Compacting Boolean Formulae for Inference in PLP

By Theofrastos Mantadelis, Dimitar Shterionov and Gerda Janssens

U.PORTO

Hard Problems

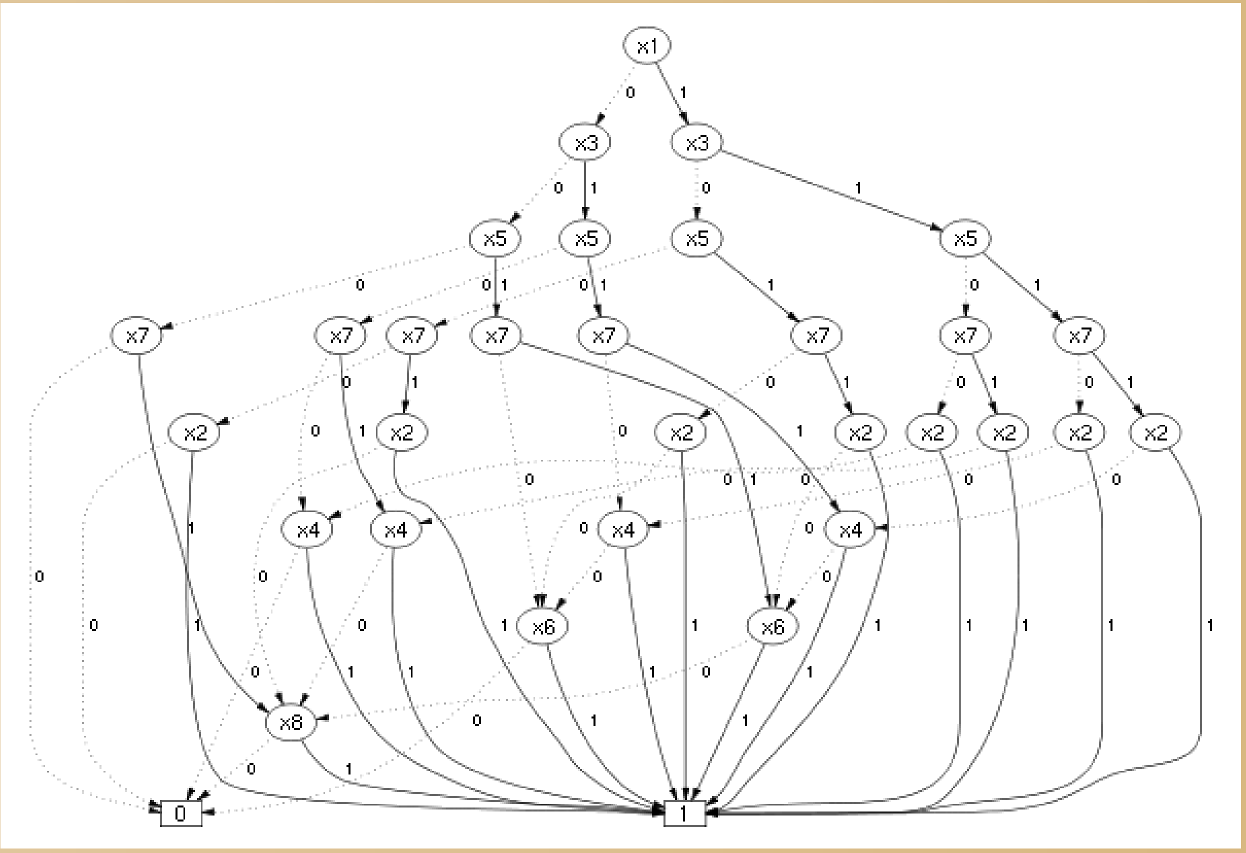
- Boolean Formulae are used:
  - Satisfiability (SAT solvers)
  - Formal Verification
  - Scheduling
- Knowledge Compilation
  - Compile Boolean Formulae

Knowledge Compilation

- Compilation Languages
  - NNF, DNF, CNF, ...
  - sd-DNNF, ROBBD, SDD, ...
- Properties
  - Decomposability (Conjuncts do not share variables)
  - Determinism (Disjuncts are logically disjoint)
  - Smoothness (Disjuncts mention the same set of variables)
  - Ordering (Decision variables appear in the same order)
- Polytime Operations
- Polytime Transformations

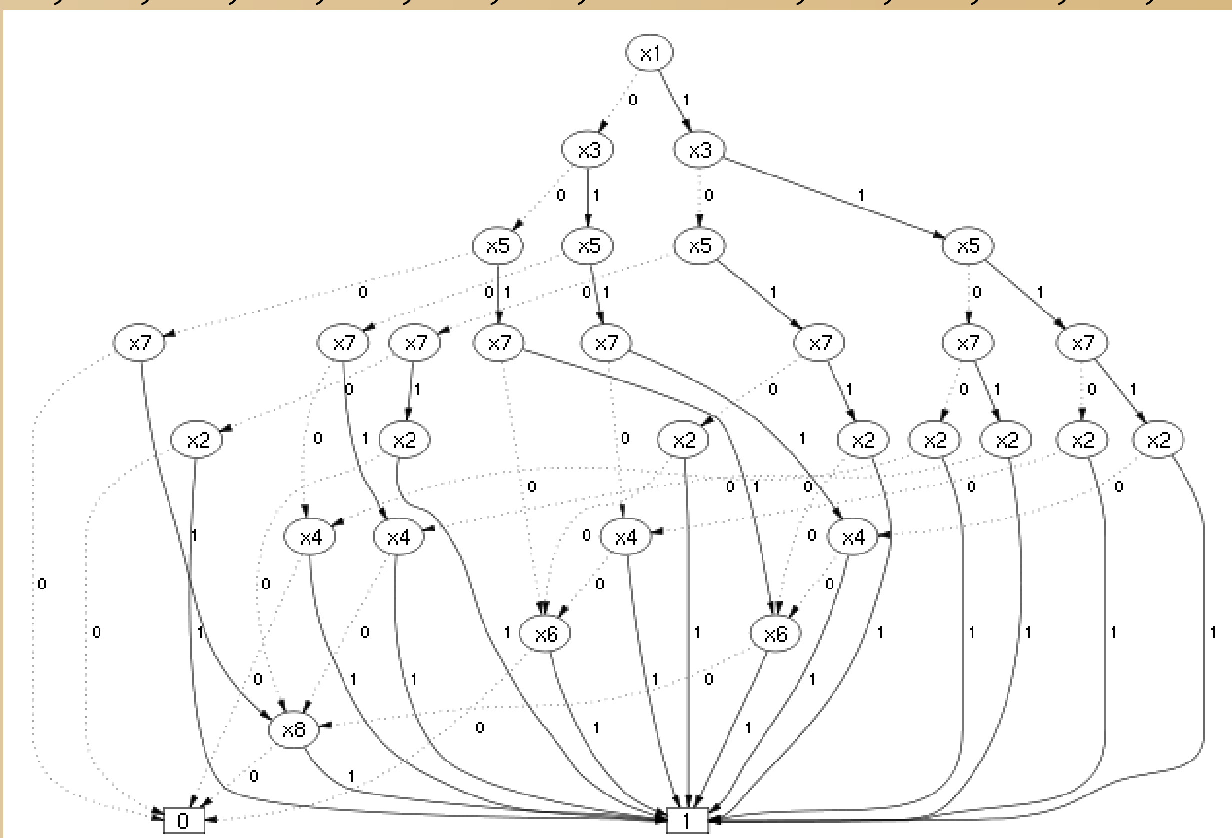
Compiling Boolean Formulae

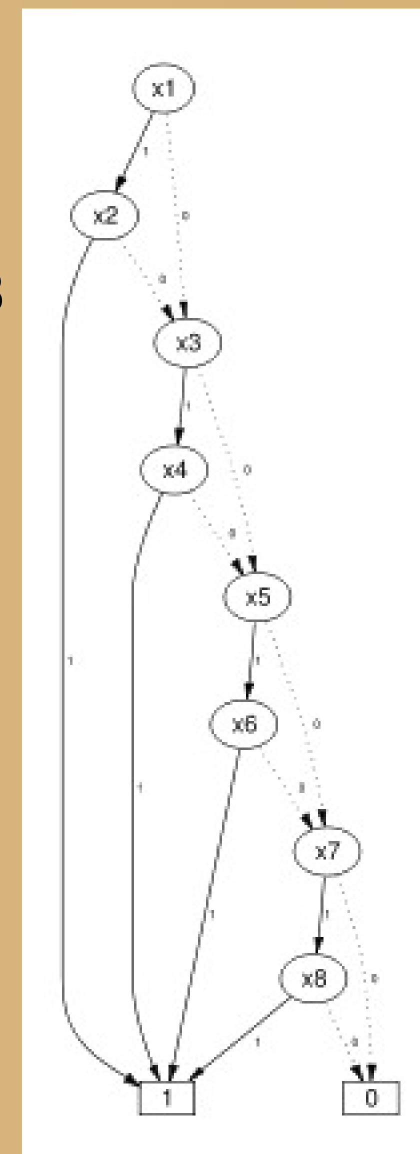
- Usually NP hard or worse!
  - ROBDDs
  - F=(X1^X2)v(X3^X4)v(X5^X6)v(X7^X8)
  - X1,X3,X5,X7,X2,X4,X6,X8



