |F)
t	0.9
f	0.01

Alarm	P(Leaving=t|A)
t	0.88
f	0.001

Leaving	P(Report=t|A)
t	0.75
f	0.01

Representing Bayesian Networks in LP^{MLN}

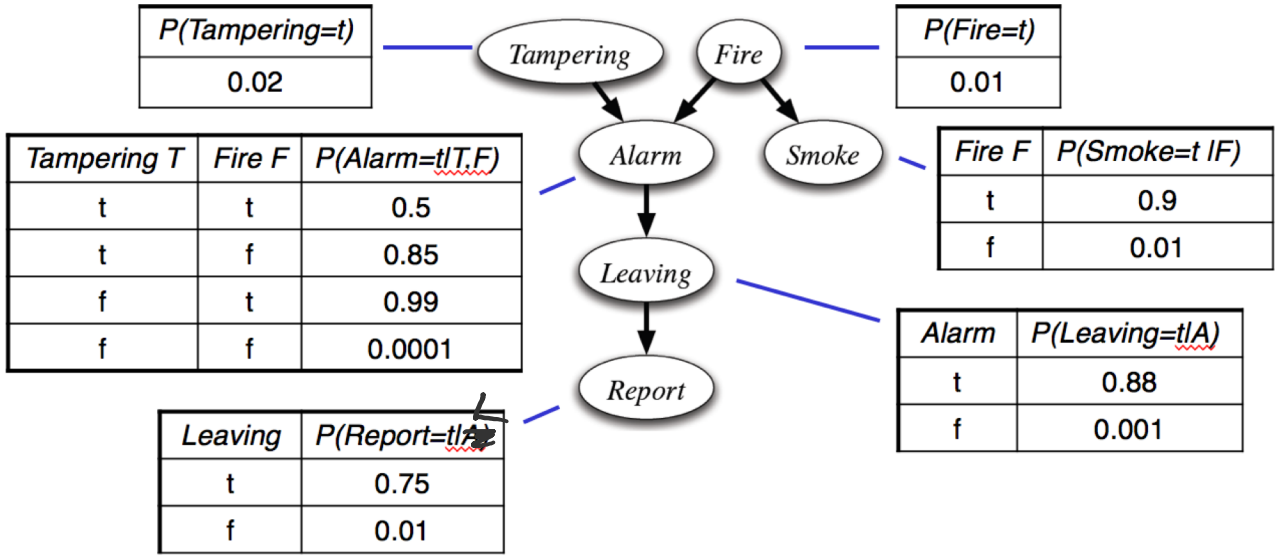
Encode DAG in rules:

tampering :- pf(t).

fire :- pf(f).

alarm :- tampering, fire, pf(a,t1f1).
alarm :- tampering, not fire, pf(a,t1f0).
alarm :- not tampering, fire, pf(a,t0f1).
alarm :- not tampering, not fire, pf(a,t0f0).

smoke :- fire, pf(s,f1).
smoke :- not fire, pf(s,f0).



P(Tampering=t)
0.02

P(Fire=t)
0.01

Tampering T	Fire F	P(Alarm=t|T,F)
t	t	0.5
t	f	0.85
f	t	0.99
f	f	0.0001

Fire F	P(Smoke=t |F)
t	0.9
f	0.01

Alarm	P(Leaving=t|A)
t	0.88
f	0.001

Leaving	P(Report=t|L)
t	0.75
f	0.01

leaving :- alarm, pf(l,a1).
leaving :- not alarm, pf (l,a0).

report :- leaving, pf(r,l1).
report :- not leaving, pf(r,l0).

