

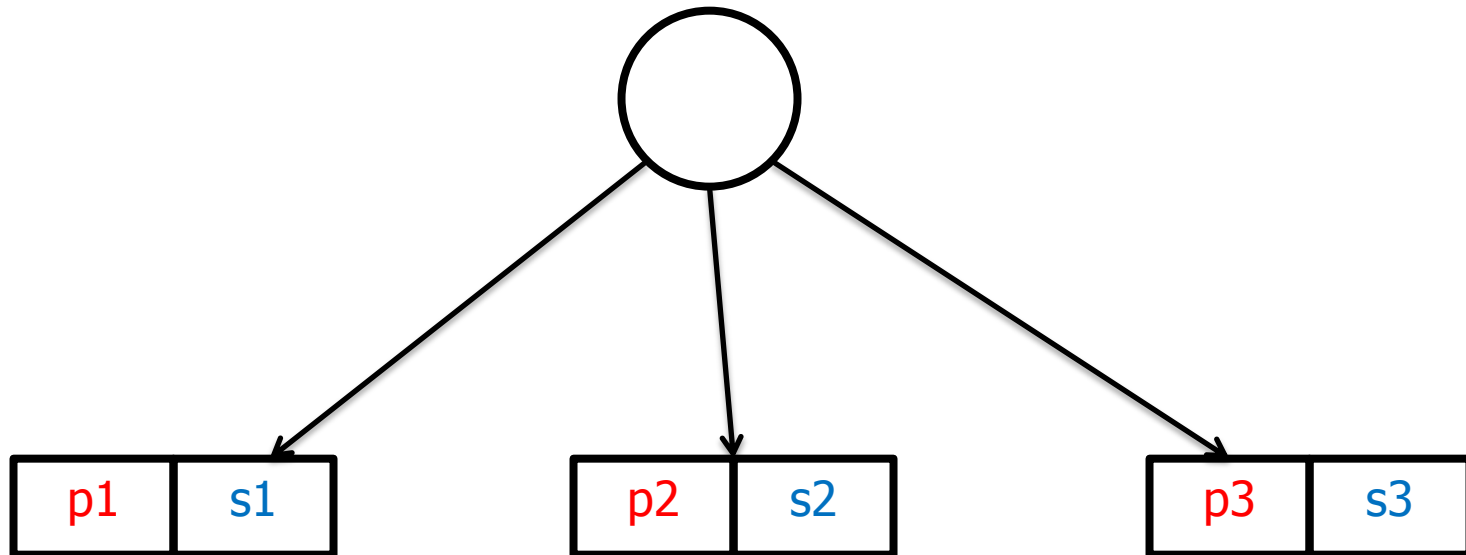
Sentential Decision Diagrams and their Applications

Guy Van den Broeck,
Arthur Choi, and Adnan Darwiche

UCLA

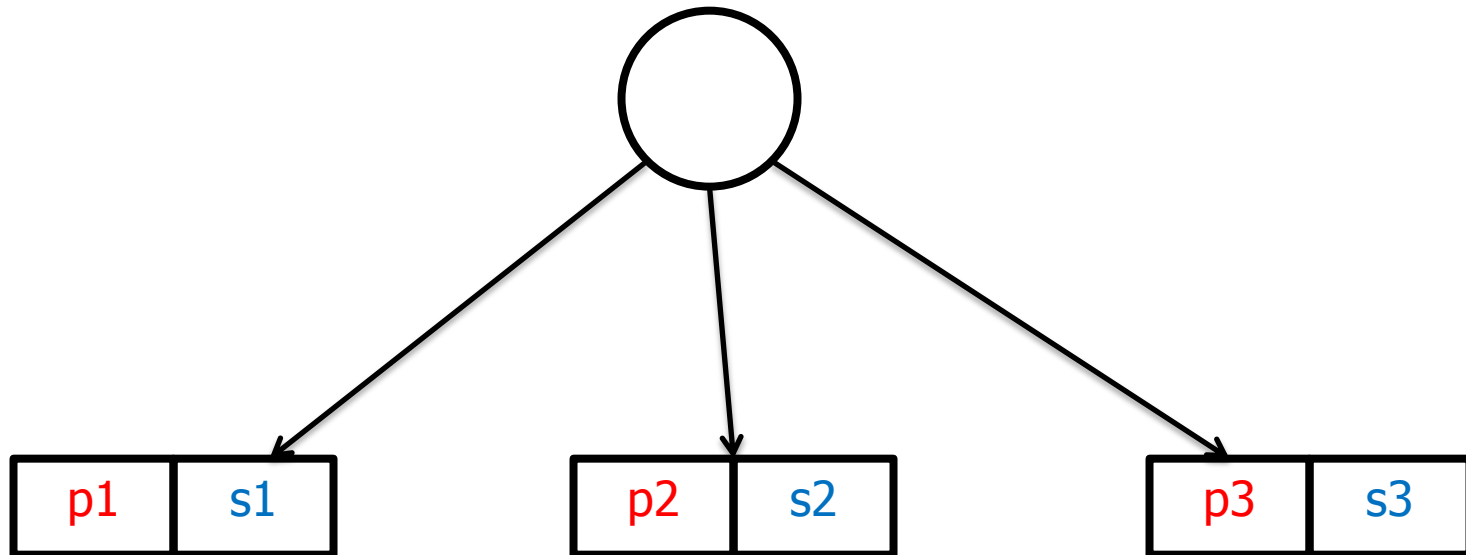
Basing Decisions on Sentences

US Senate: 54 Rep., 44 Dem., and 2 Indep.



Basing Decisions on Sentences

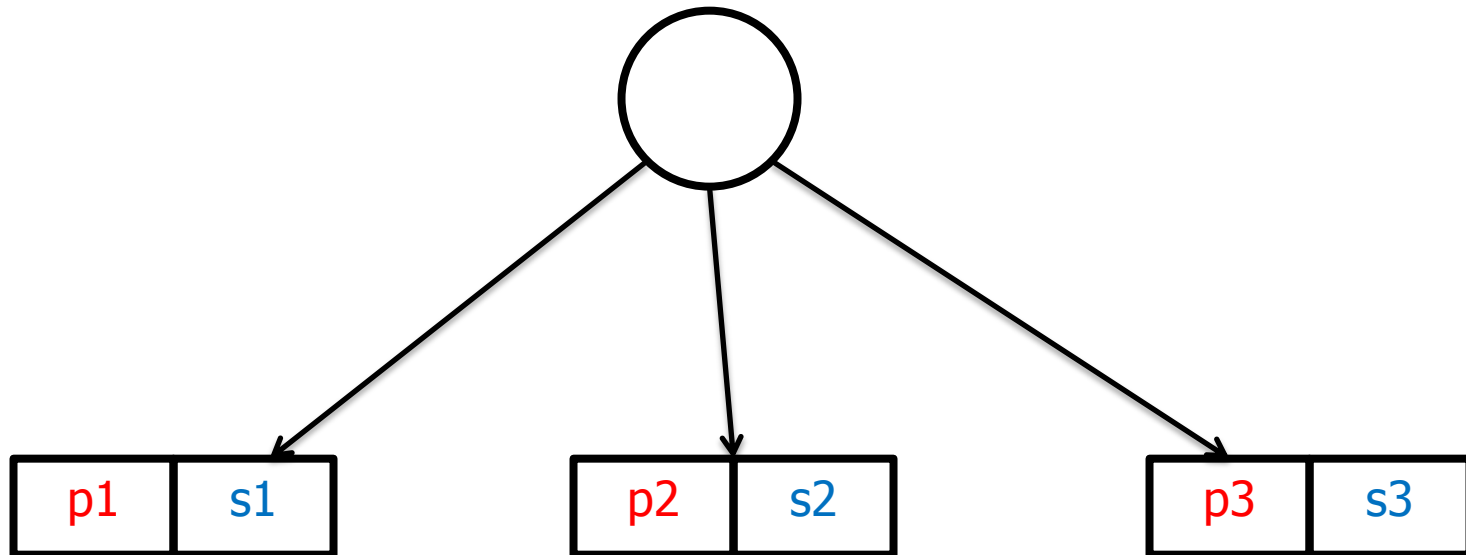
US Senate: 54 Rep., 44 Dem., and 2 Indep.



> 50
Rep.

Basing Decisions on Sentences

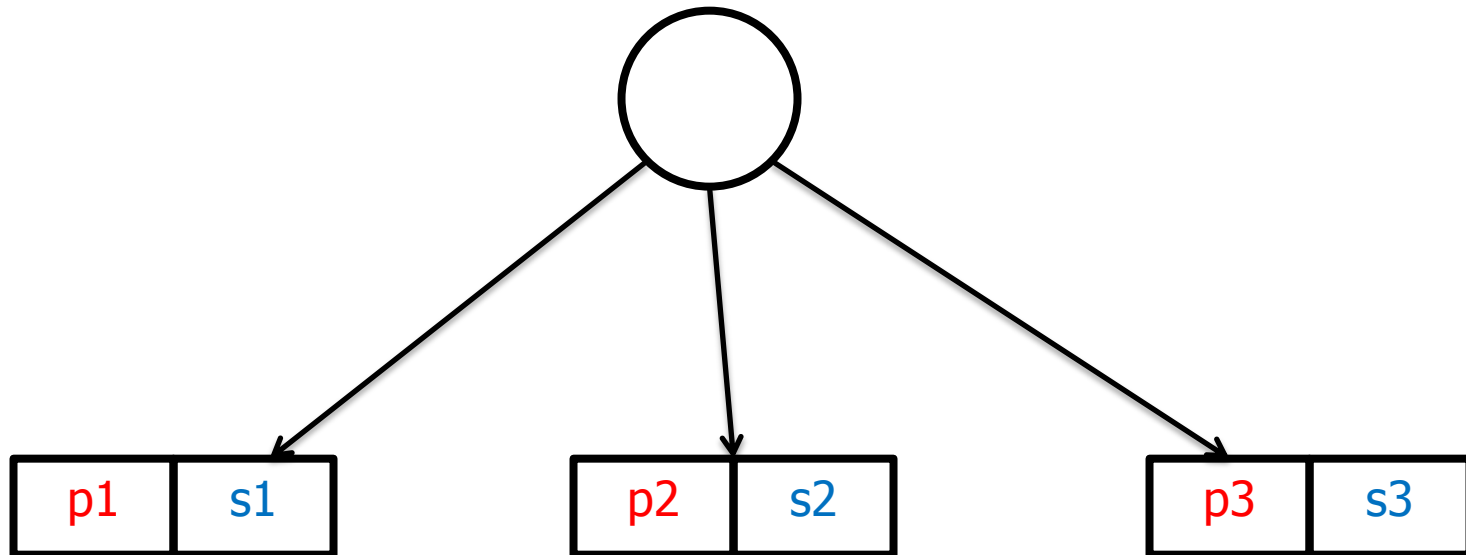
US Senate: 54 Rep., 44 Dem., and 2 Indep.