Compiling Boolean Formulae

- Usually NP hard or worse!
  - ROBDDs
  - F=(X1^X2)v(X3^X4)v(X5^X6)v(X7^X8)
  - X1,X3,X5,X7,X2,X4,X6,X8 → X1,X2,X3,X4,X5,X6,X7,X8





Motivation

- We use Boolean Formulae
  - To represent models
  - We use ROBBDs and sd-DNNFs for compilation
  - Probabilistic Logic Programs often Intractable

ProbLog

- Probabilistic Logic Programming Language
  - Extents Prolog with probabilities

```
0.6::e(a, b).   0.7::e(c, d).
0.2::e(e, f).   0.3::e(a, d).
0.4::e(d, f).   0.8::e(b, c).
0.4::e(d, e).
p(X, Y) :- e(X, Y).
p(X, Y) :- e(X, X1), p(X1, Y).
```

ProbLog

- Probabilistic Logic Programming Language
  - Extents Prolog with probabilities

```
0.6::e(a, b).   0.7::e(c, d).
0.2::e(e, f).   0.3::e(a, d).
0.4::e(d, f).   0.8::e(b, c).
0.4::e(d, e).
p(X, Y) :- e(X, Y).
p(X, Y) :- e(X, X1), p(X1, Y).
```

  - Defines a Probabilistic Distribution

$$P\left(L^d\right)=\prod_{f_i\in L^d} p_i \cdot \prod_{f_i\in L\setminus L^d} \left(1-p_i\right)$$

ProbLog

- Probabilistic Logic Programming Language
  - Extents Prolog with probabilities

```
0.6::e(a, b).   0.7::e(c, d).
0.2::e(e, f).   0.3::e(a, d).
0.4::e(d, f).   0.8::e(b, c).
0.4::e(d, e).
p(X, Y) :- e(X, Y).
p(X, Y) :- e(X, X1), p(X1, Y).
```

  - Defines a Probabilistic Distribution

$$P(L^d) = \prod_{f_i \in L^d} p_i \cdot \prod_{f_i \in L \setminus L^d} (1 - p_i)$$

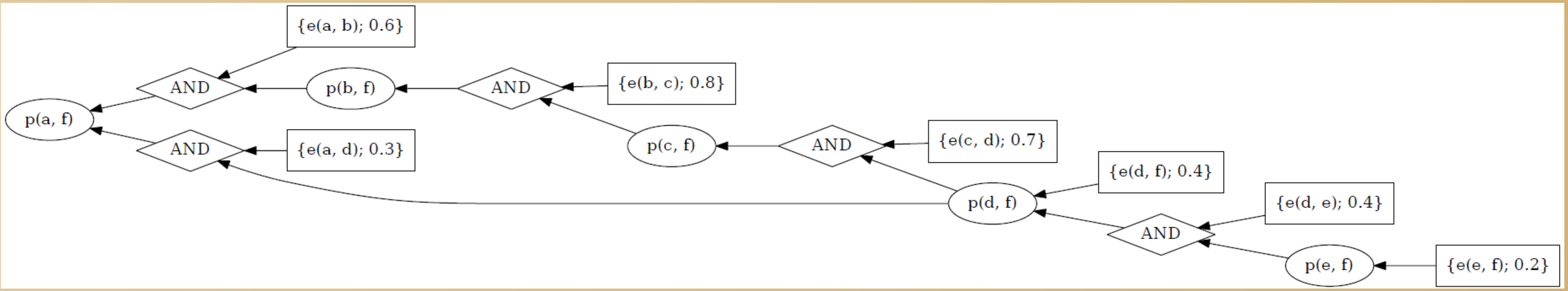
  - Queries
    - Marginal and Conditional

AND-OR Tree

- Intermediate Structure used for BF representation

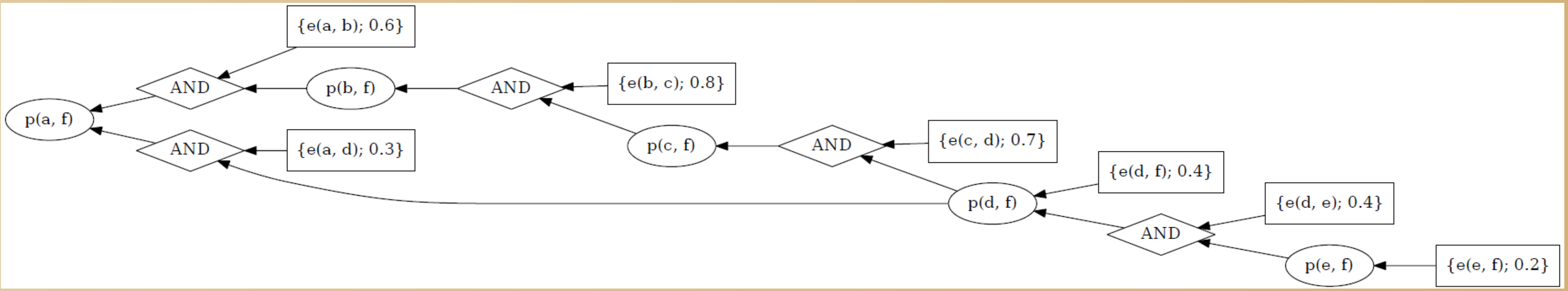
```
0.6::e(a, b).   0.7::e(c, d).
0.2::e(e, f).   0.3::e(a, d).
0.4::e(d, f).   0.8::e(b, c).
0.4::e(d, e).
p(X, Y) :- e(X, Y).
p(X, Y) :- e(X, X1), p(X1, Y).

?- problog_exact(p(a, f), P).
```



AND-OR Tree

- Intermediate Structure used for BF representation
  - AND nodes
  - OR nodes
  - Terminal nodes (probabilistic facts)
  - **Cyclic Structure & General Negation**

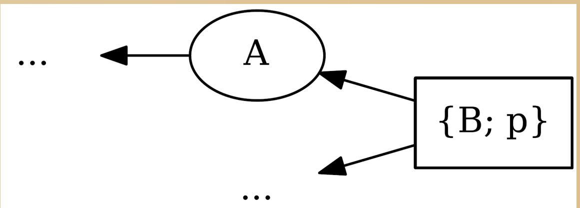


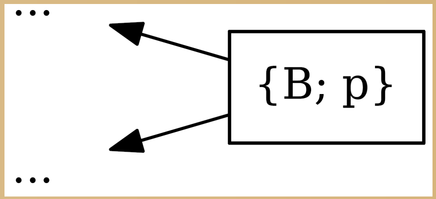
Compacting Boolean Formulae

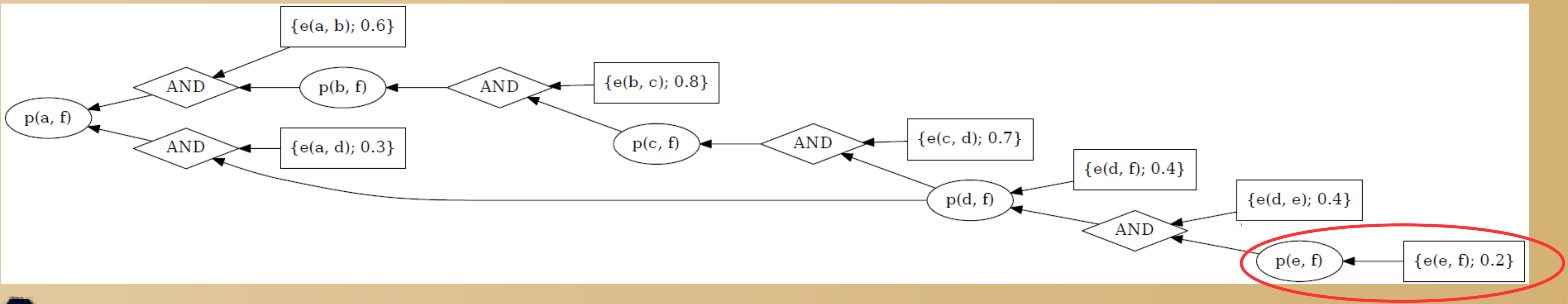
- Perform Polytime pattern detections
- Perform linear pattern compactions
  - Simplify the Boolean Formulae
    - Reduces the number of operations
  - Reduce the number of binary variables
    - Reduces the search space for Compilation