Representing Bayesian Networks in LP^{MLN}

Given a Bayes net N, the LP^{MLN} program contains the following:

- **a propositional atom V for each node in N**
- **a set of atoms PF(V, S(parent(V))) for each CPT P(V=1 | S(parent(V))), where S denotes a truth assignment to Parent(V)**
- **For each P(V=1 | S(parent(V))) = p include**
  - ln (p/(1-p)): PF(V, S(parent(V))) if 0<p<1
  - α: :- PF(V, S(parent(V))) if p=0
  - α: PF(V, S(parent(V))) if p=1
- **For each node V whose parents are V_1, …, V_n ,**
  - α: V :- A(V_1, S(V_1)), ..., A(V_n, S(V_n)), PF(V, S(V_1,..,V_n))
  - where A(V_i, S(V_i)) is
    - V_i if S(V_i)) is 1 and
    - not V_i otherwise

Representing Bayesian Networks in LP^{MLN}

```
// fire-bayes.lpmln

@log(0.02/0.98) pf(t).
@log(0.01/0.99) pf(f).
@log(0.5/0.5)   pf(a,t1f1).
@log(0.85/0.15)  pf(a,t1f0).
@log(0.99/0.01)  pf(a,t0f1).
@log(0.0001/0.9999) pf(a,t0f0).

@log(0.9/0.1)  pf(s,f1).
@log(0.01/0.99) pf(s,f0).

@log(0.88/0.12)   pf(l,a1).
@log(0.001/0.999) pf(l,a0).

@log(0.75/0.25) pf(r,l1).
@log(0.01/0.99) pf(r,l0).
```

```
tampering :- pf(t).

fire :- pf(f).

alarm :- tampering, fire, pf(a,t1f1).
alarm :- tampering, not fire, pf(a,t1f0).
alarm :- not tampering, fire, pf(a,t0f1).
alarm :- not tampering, not fire, pf(a,t0f0).

smoke :- fire, pf(s,f1).
smoke :- not fire, pf(s,f0).

leaving :- alarm, pf(l,a1).
leaving :- not alarm, pf (l,a0).

report :- leaving, pf(r,l1).
report :- not leaving, pf(r,l0).
```

Example Run

To compute P(fire | alarm, ¬tampering)

- Write into fire-evid.db contains

```
:- not alarm.
:- tampering.
```

- Call

```
$ lpmln-infer fire-bayes.lpmln –e fire-evid.db –q fire
```

Example Run

P(Fire)

```
$ lpmln-infer fire-bayes.lpmln -q fire
```

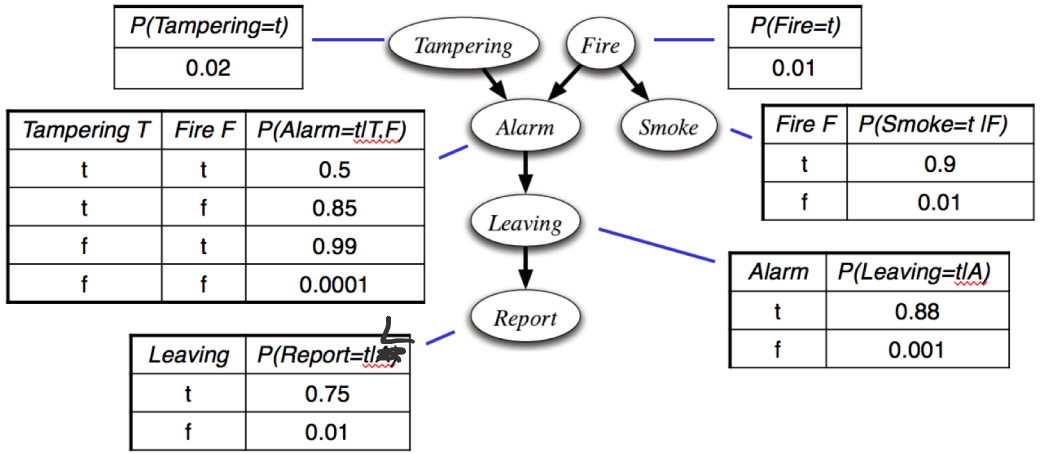
P(Fire | Report)

```
$ lpmln-infer fire-bayes.lpmln -e fire-evid.db -q fire
```

P(Fire | Alarm)

P(Fire | Alarm, ¬Tampering)

P(Fire | Alarm, Tampering)



Diagnostic Inference

Compute the probability of the cause given the effect

To compute P(fire = t | leaving = t), the user can invoke

```
$ lpmln-infer fire-bayes.lpmln -e fire-evid.db -q fire
```

where fire-evid.db contains the line

```
:- not leaving.
```

This outputs

```
fire : 0.35215453804538244
```

Predictive Inference

Compute the probability of effect given the cause.