> 50 Veto
Rep.

Basing Decisions on Sentences

US Senate: 54 Rep., 44 Dem., and 2 Indep.

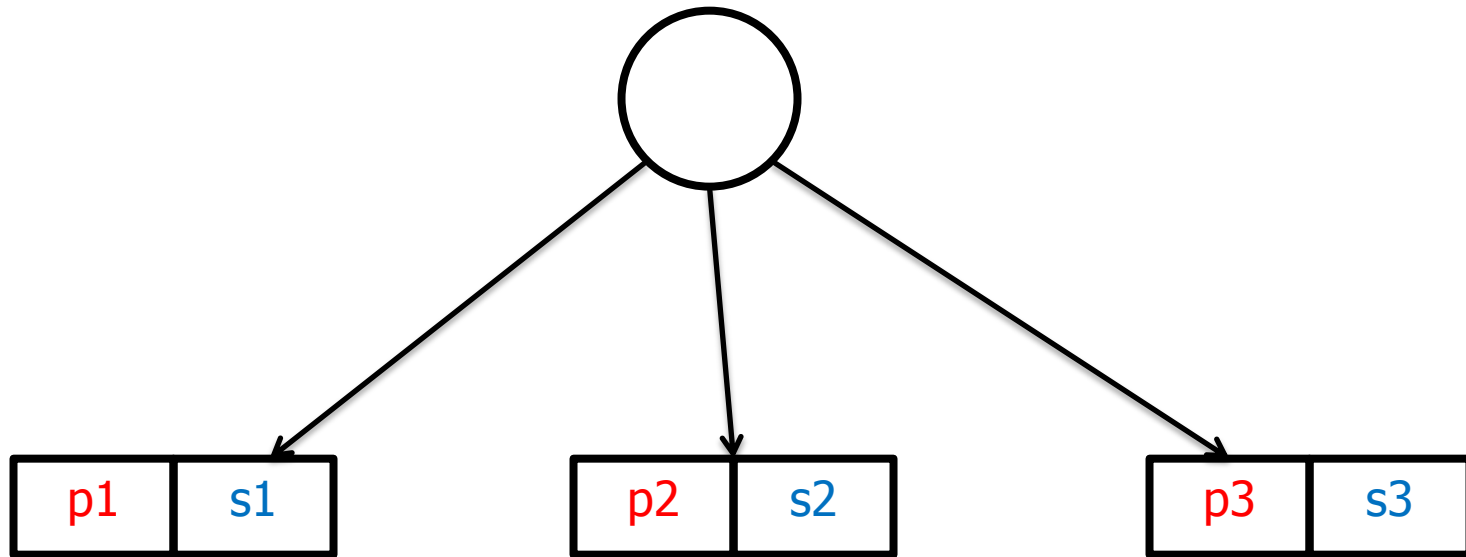


> 50 Veto
Rep.

48-50
Rep.

Basing Decisions on Sentences

US Senate: 54 Rep., 44 Dem., and 2 Indep.

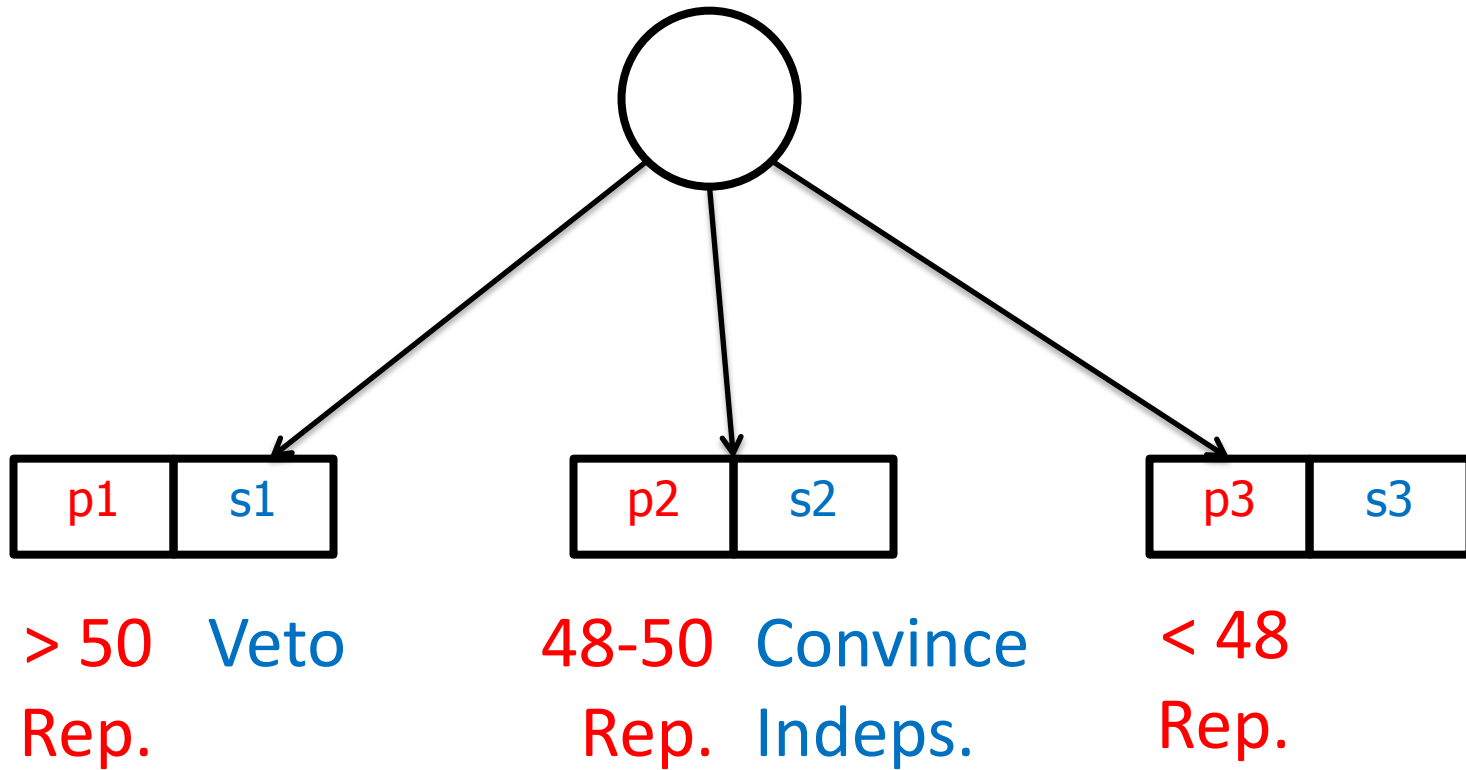


> 50 Veto
Rep.

48-50 Convince
Rep. Indeps.

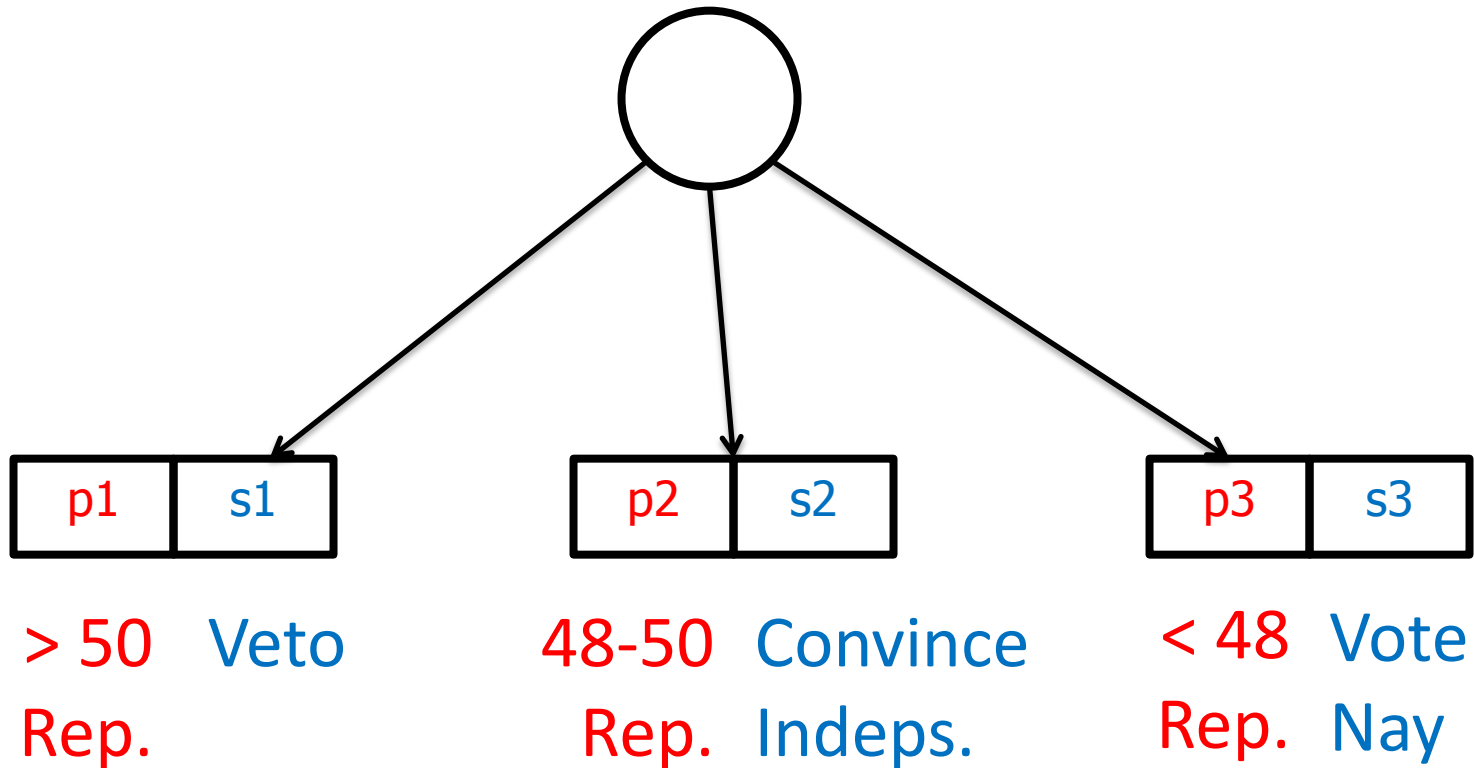
Basing Decisions on Sentences

US Senate: 54 Rep., 44 Dem., and 2 Indep.