Detect Patterns

- Single Variable $O(N_{or} \cdot (\log(N_{or}) + \log(N_{and})))$ Compaction

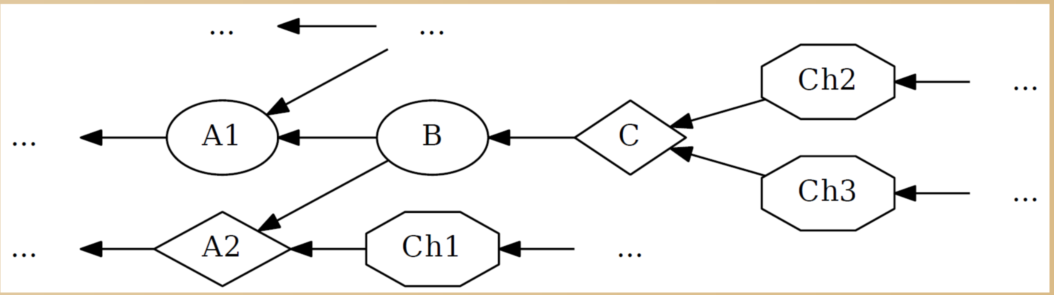


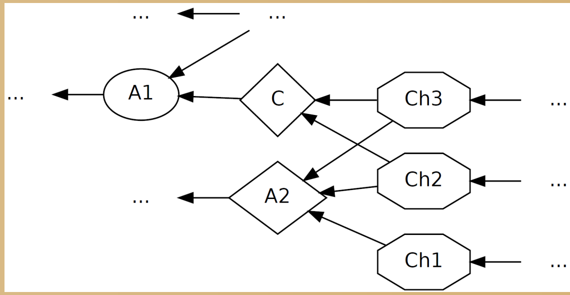


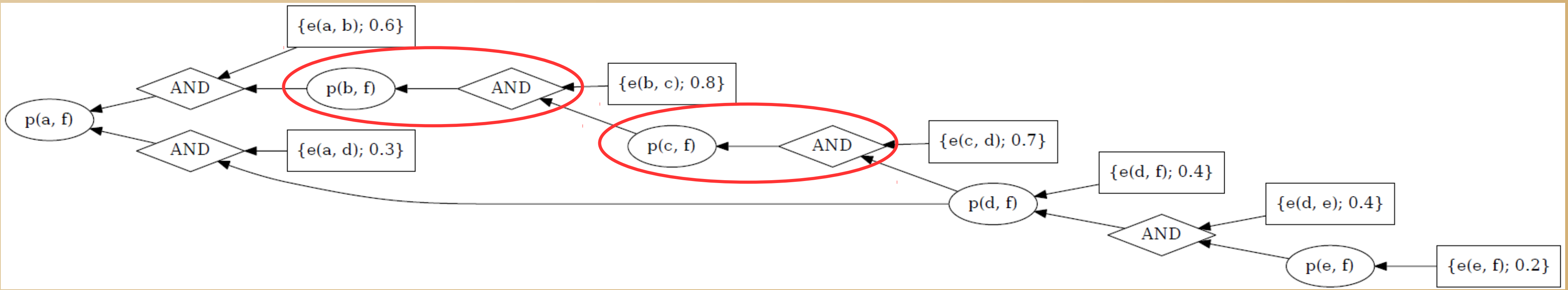


Detect Patterns

- Single Branch I $O(N_{or} \cdot (\log(N_{or}) + \log(N_{and})))$ Compaction

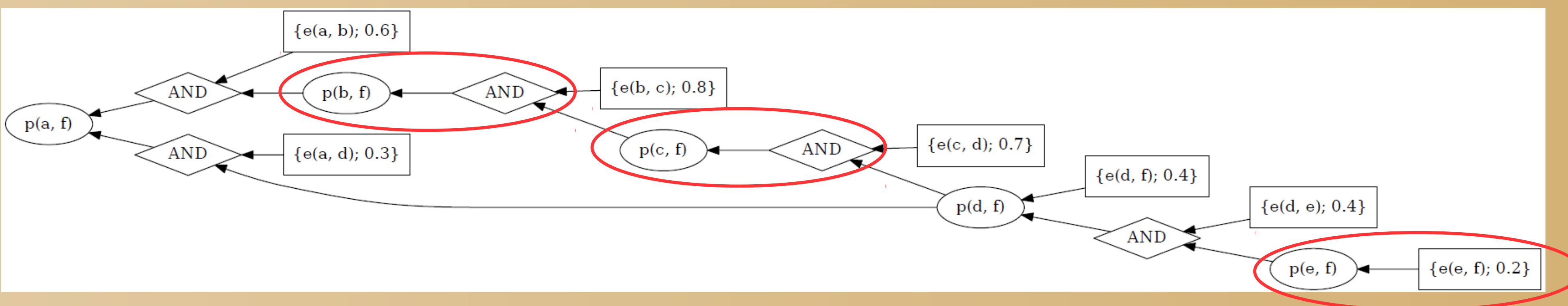


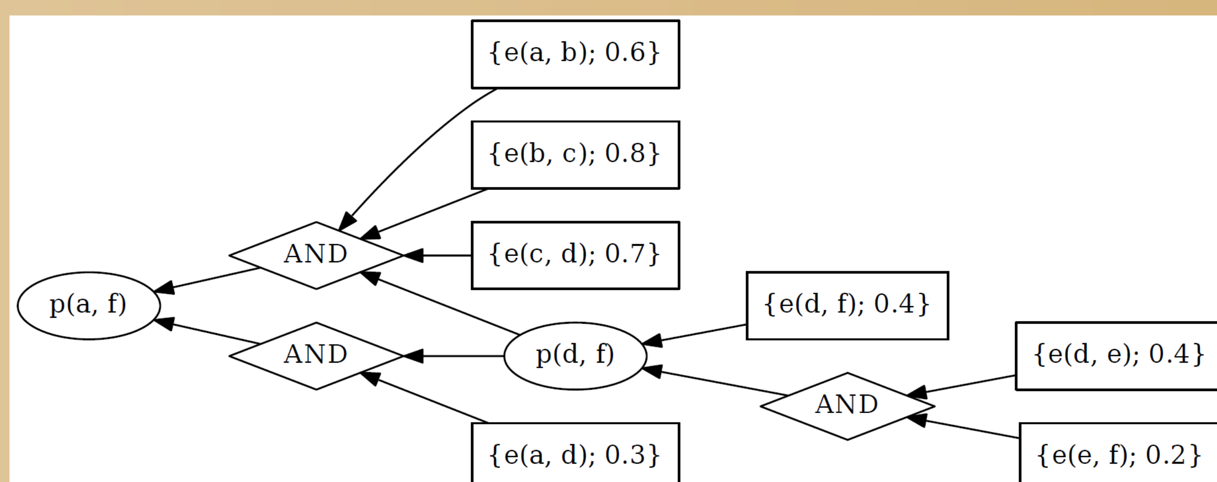




Detect Patterns

- Compacting

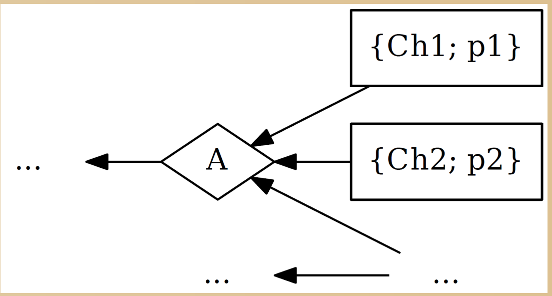


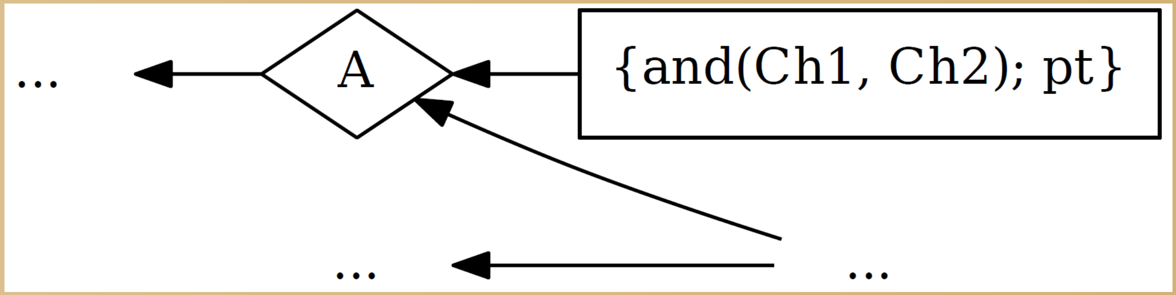


Detect Patterns

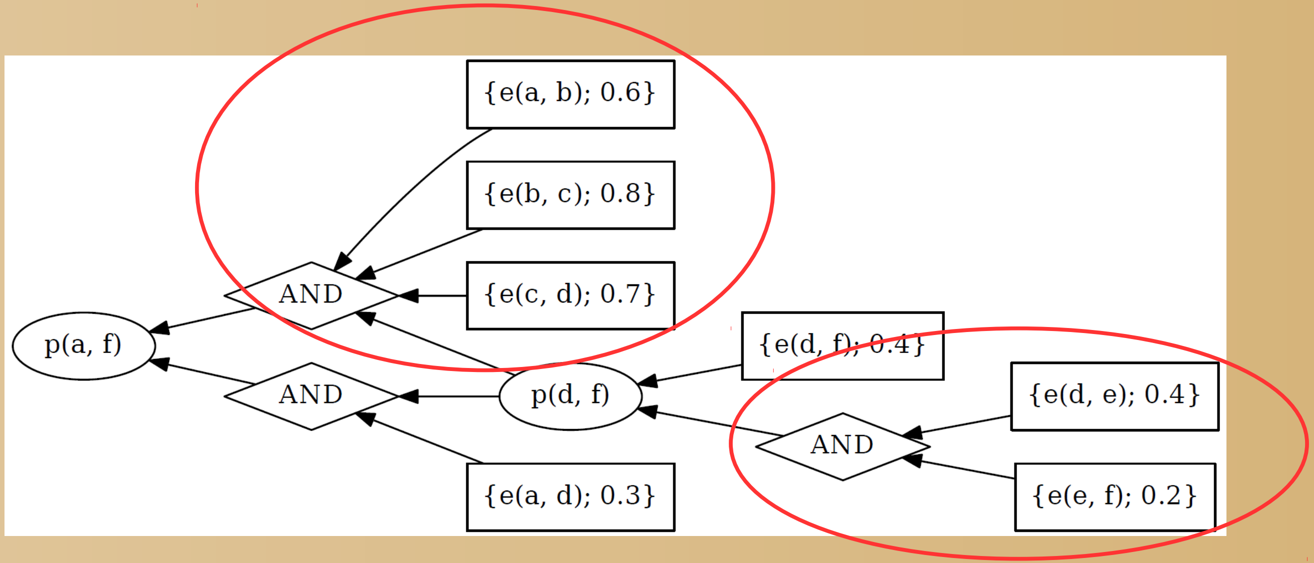
- AND-Cluster $O(N_{\text{and}}^2 \cdot N_{\text{term}})$