To compute P(leaving = t | fire = t), the user can invoke

```
$ lpmln-infer fire-bayes.lpmln -e fire-evid.db -q leaving
```

where fire-evid.db contains the line

```
:- not fire.
```

This outputs

```
leaving 0.862603541626
```

Mixed Inference

Combine predictive and diagnostic inference.

To compute P(alarm = t | fire = f, leaving = t), the user can invoke

```
$ lpmln-infer fire-bayes.lpmln -e fire-evid.db -q alarm
```

where fire-evid.db contains two lines

```
:- fire.
:- not leaving.
```

This outputs

```
alarm : 0.9386803111482813
```

Intercausal inference (Explaining Away)

Reasons about the mutual causes (effects) of a common effect

Knowing that there was tampering explains away alarm, and hence affecting the probability of fire.

P(fire = t | alarm = t, tampering = t) using lpmln-infer outputs

```
fire : 0.005906674542232707
```

P(fire = t | alarm = t, tampering = f) using lpmln-infer outputs

```
fire : 0.9900990099009899
```

Representing Probabilistic Graph Problems

Example: Probabilistic Path (1 of 2)

- **ASP encoding of graph problems can be easily turned into probabilistic extensions. E.g.,**
  - "given that there is a path between two nodes, what is the most likely graph?": MAP inference
  - "given two nodes, what is the probability that there exists a path between them?": probabilistic query

- **We put ln(p/(1-p)) as the weight of the rule edge(X, Y)**

```
@log(0.3/0.7)  edge(0, 1).
@log(0.2/0.8)  edge(1, 2).
    …
```

Example: Probabilistic Path (2 of 2)

We represent path relation as hard rules:

```
path(X,Y) :- edge(X, Y).
path(X,Y) :- path(X, Z), path(Z, Y), Y != Z.
```

Probabilistic Traveling Salesman: ”Given a graph with uncertain edges, what is the probability that there is a Hamiltonian circuit? “

Example: Network Connectivity (1 of 3)

```
node(1..4).

@log(0.8/0.2)  fail(2).
@log(0.5/0.5)  fail(3).
@log(0.2/0.8)  fail(4).

edge(1,2).   edge(2,4).     edge(1,3).     edge(3,4).
edge(2,3).

connected(X,Y) :- edge(X, Y), not fail(X), not fail(Y).
connected(X,Y) :- connected(X,Z), connected(Z,Y).
```

Example: Network Connectivity (2 of 3)

Q: What is the probability that 1 and 4 are connected?