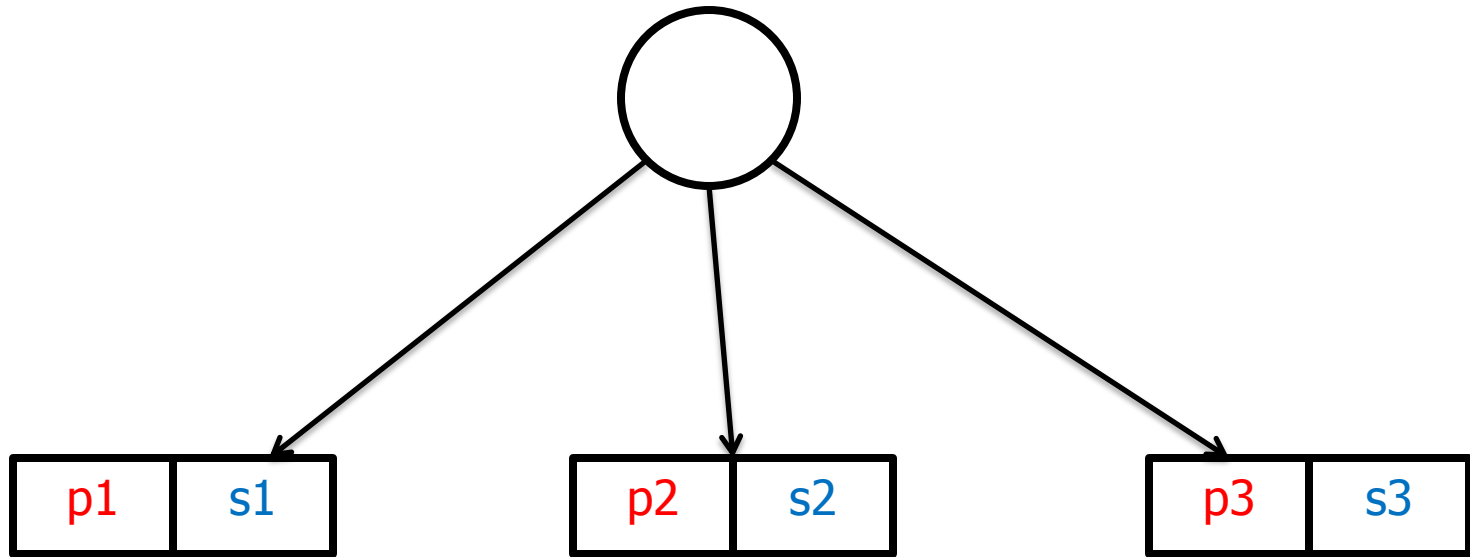


Basing Decisions on Sentences

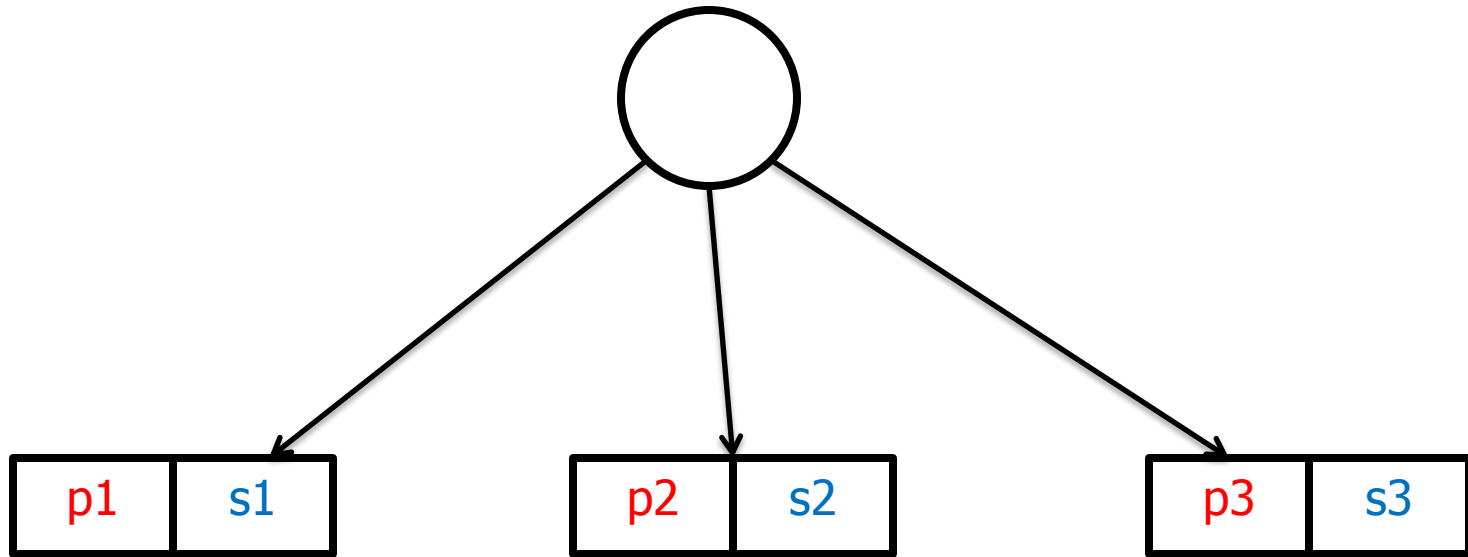
US Senate: 54 Rep., 44 Dem., and 2 Indep.



Basing Decisions on Sentences

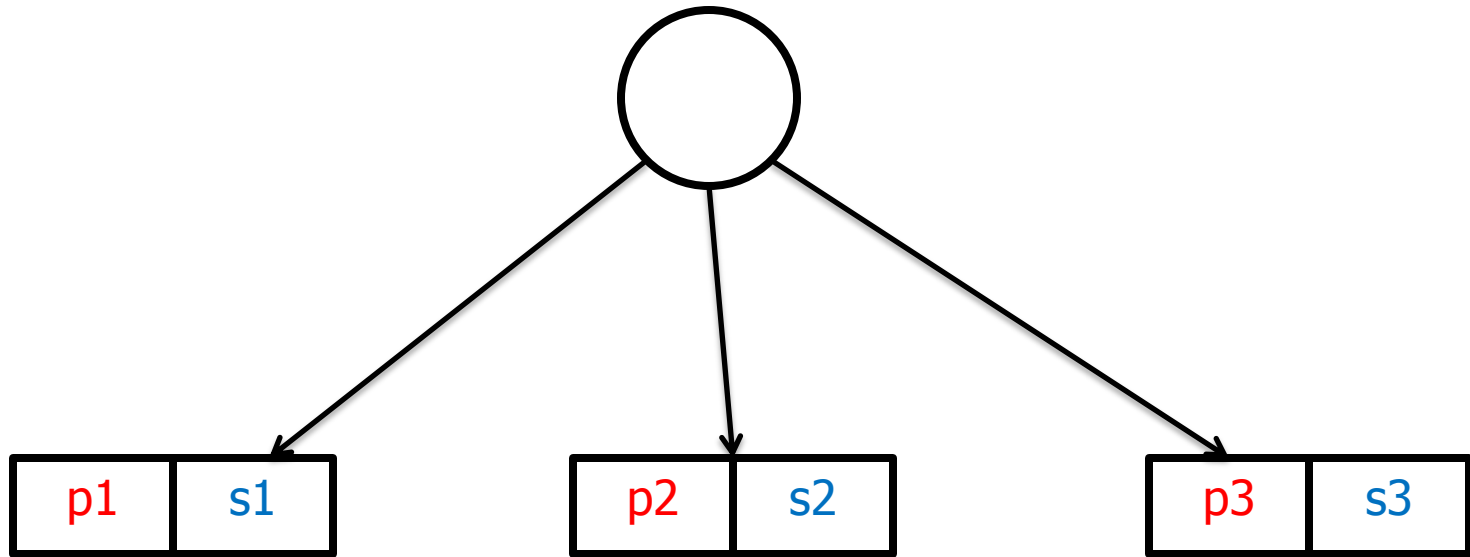


Basing Decisions on Sentences



Branch on **sentences** **p1**, **p2**, and **p3**:

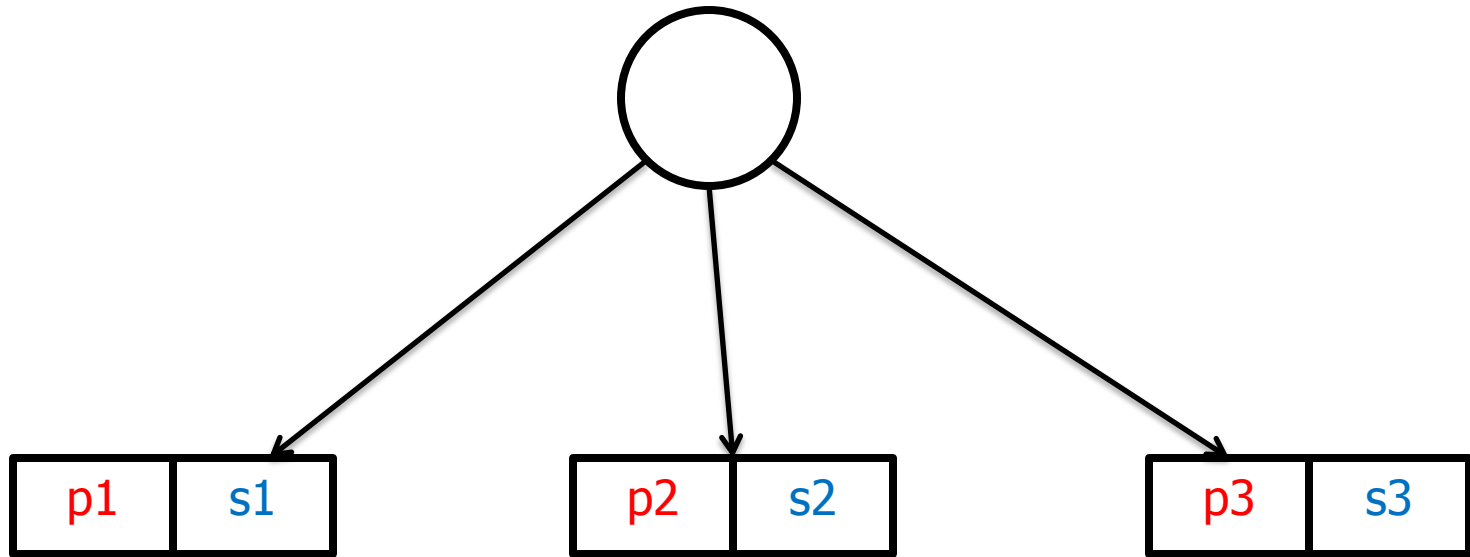
Basing Decisions on Sentences



Branch on **sentences** p_1 , p_2 , and p_3 :