Compaction



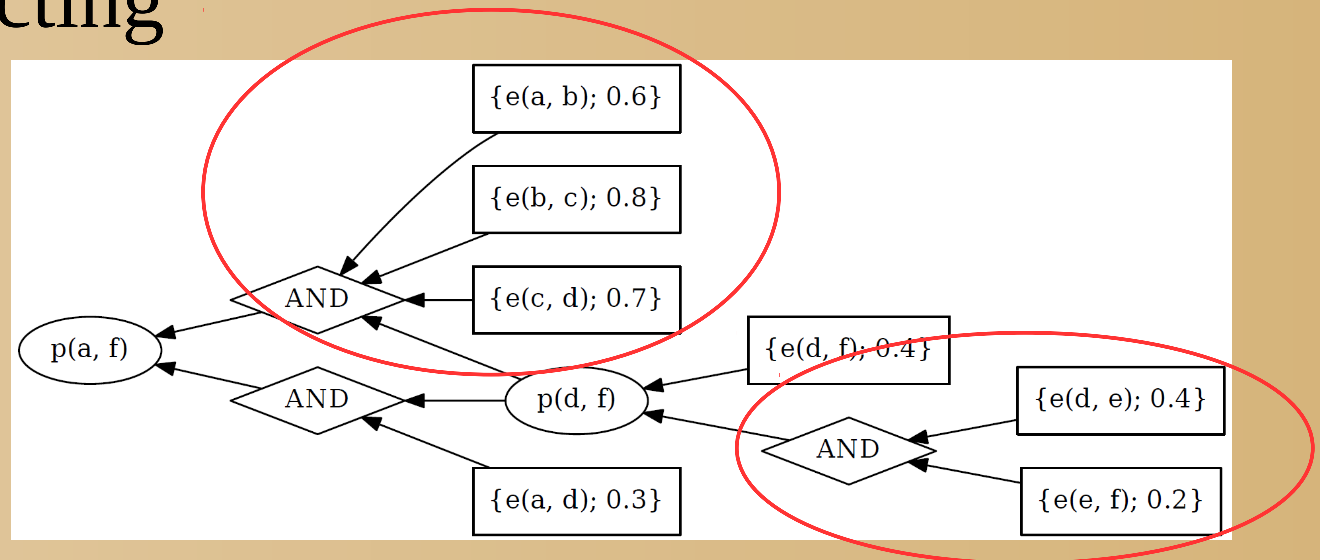


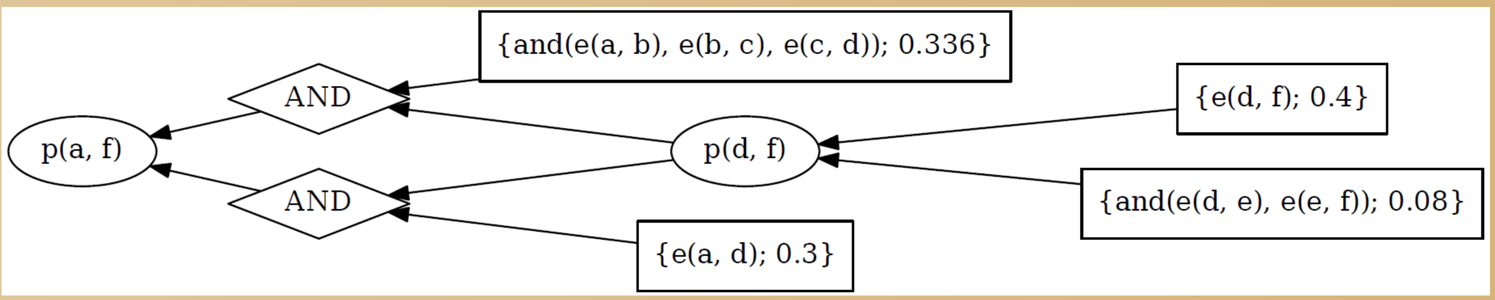
$$p_t = \prod_{C_i \in Ch'_A} p_i$$



Detect Patterns

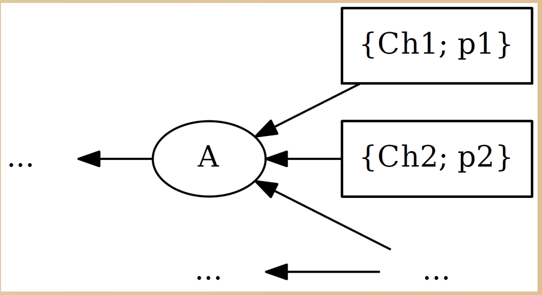
- Compacting



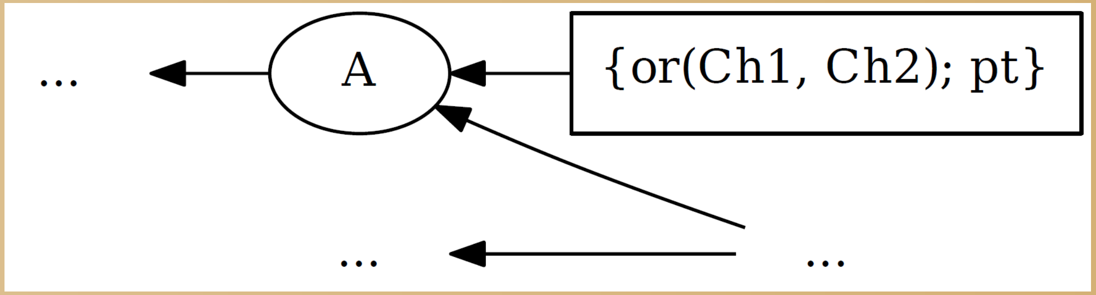


Detect Patterns

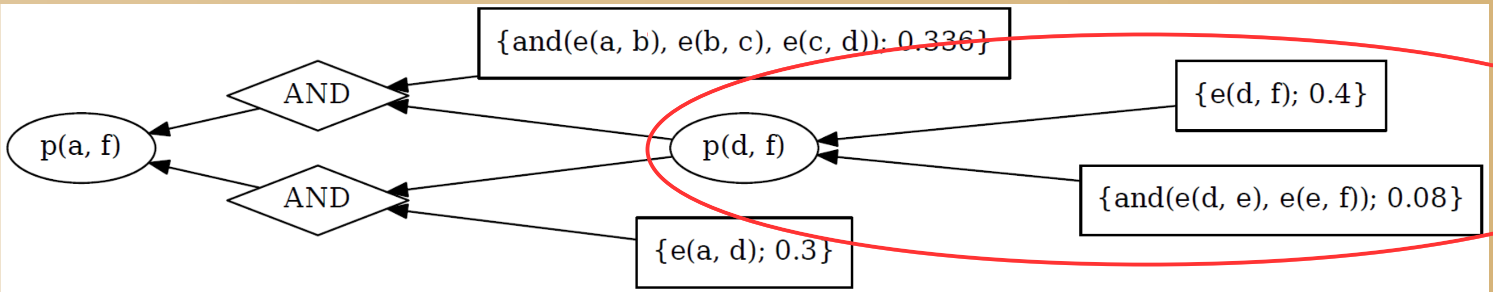
- OR-Cluster I $O(N_{\text{or}}^2 \cdot N_{\text{term}})$



Compaction

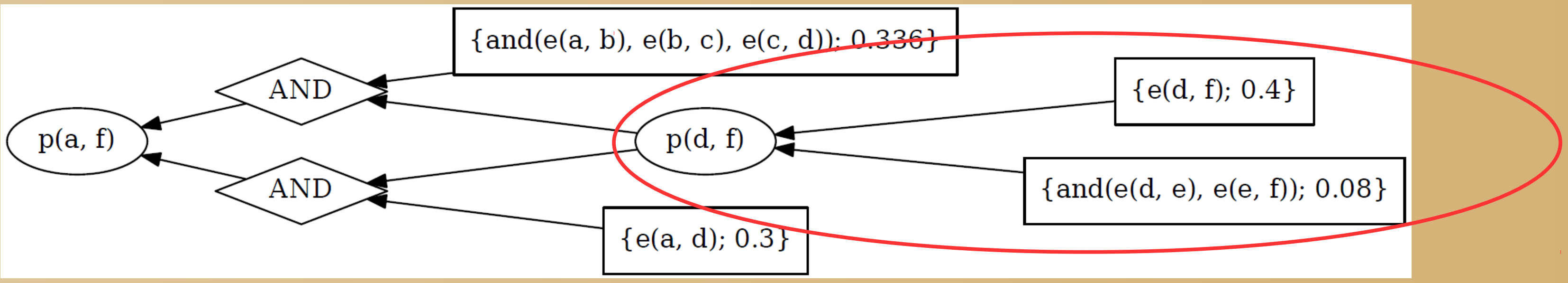