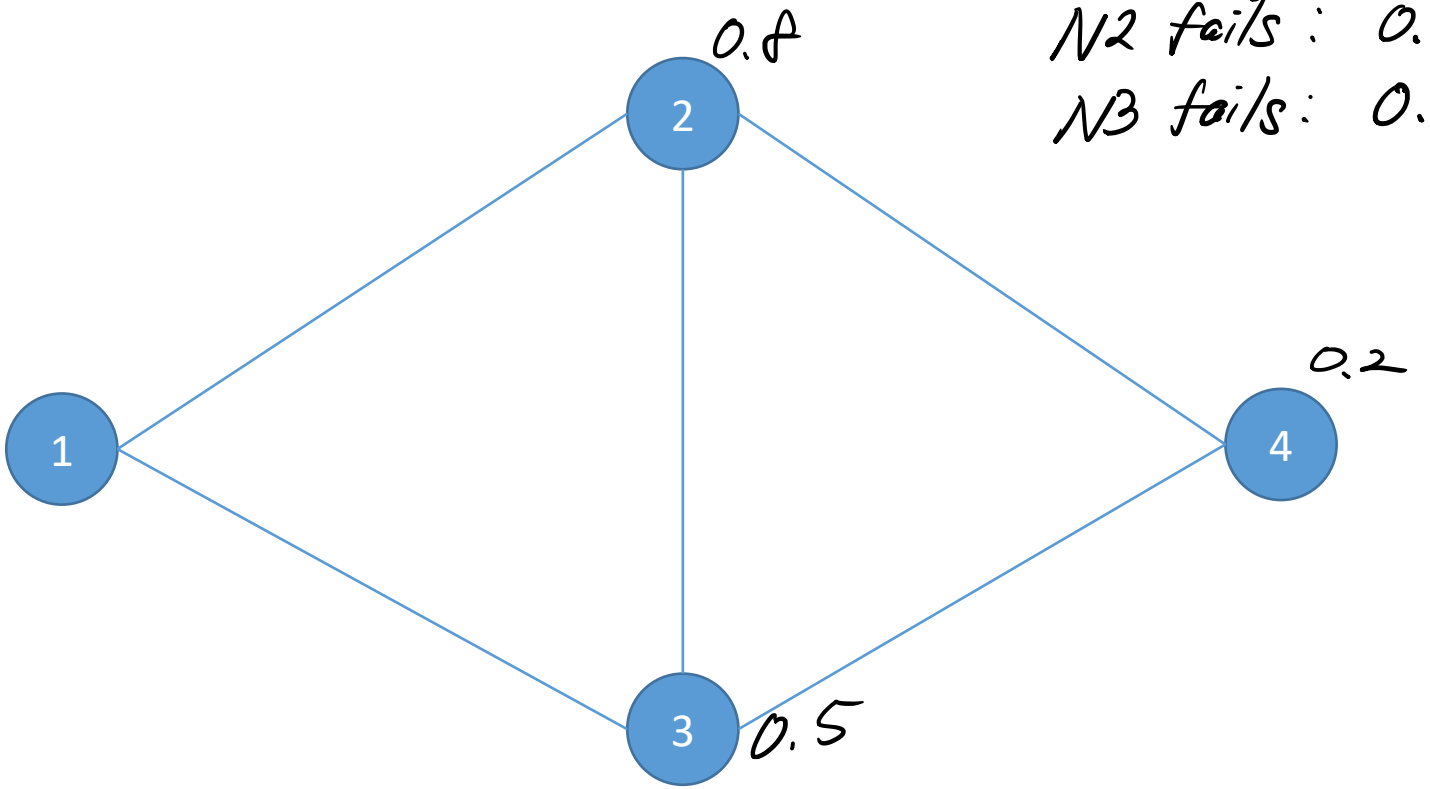
A. 0.32 B. 0.4 C. 0.16 D. 0.6

All okay : $0.2 \times 0.5 \times 0.8 = 0.08$

N2 fails : $0.8 \times 0.5 \times 0.8 = 0.32$

N3 fails : $0.2 \times 0.5 \times 0.8 = 0.08$

0.48



Example: Network Connectivity (3 of 3)

```
$ lpmln-infer networks.lpmln -q connected

connected(1, 2) : 0.19999999999999998
connected(2, 4) : 0.16
connected(1, 3) : 0.5
connected(3, 4) : 0.4
connected(2, 3) : 0.1
connected(1, 4) : 0.48000000000000004
```

Example: Virus (1 of 2)

person(a;b;c;d;e;f;g).

1.5 has_disease(X) :- carries_virus(X).

1.1 carries_virus(Y) :- contact(X, Y), carries_virus(X).