- p_1 , p_2 , p_3 are **mutually exclusive, exhaustive** and not false

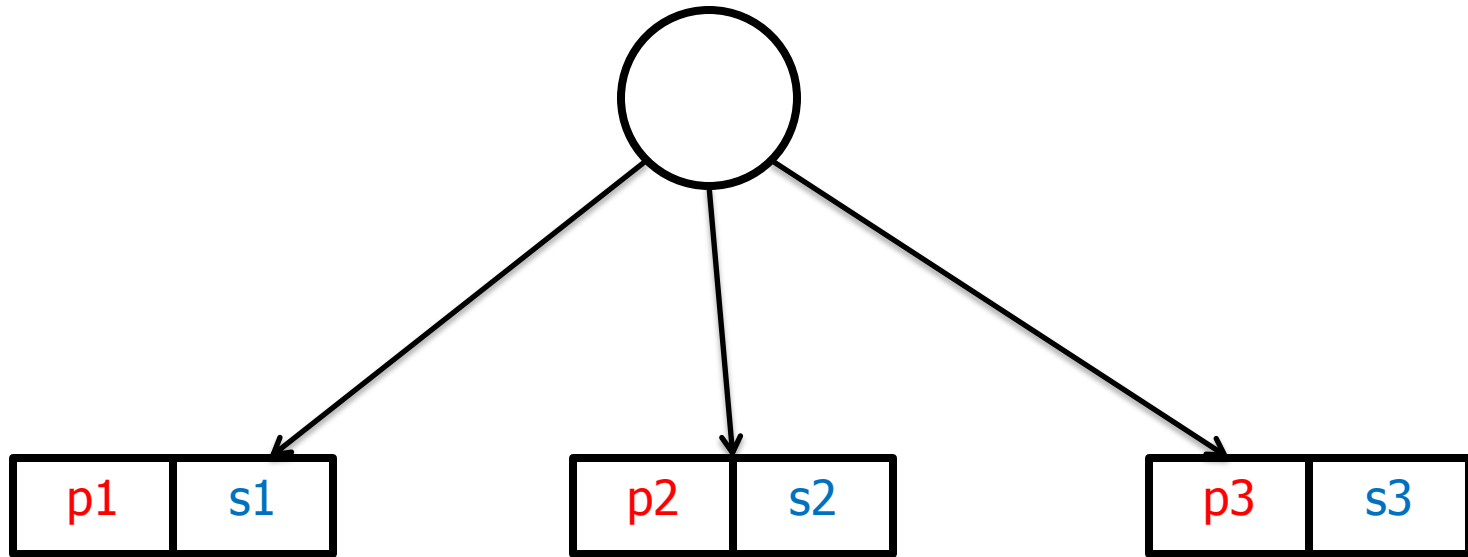
Basing Decisions on Sentences



Branch on **sentences** p_1 , p_2 , and p_3 :

- p_1 , p_2 , p_3 are **mutually exclusive, exhaustive** and not false
- p_1 , p_2 , p_3 are called **primes** and represented by SDDs

Basing Decisions on Sentences

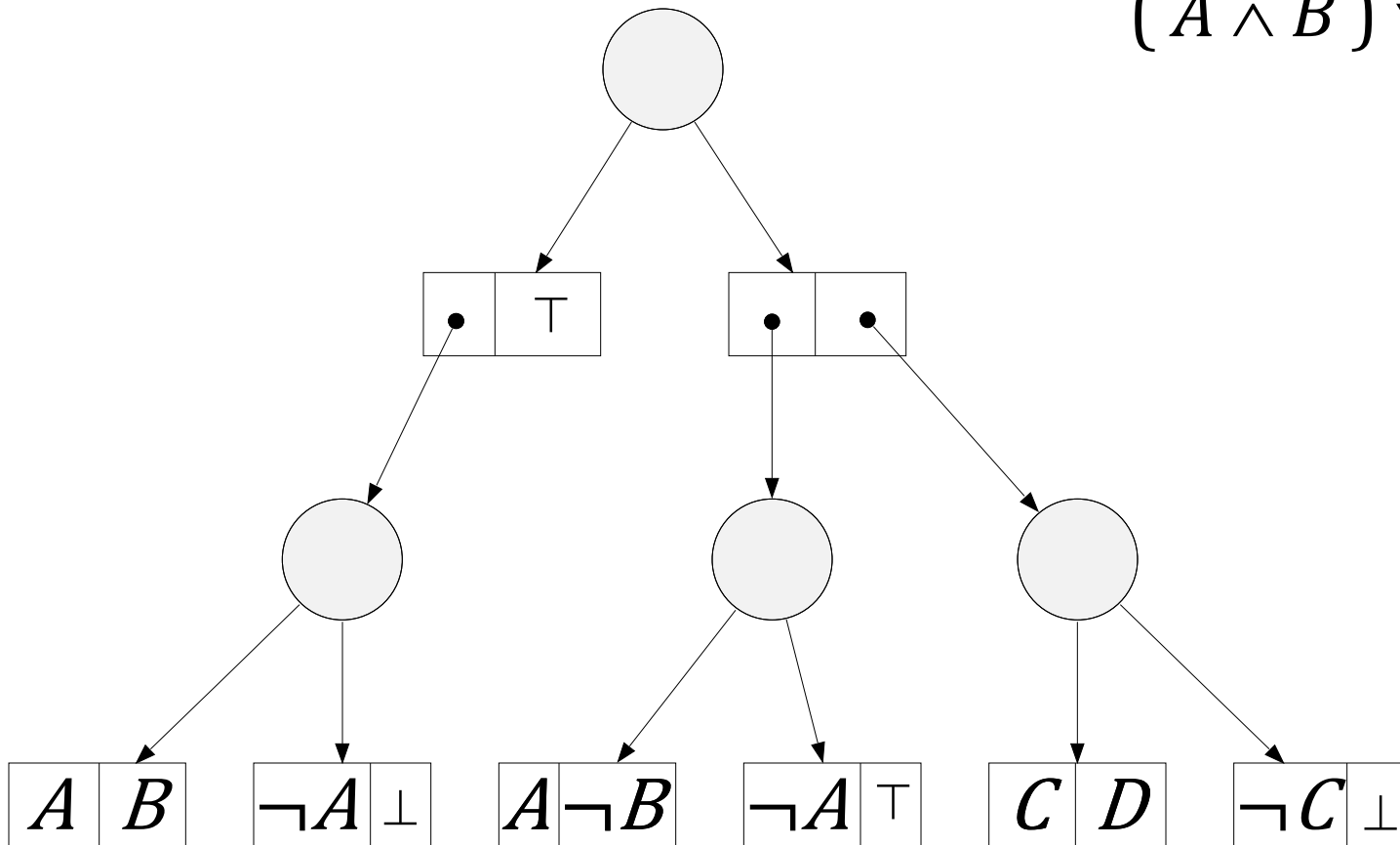


Branch on **sentences** $p1$, $p2$, and $p3$:

- $p1$, $p2$, $p3$ are **mutually exclusive, exhaustive** and not false
- $p1$, $p2$, $p3$ are called **primes** and represented by SDDs
- $s1$, $s2$, $s3$ are called **subs** and represented by SDDs