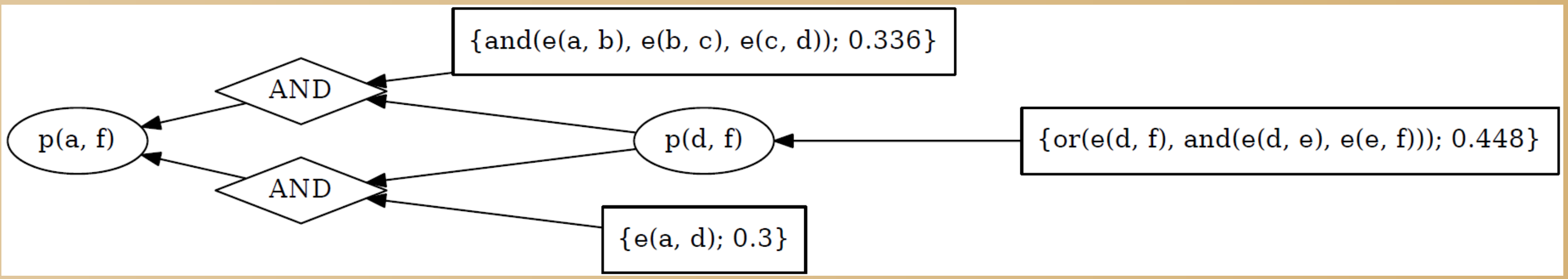
$$p_t = (\ldots((p_1 \cdot (1-p_2) + p_2) \cdot (1-p_3) + p_3) \ldots + p_n)$$



Detect Patterns

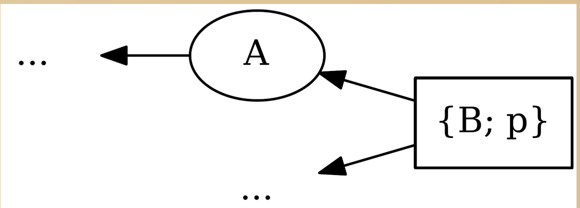
- Compaction

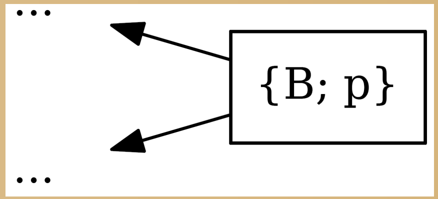


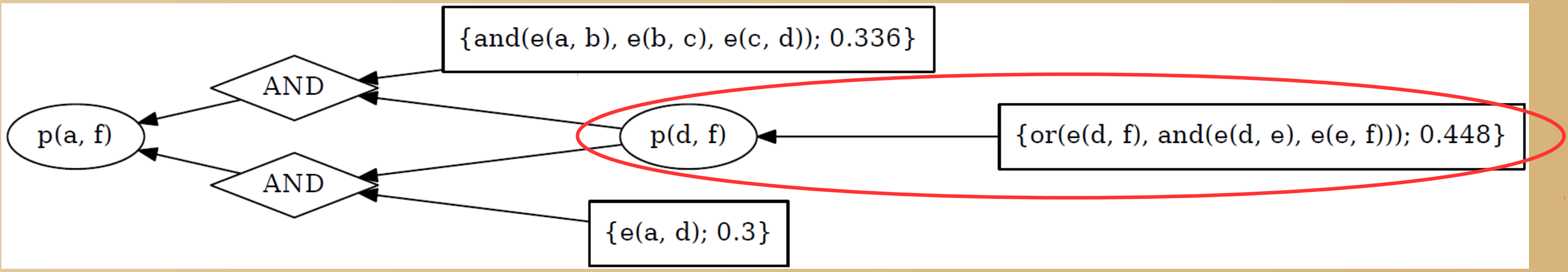


Detect Patterns

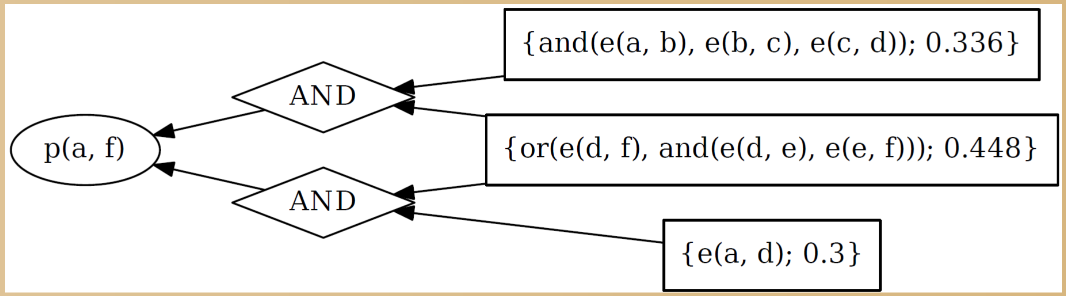
- Single Variable $O(N_{or} \cdot (\log(N_{or}) + \log(N_{and})))$ Compaction