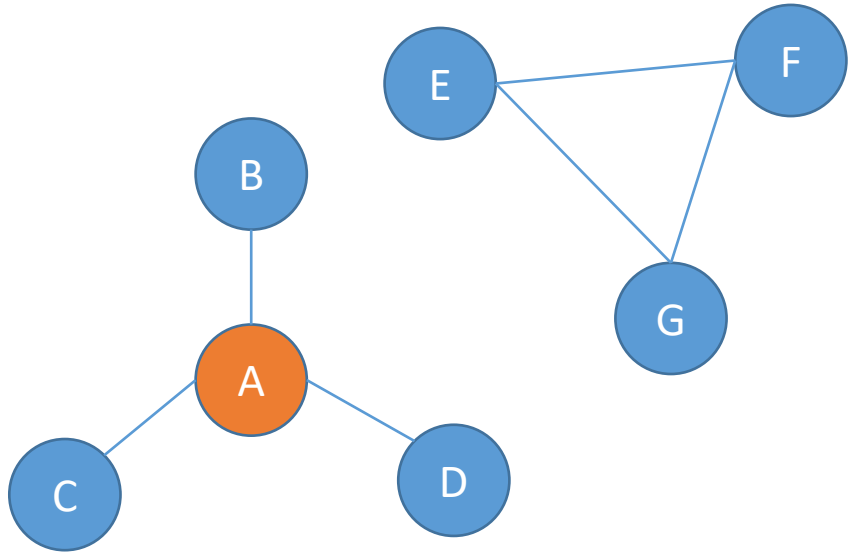
carries_virus(a).

contact(a,(b;c;d)).

contact(e,(f;g)).

contact(f,g).

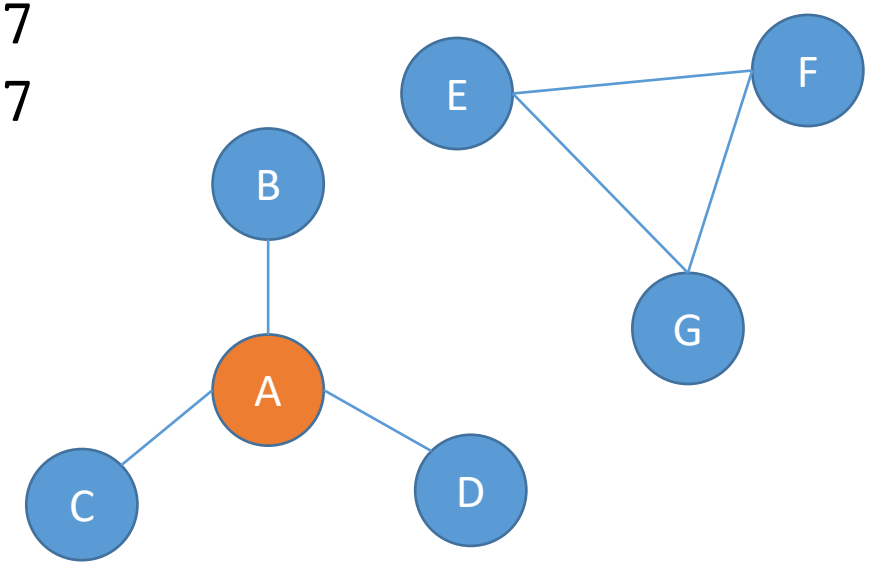
contact(X,Y) :- contact(Y,X).



Example: Virus (2 of 2)

```
$ lpmln-infer input.lpmln -exact -q carries_virus,has_disease
```

```
carries_virus("A") : 1.0000000000000002
carries_virus("B") : 0.7860727393281469
carries_virus("C") : 0.786072739328147
carries_virus("D") : 0.786072739328147
has_disease("B") : 0.6426730081063122
has_disease("C") : 0.6426730081063122
has_disease("D") : 0.6426730081063122
has_disease("A") : 0.8175744761936435
```



Wrap-Up