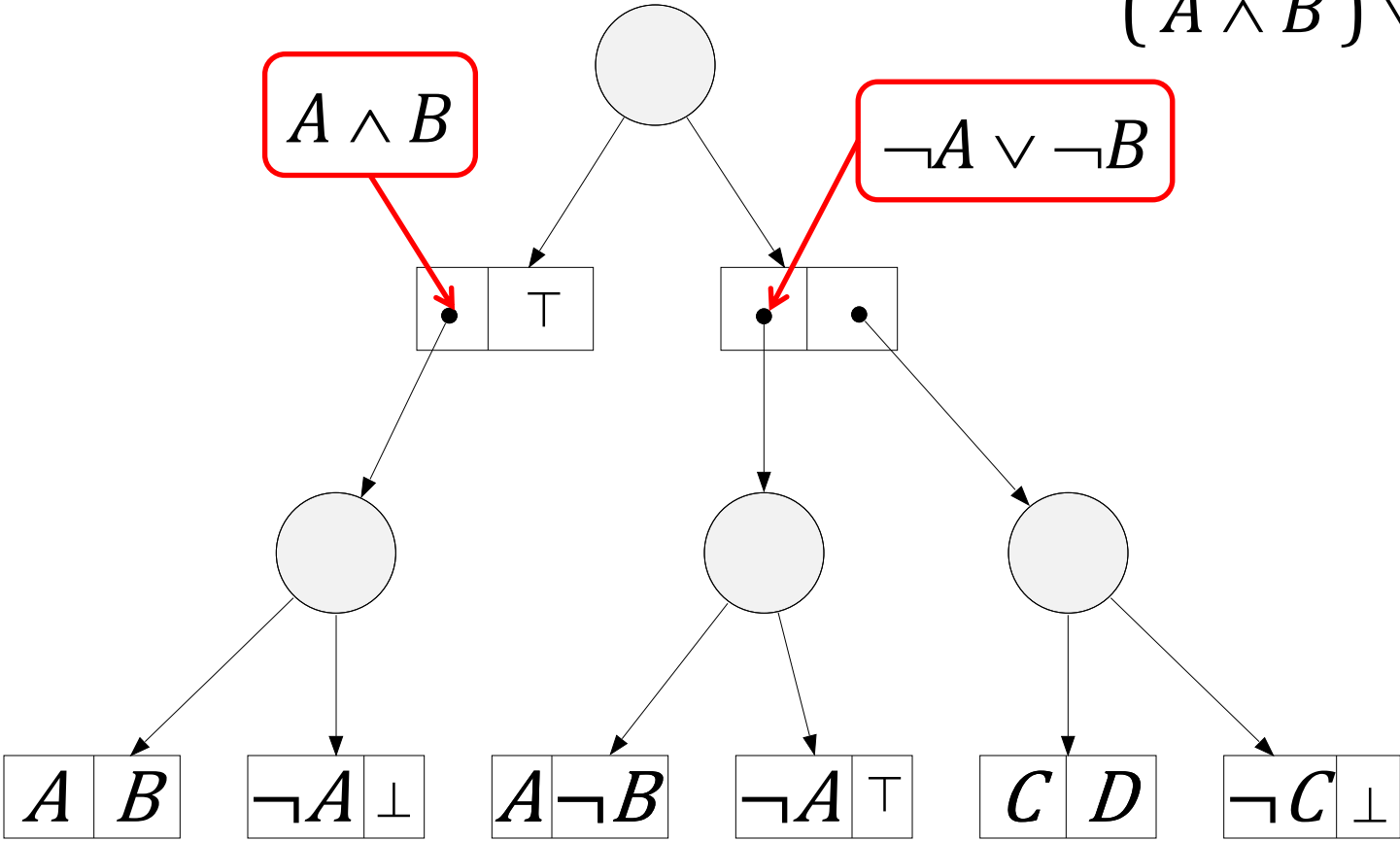
Basing Decisions on Sentences

$$f(A, B, C, D) = (A \wedge B) \vee (C \wedge D)$$



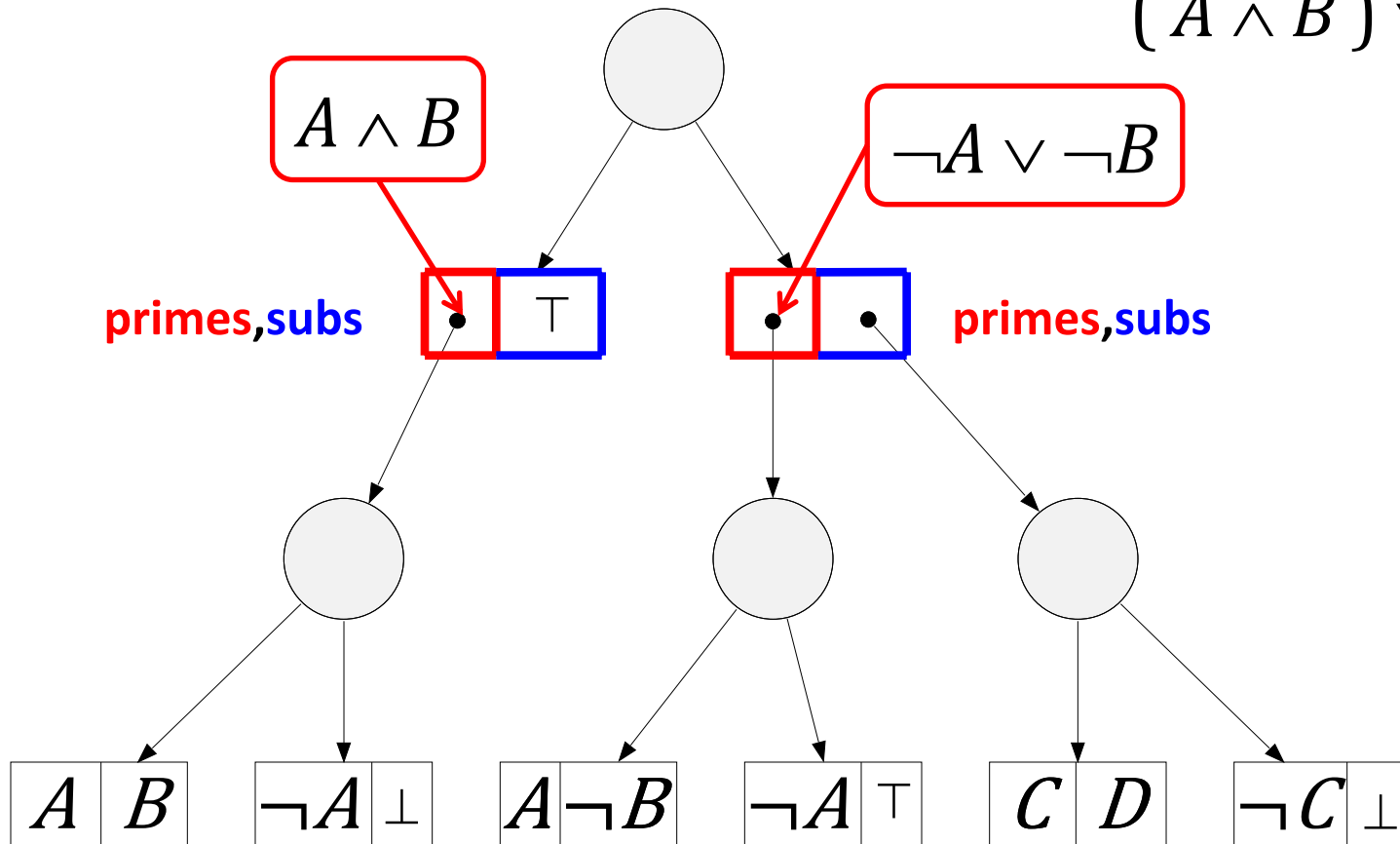
Basing Decisions on Sentences

$$f(A, B, C, D) = (A \wedge B) \vee (C \wedge D)$$



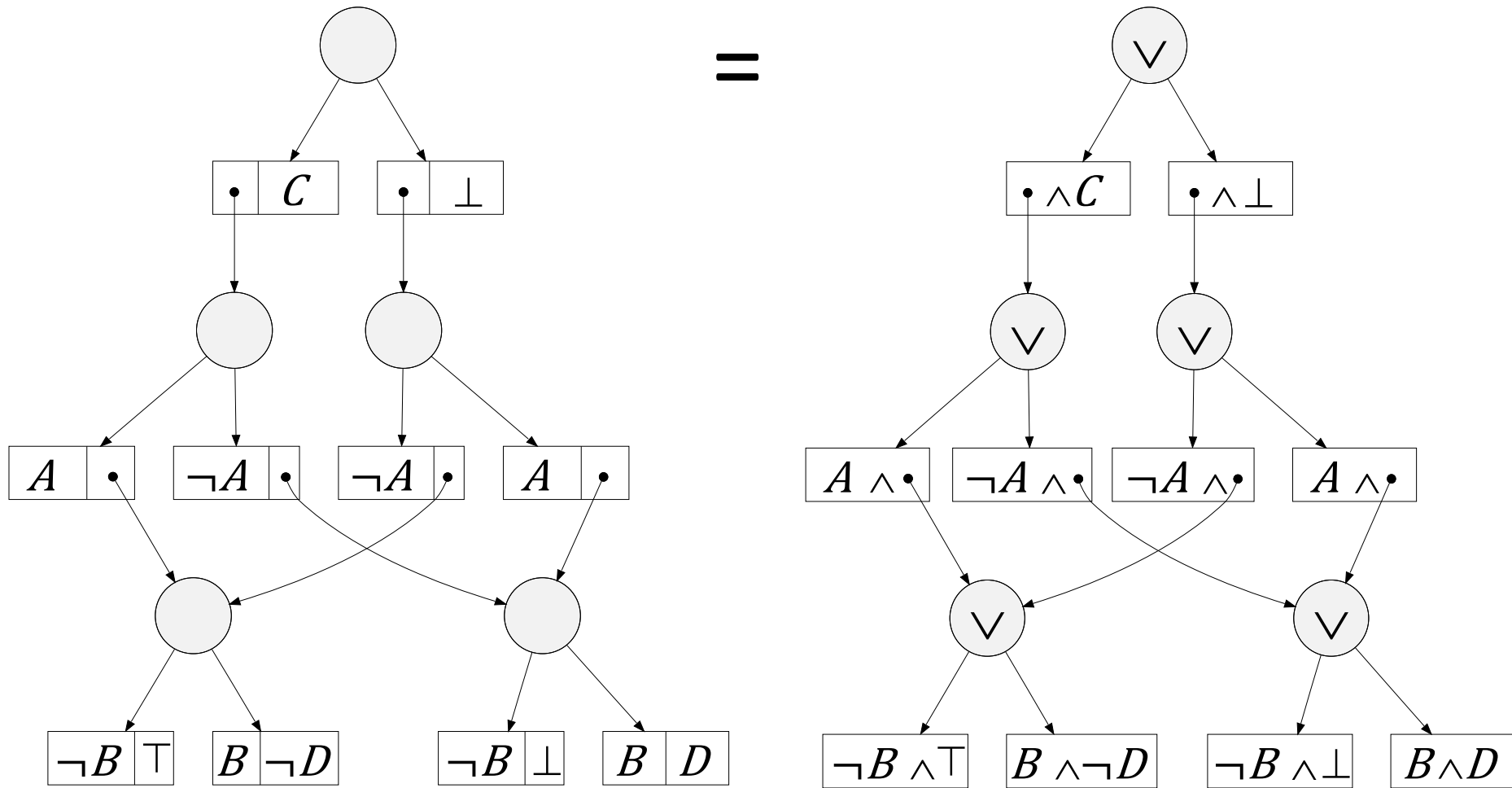
Basing Decisions on Sentences

$$f(A, B, C, D) = (A \wedge B) \vee (C \wedge D)$$