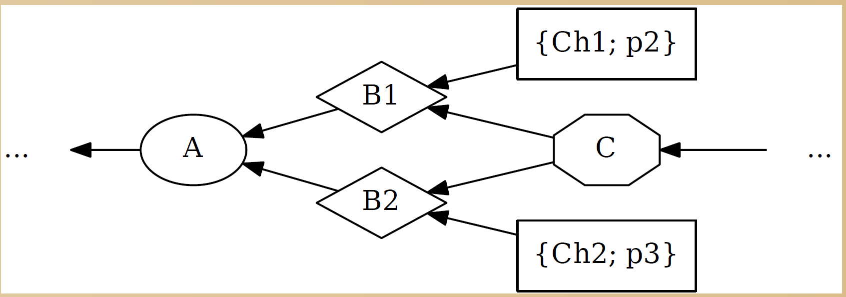


- Compaction

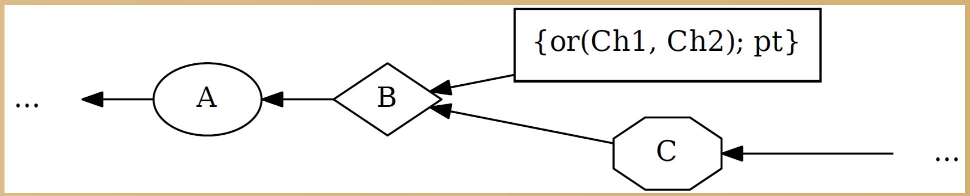


Detect Patterns

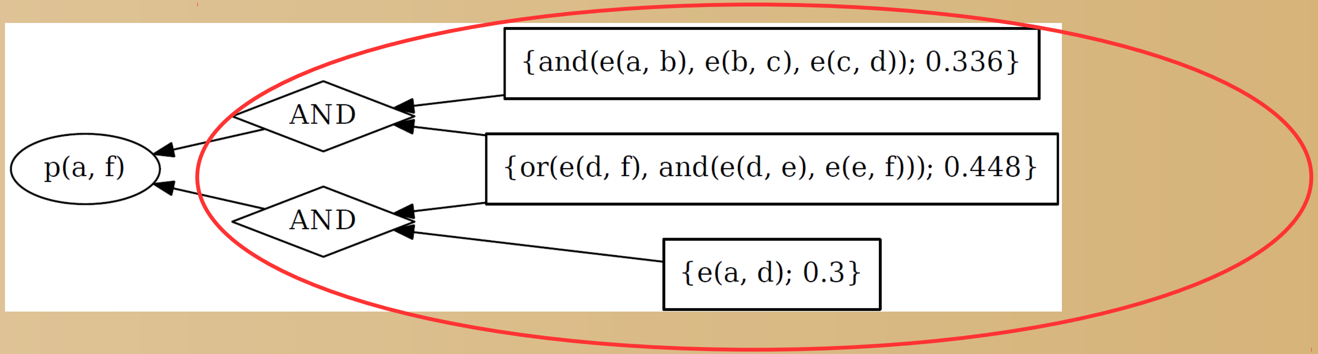
- OR-Cluster II *(not supported)*



Compaction

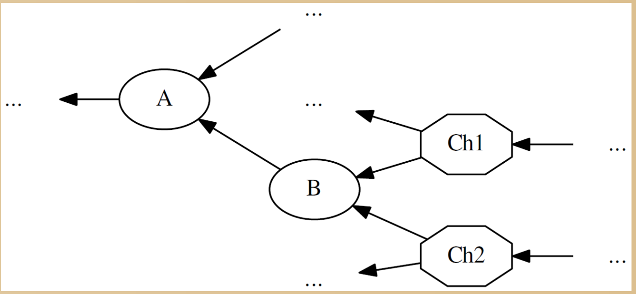


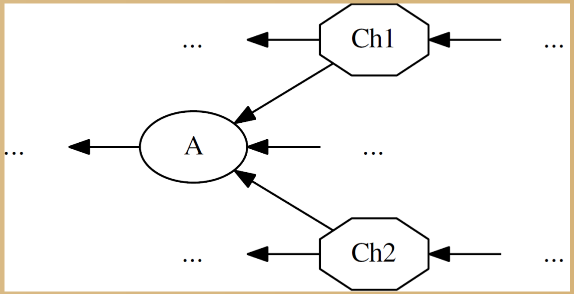
$$p_t = (\ldots((p_1 \cdot (1-p_2) + p_2) \cdot (1-p_3) + p_3) \ldots + p_n)$$



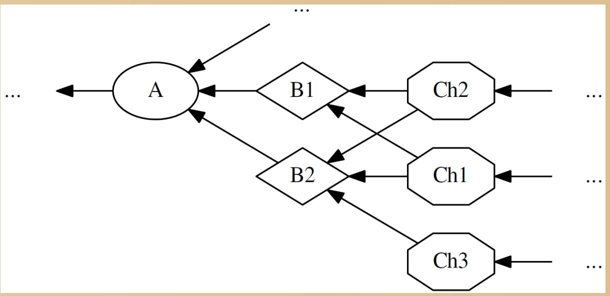
Detect Patterns

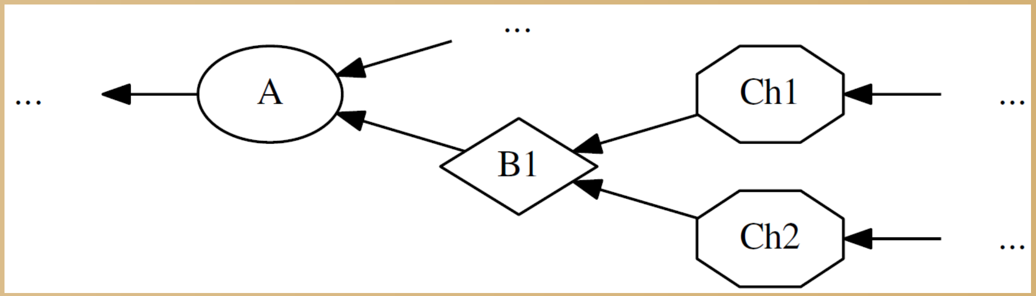
- Single Branch II $O(N_{or} \cdot (\log(N_{or}) + \log(N_{and})))$ Compaction





- Minimal Proof $O(N_{or} \cdot (\log(N_{or}) + \log(N_{and}) + N_{term}))$ Compaction

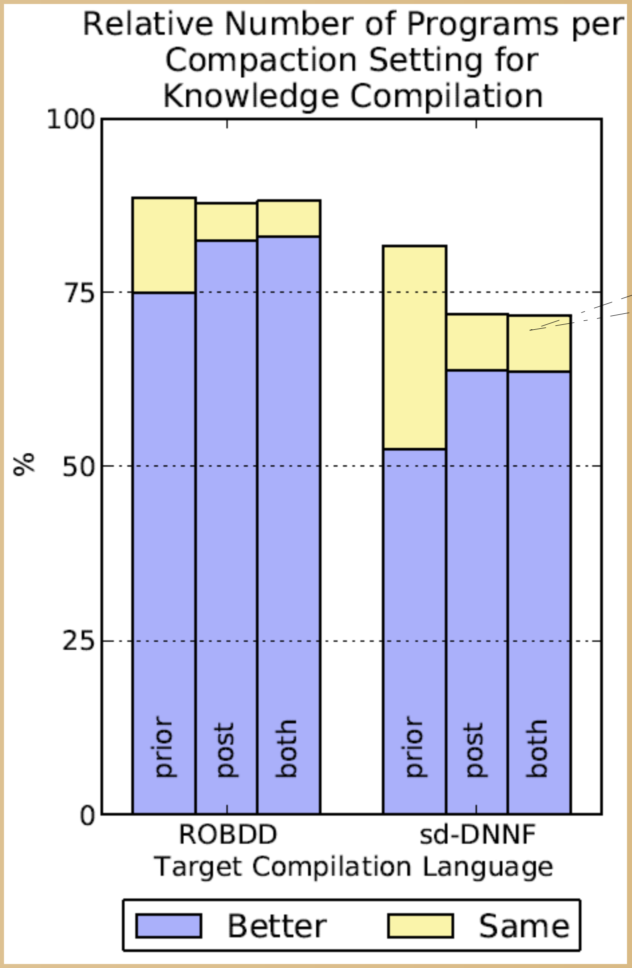




Experiments

- 2 PLP Systems: *MetaProbLog and ProbLog 2*
- 2 Compilation Languages: *ROBDDs, sd-DNNFs*
- *3 Different Compaction Settings: Prior, Post, Both*
- 7 Benchmark sets: *Alzheimer, Balls, Dictionary, Grid, Les Miserables, Smokers*, WebKB**
- Total of 738 ProbLog programs

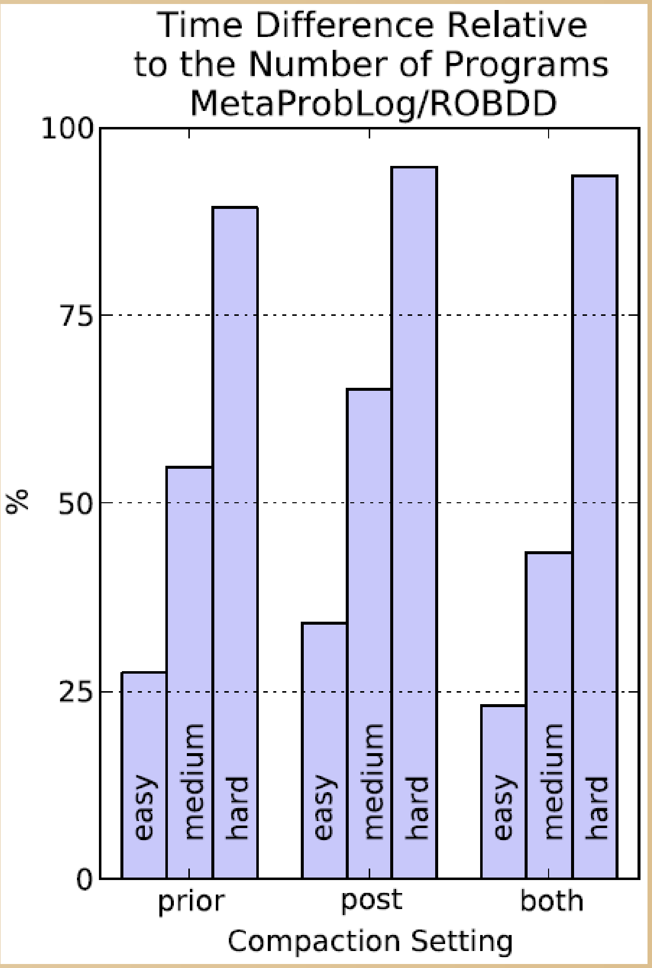
Results

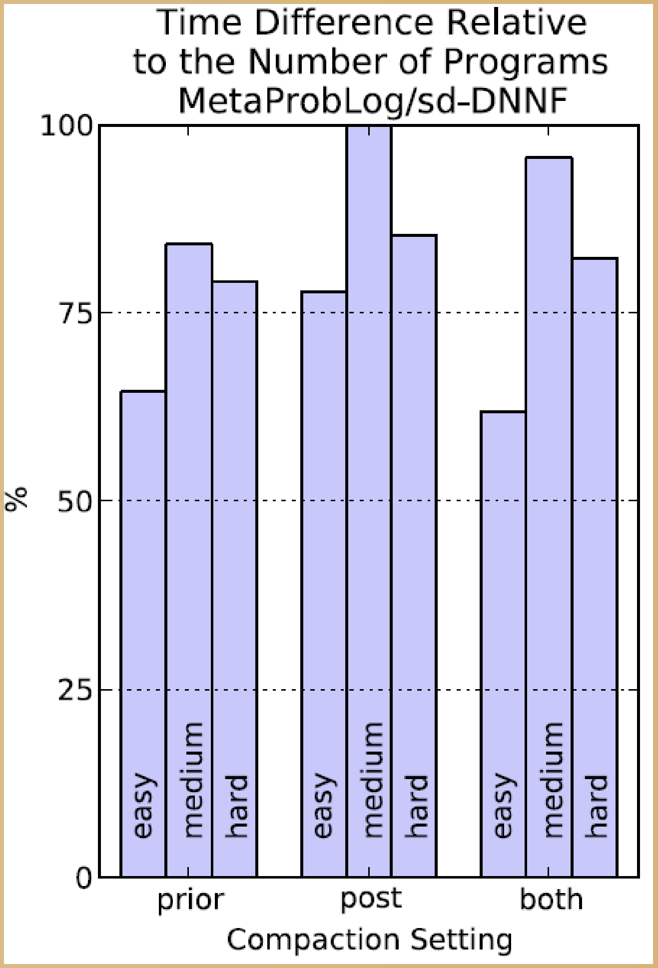


Relative Number of Programs per Compaction Setting for Knowledge Compilation
100
75
50
25
0
%
prior
post
both
prior
post
both
ROBDD
sd-DNNF
Target Compilation Language
Better
Same