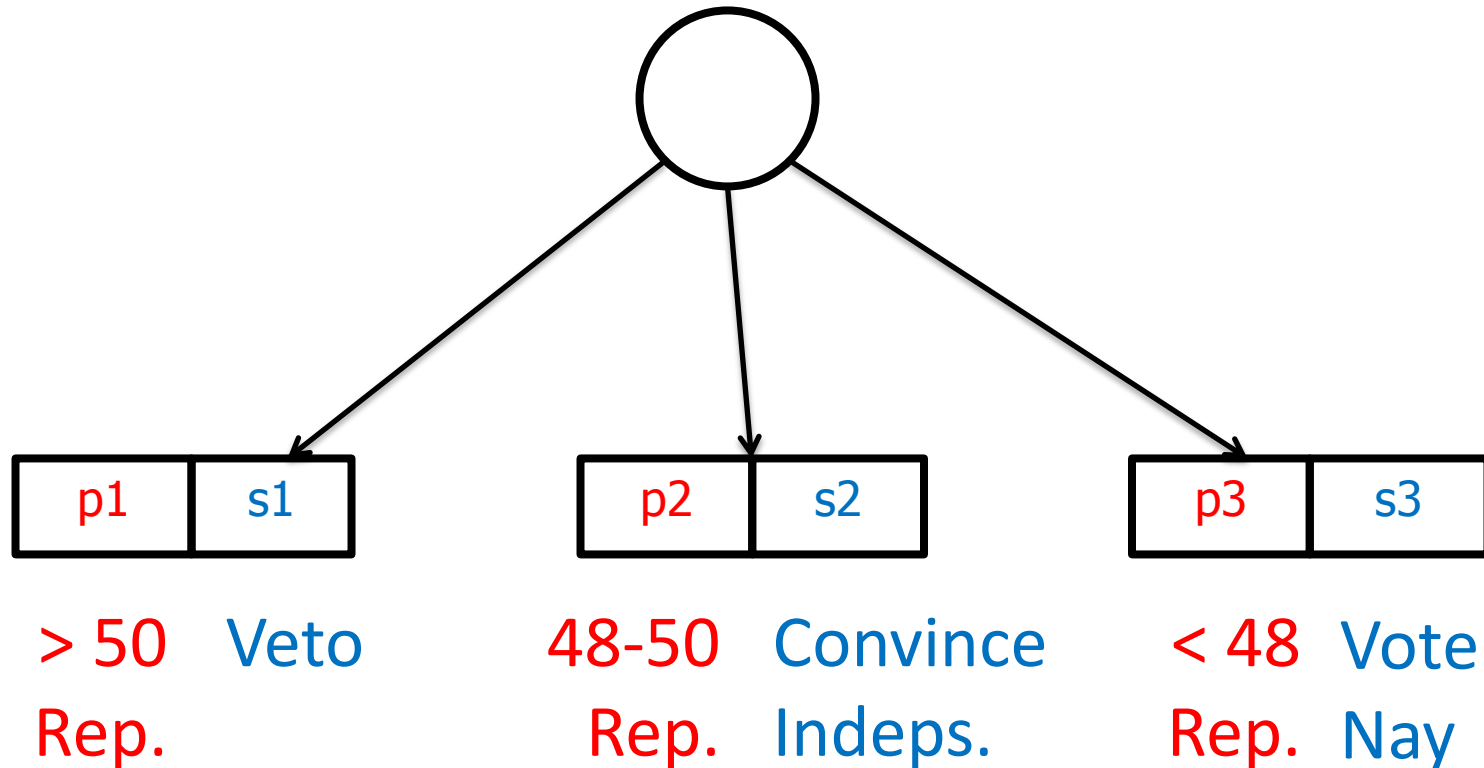
SDDs as Boolean Circuits

$$f(A, B, C, D) (A \oplus (B \wedge D)) \wedge C$$



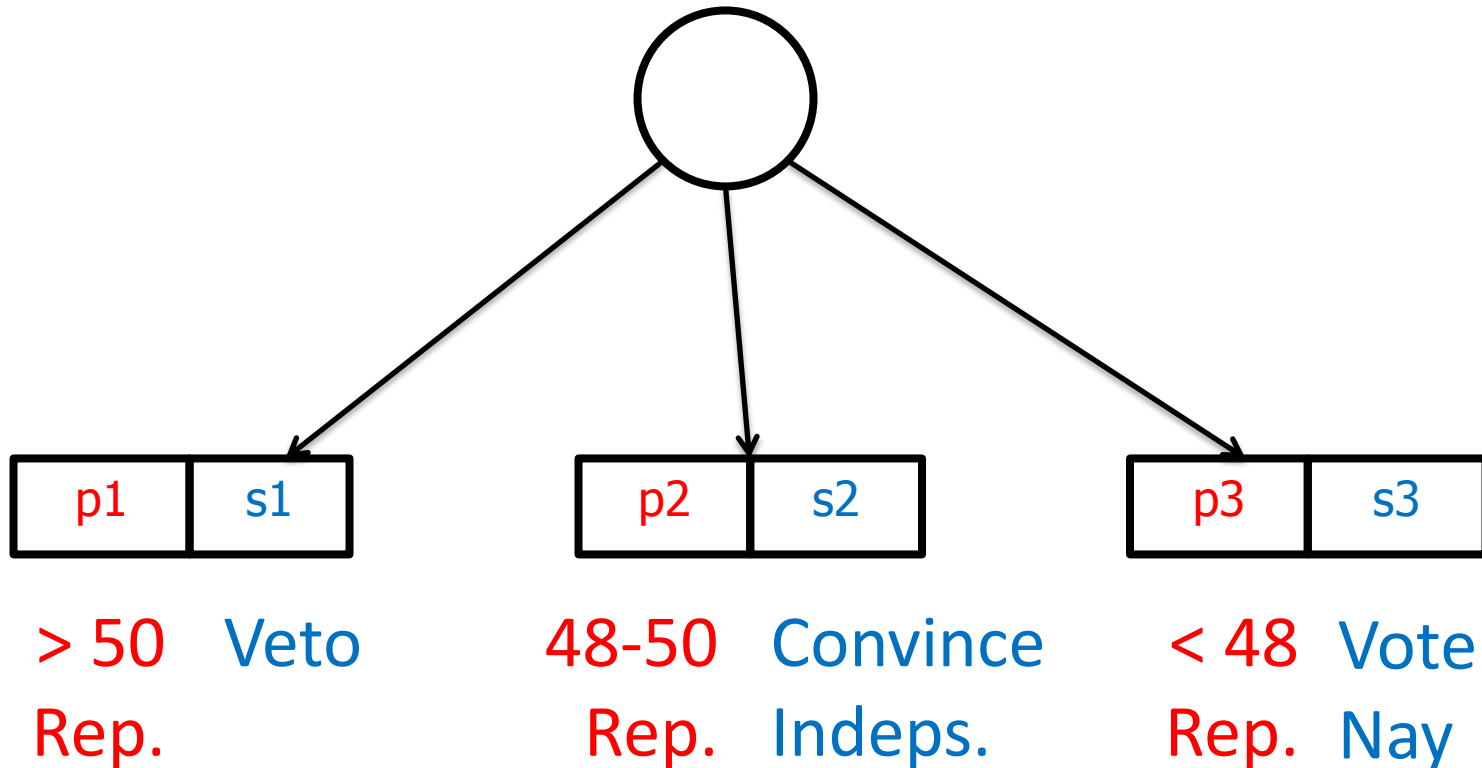
(X,Y)-Partitions

US Senate: 54 Rep., 44 Dem., and 2 Indep.



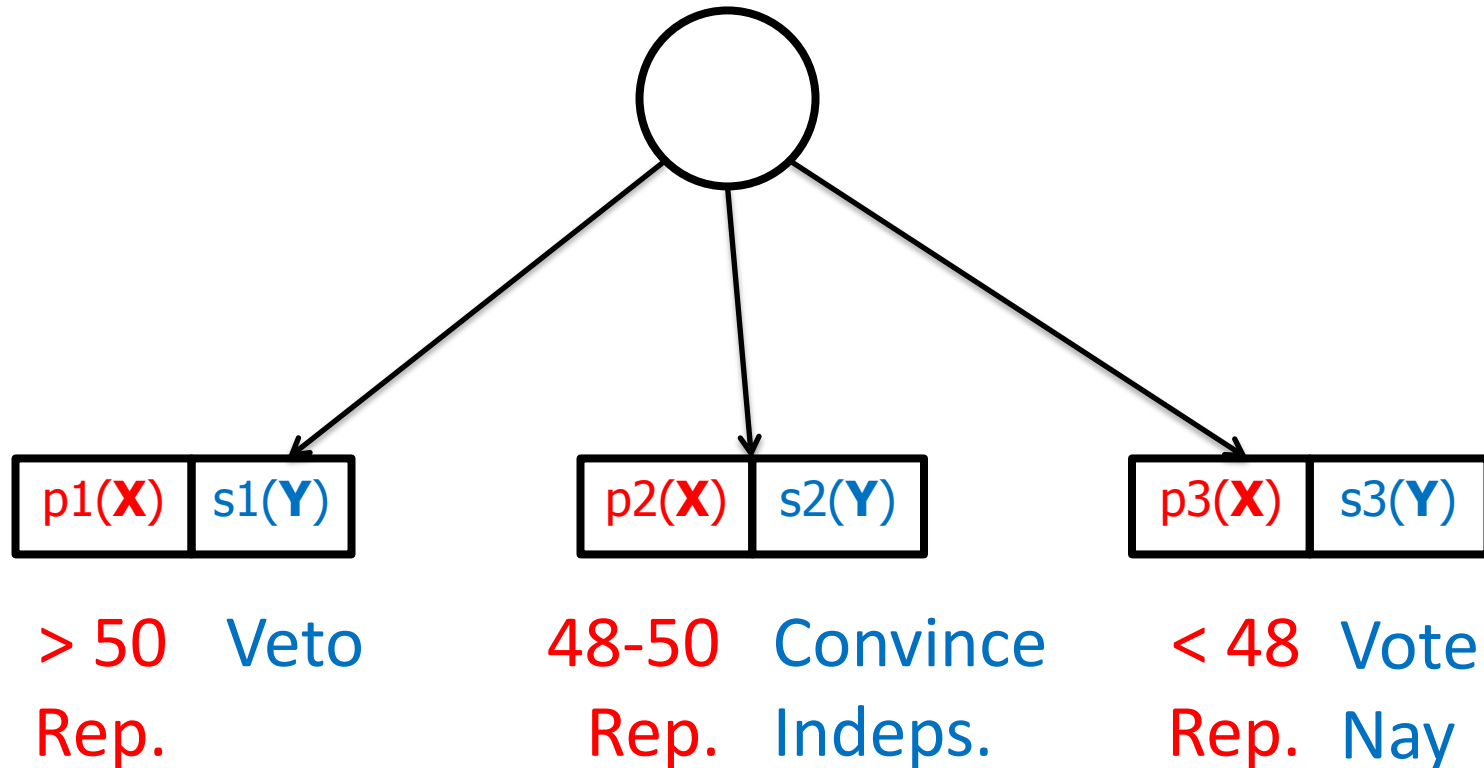
(X,Y)-Partitions

US Senate: 54 Rep., 44 Dem., and 2 Indep.