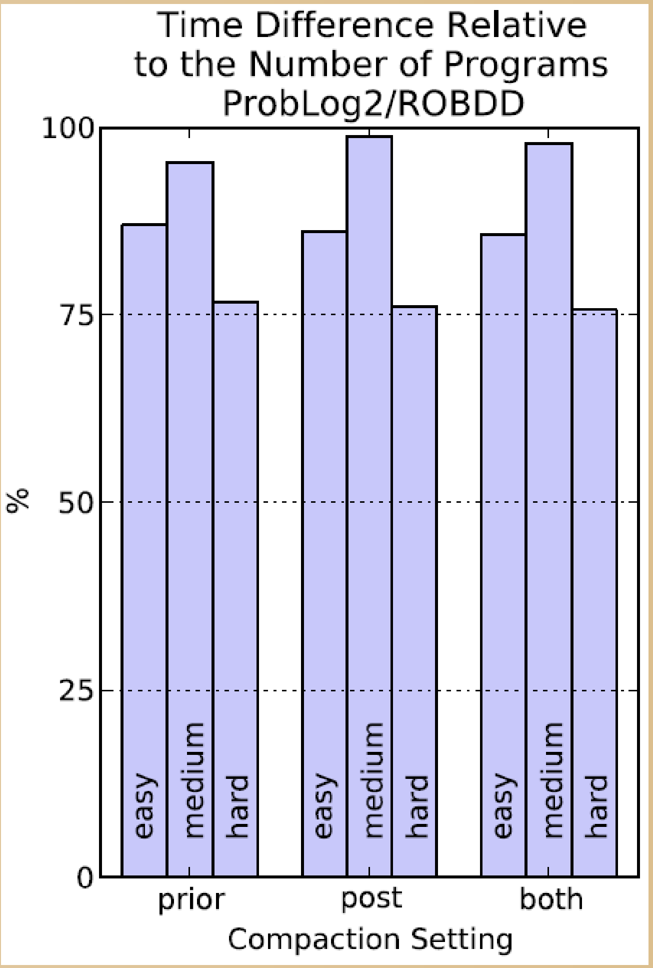
5%

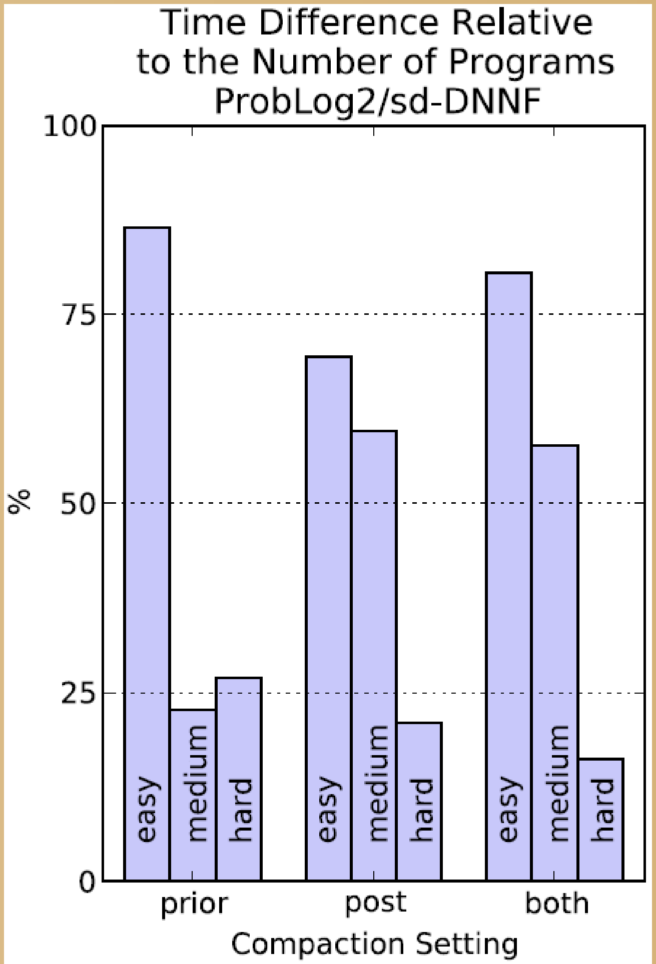
Results - MetaProbLog





Results ProbLog 2





Experiments - Timeouts

Benchmark	# Programs solved	No Compaction				Prior Compaction				Post Compaction				Both Compactions			
		MetaProbLog/ROBDD	MetaProbLog/sd-DNNF	ProbLog2/ROBDD	ProbLog2/sd-DNNF	MetaProbLog/ROBDD	MetaProbLog/sd-DNNF	ProbLog2/ROBDD	ProbLog2/sd-DNNF	MetaProbLog/ROBDD	MetaProbLog/sd-DNNF	ProbLog2/ROBDD	ProbLog2/sd-DNNF	MetaProbLog/ROBDD	MetaProbLog/sd-DNNF	ProbLog2/ROBDD	ProbLog2/sd-DNNF
alzheimer_q1	5	2	4	1	4	2	4	0	4	1	4	1	4	1	4	1	4
alzheimer_q2	9	4	6	0	5	3	6	0	5	3	6	0	5	3	6	0	6
alzheimer_q3	9	0	0	3	3	0	0	3	3	0	0	3	3	0	0	3	3
alzheimer_q4	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alzheimer_q5	9	3	3	1	5	3	3	3	5	1	3	3	4	0	3	3	4
alzheimer_q6	6	1	3	0	2	0	3	0	2	0	3	0	2	0	3	0	2
balls	150	0	0	35	0	0	0	13	0	0	0	0	0	0	0	0	0
dictionary	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
grid	12	4	0	1	0	4	2	1	1	4	1	1	1	4	1	1	1
les_miserables	47	2	10	2	27	2	14	2	24	2	16	3	26	2	21	2	28
smokers1	20	6	13	0	3	6	12	0	3	7	13	0	3	6	12	0	3
smokers2	20	5	16	1	3	5	16	0	3	5	16	1	3	5	16	1	3
smokers3	18	3	11	0	1	3	13	0	1	4	12	0	1	3	13	0	1
smokers4	20	5	13	1	3	3	14	1	3	5	14	1	4	3	15	1	4
smokers5	17	1	12	0	0	2	12	0	0	1	12	0	0	3	11	0	0
webkb1	48	18	29	9	0	10	14	3	0	8	14	3	0	8	14	4	3
webkb2	33	8	19	8	3	8	13	7	0	8	13	8	2	8	14	8	2
webkb4	48	20	41	17	0	7	41	1	12	1	41	1	14	0	41	1	10
webkb5	48	17	29	10	0	8	14	4	0	7	14	7	0	9	14	7	3
Total:	636	99	209	89	59	66	181	38	66	57	182	32	72	55	188	32	77
Average time-outs:		4.13	8.57	2.10	3.75	3.19	8.12	1.31	3.93	2.73	8.08	1.59	4.01	2.55	8.21	1.58	4.21

Experiments – MetaProbLog - Times

	Prior		Post		Both	
Benchmark	Min	Max	Min	Max	Min	Max
alzheimer_q1	1.04/2.66	2.0/17.51	0.07/0.58	17.69/139.28	1.11/2.68	20.38/141.88
alzheimer_q2	1.13/2.54	4.17/331.08	0.02/0.14	7.82/308.4	1.15/2.54	11.3/259.23
alzheimer_q3	0.05/0.13	5.51/321.67	0.0/0.03	0.0/281.88	0.05/0.13	5.54/302.82
alzheimer_q4	0.0/0.03	0.16/0.54	0.0/0.02	0.0/0.18	0.0/0.03	0.16/0.52
alzheimer_q5	0.05/0.14	0.38/158.32	0.01/0.04	0.24/230.54	0.05/0.14	3.15/438.45
alzheimer_q6	1.13/2.53	4.15/327.9	0.02/0.14	7.79/345.25	1.17/2.57	11.21/240.1
balls	0.0/0.02	5.99/78.89	0.0/0.02	6.07/78.4	0.0/0.02	8.77/84.84
dictionary	0.0/0.02	0.01/1.22	0.0/0.02	0.01/1.87	0.0/0.02	0.02/1.66
grid	0.0/0.02	0.02/271.24	0.0/0.02	0.03/110.92	0.0/0.02	0.09/149.19
les_miserables	0.0/0.02	0.0/178.72	0.0/0.02	0.26/95.19	0.0/0.02	0.26/94.95
smokers1	0.0/0.02	0.01/436.57	0.0/0.02	3.68/108.58	0.0/0.02	38.45/404.6
smokers2	0.0/0.02	0.01/147.54	0.0/0.02	29.98/201.61	0.0/0.02	64.95/297.79
smokers3	0.0/0.02	0.01/64.96	0.0/0.03	8.02/98.45	0.0/0.03	25.36/117.75
smokers4	0.0/0.03	0.01/368.1	0.0/0.03	20.93/306.63	0.0/0.03	21.8/352.89
smokers5	0.0/0.02	0.01/311.78	0.0/0.02	7.87/354.27	0.0/0.02	0.02/140.88
webkb1	0.01/0.06	1.94/370.12	0.01/0.05	3.93/259.48	0.01/0.05	4.72/479.17
webkb2	0.0/0.03	0.17/411.62	0.0/0.03	1.35/453.89	0.0/0.03	0.64/423.95
webkb4	0.0/0.03	2.77/466.59	0.0/0.03	8.46/101.19	0.0/0.03	15.47/174.75
webkb5	0.01/0.08	1.97/413.38	0.01/0.06	9.66/443.91	0.01/0.07	5.34/195.45

Experiments – ProbLog 2 - Times

	Prior		Post		Both	
Benchmark	Min	Max	Min	Max	Min	Max
alzheimer_q1	1.32/2.08	3.94/519.63	0.19/0.88	9.42/23.69	1.5/2.02	11.87/24.19
alzheimer_q2	1.42/1.72	7.95/451.17	0.09/0.39	10.29/257.59	1.52/1.74	18.25/264.27
alzheimer_q3	0.1/0.17	3.93/268.92	0.05/0.16	0.26/382.5	0.14/0.21	3.98/269.76
alzheimer_q4	0.05/0.1	0.24/0.46	0.04/0.13	0.04/0.3	0.09/0.14	0.28/0.48
alzheimer_q5	0.11/0.24	0.5/267.79	0.05/0.2	1.87/113.28	0.16/0.26	2.37/113.61
alzheimer_q6	1.4/1.68	3.32/92.3	0.1/0.4	5.78/131.44	1.48/1.71	10.79/136.35
balls	0.04/0.08	3.83/93.48	0.04/0.12	6.96/76.37	0.09/0.13	7.48/77.24
dictionary	0.04/0.08	0.05/1.4	0.04/0.13	0.04/1.09	0.09/0.13	0.09/1.1
grid	0.06/0.15	0.11/107.21	0.05/0.21	0.15/100.54	0.11/0.22	0.21/438.91
les_miserables	0.04/0.08	0.08/534.84	0.04/0.12	0.42/222.54	0.08/0.13	1.57/369.71
smokers1	0.03/0.07	0.05/173.61	0.03/0.1	0.09/133.69	0.06/0.1	0.13/202.9
smokers2	0.04/0.08	0.06/538.9	0.04/0.13	0.25/531.27	0.08/0.13	0.28/531.63
smokers3	0.04/0.1	0.06/190.54	0.04/0.15	0.58/301.3	0.09/0.15	0.59/299.9
smokers4	0.04/0.08	0.05/360.08	0.04/0.13	0.6/464.78	0.08/0.13	0.58/462.04
smokers5	0.04/0.1	0.06/74.4	0.04/0.14	0.08/26.18	0.09/0.14	0.14/21.59
webkb1	0.05/0.79	1.11/340.08	0.05/0.86	1.25/414.36	0.09/0.84	1.45/359.86
webkb2	0.04/0.73	0.29/237.8	0.04/0.78	2.04/92.82	0.08/0.77	0.85/103.56
webkb4	0.05/0.74	0.64/242.11	0.04/0.81	0.39/377.47	0.09/0.79	1.28/486.33
webkb5	0.05/0.79	0.72/434.88	0.05/0.85	1.28/295.33	0.1/0.85	1.43/317.09

Conclusion

- Compaction of 6 out 7 Patterns
- Performance Gain in most cases
- Implementation in Prolog (Future work)
- Test on other fields (Future work)

Questions

Thank You