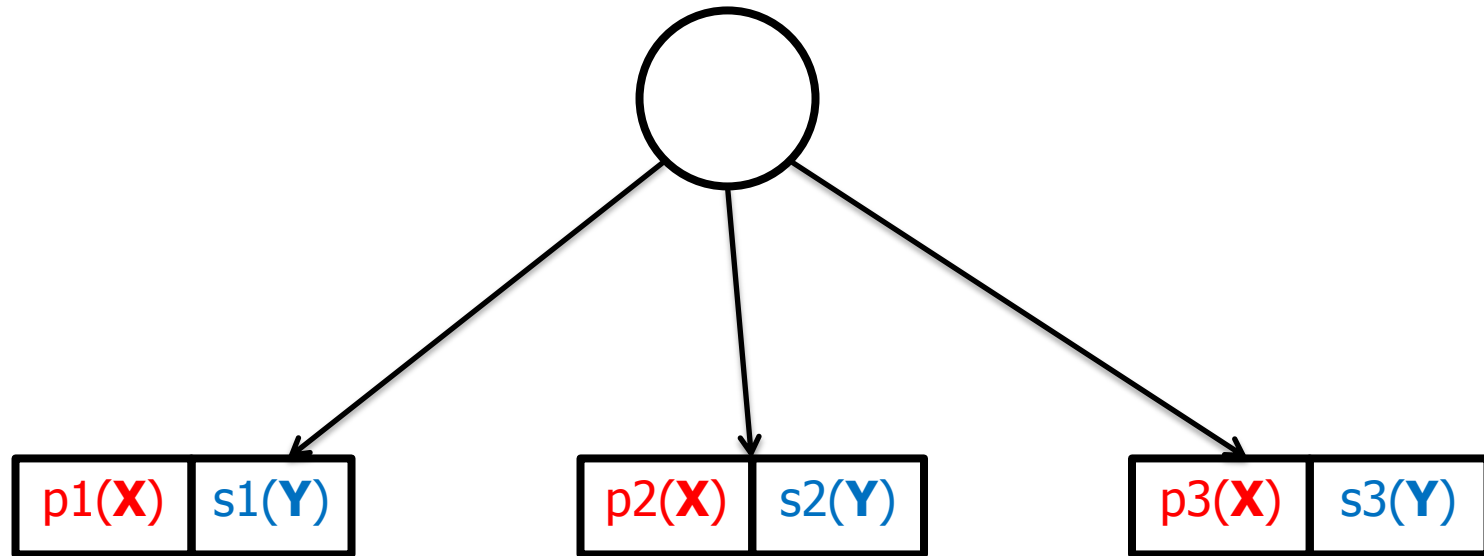
(X,Y)-Partitions

US Senate: 54 Rep., 44 Dem., and 2 Indep.



(**X**, **Y**)-Partitions

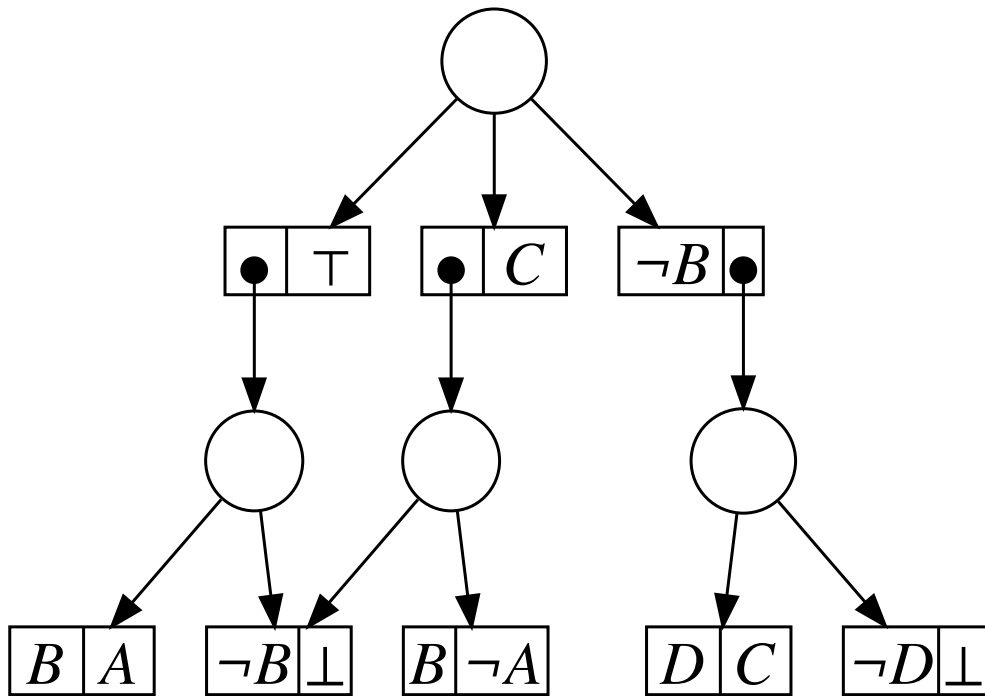
US Senate: **54 Rep.**, **44 Dem.**, and 2 Indep.



$$f(\mathbf{X}, \mathbf{Y}) = p_1(\mathbf{X}) s_1(\mathbf{Y}) \vee \dots \vee p_n(\mathbf{X}) s_n(\mathbf{Y})$$

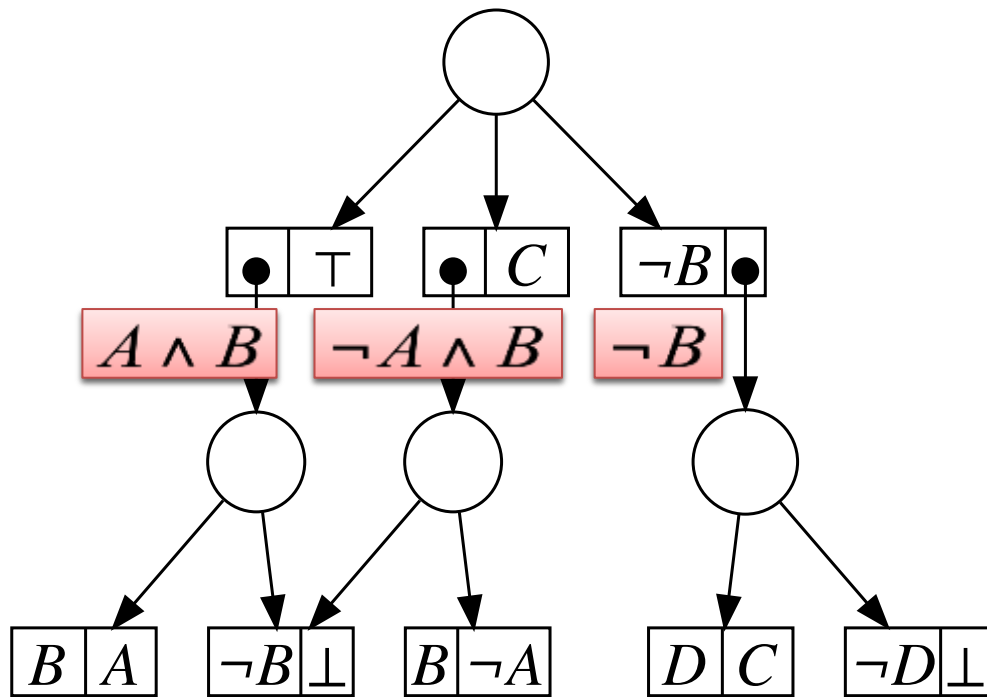
Variable **order** becomes variable **tree** (vtree)

$$f = (A \wedge B) \vee (B \wedge C) \vee (C \wedge D)$$



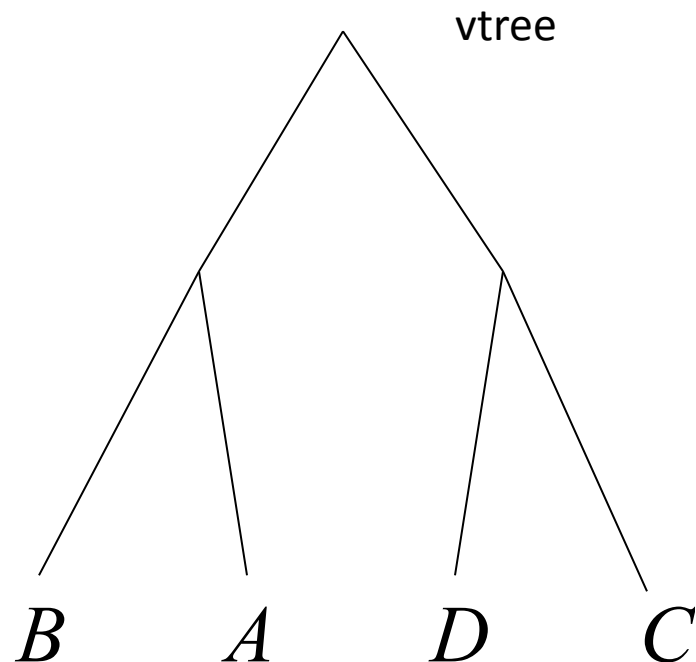
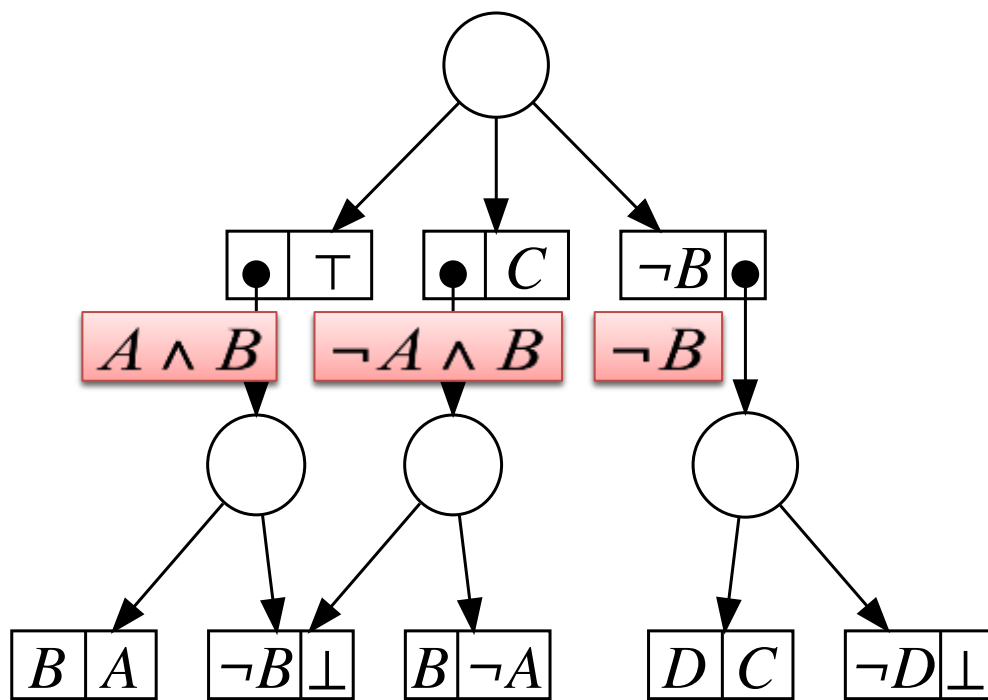
Variable **order** becomes variable **tree** (vtree)

$$f = (A \wedge B) \vee (B \wedge C) \vee (C \wedge D)$$



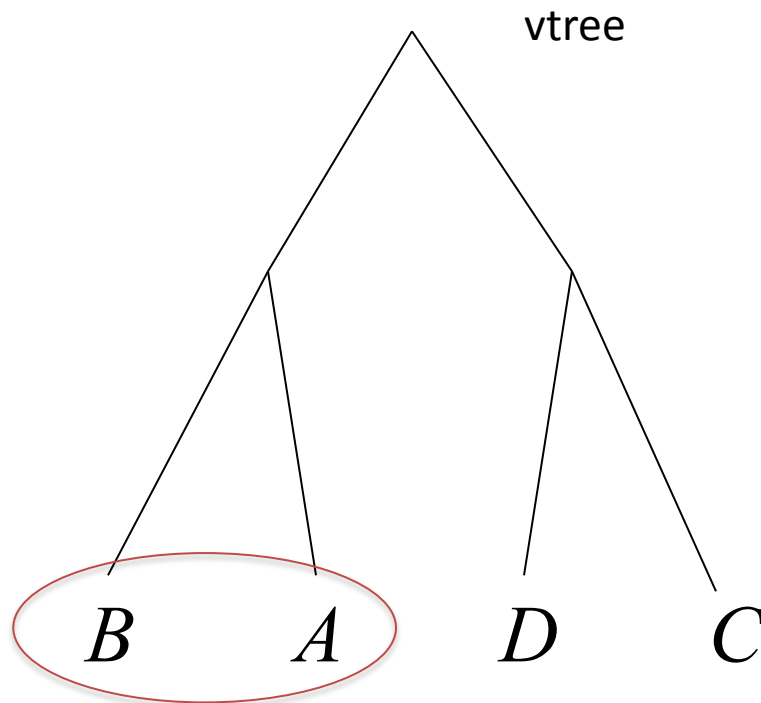
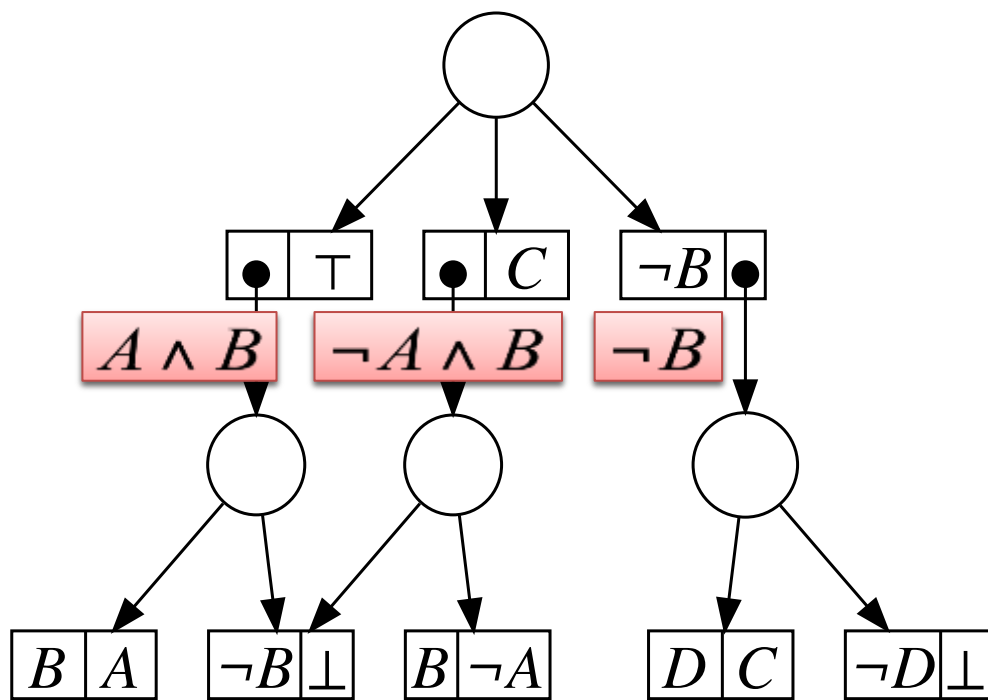
Variable **order** becomes variable **tree** (vtree)

$$f = (A \wedge B) \vee (B \wedge C) \vee (C \wedge D)$$



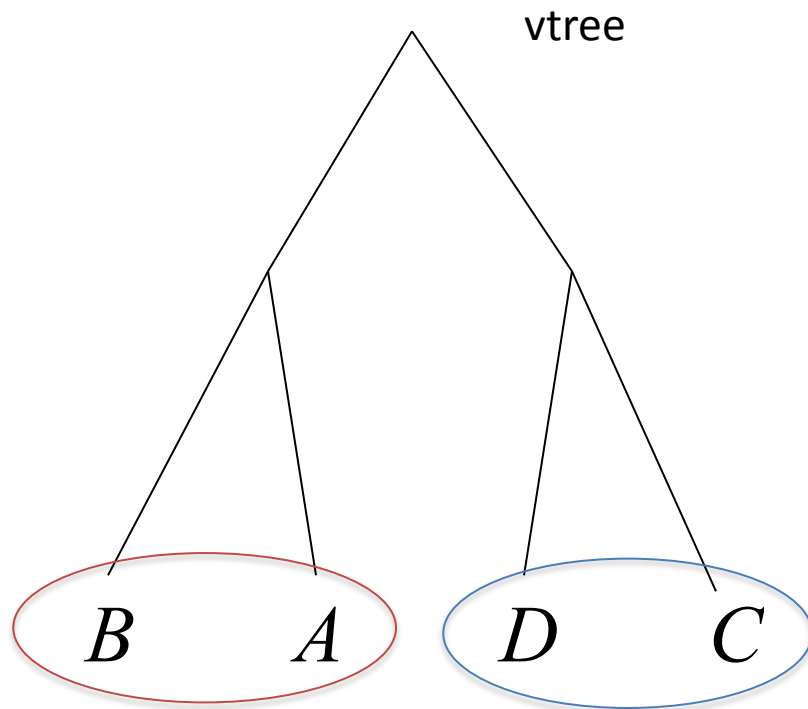
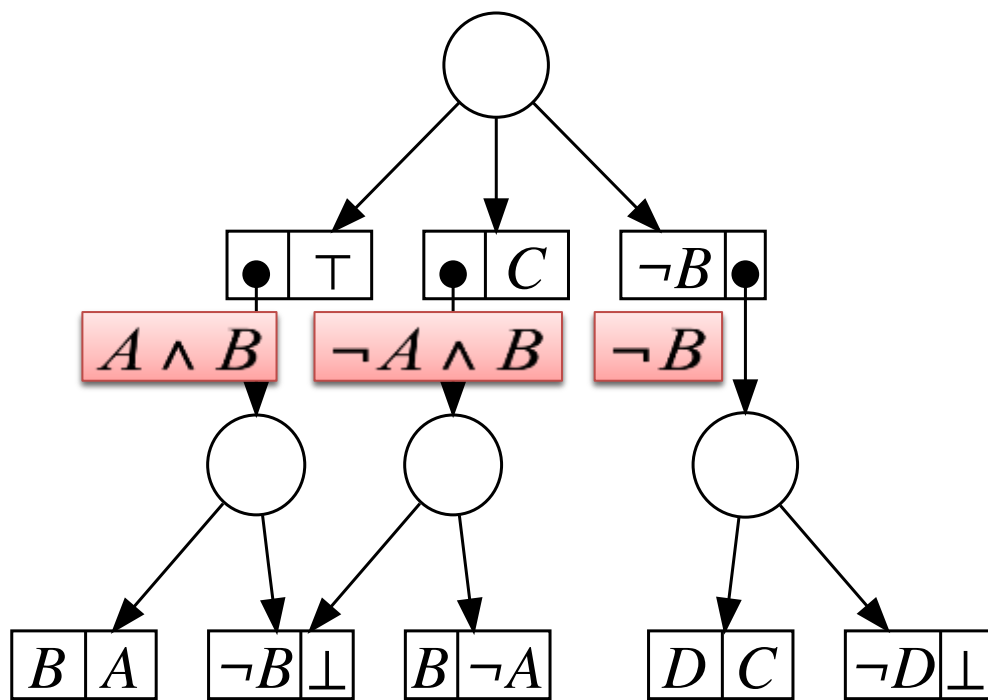
Variable **order** becomes variable **tree** (vtree)

$$f = (A \wedge B) \vee (B \wedge C) \vee (C \wedge D)$$

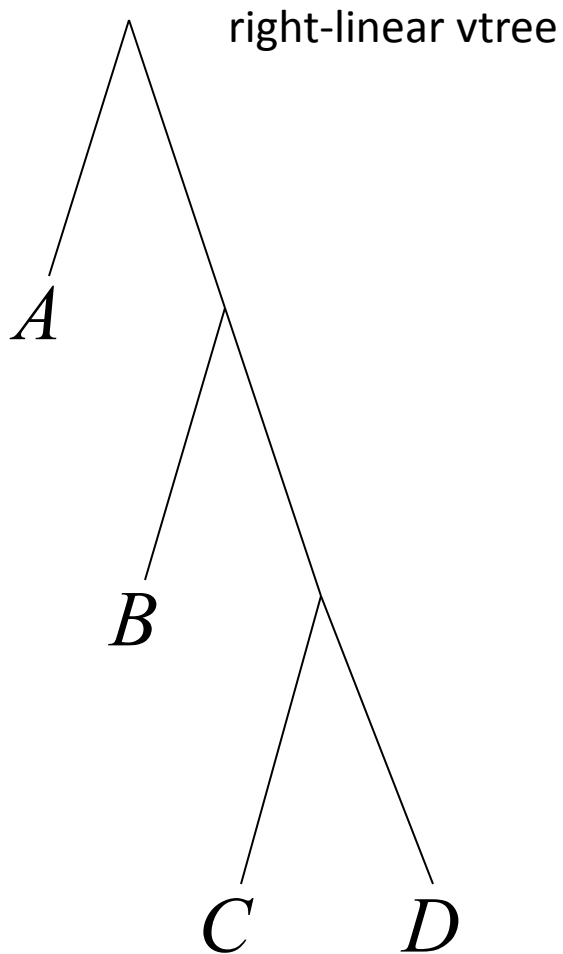


Variable **order** becomes variable **tree** (vtree)

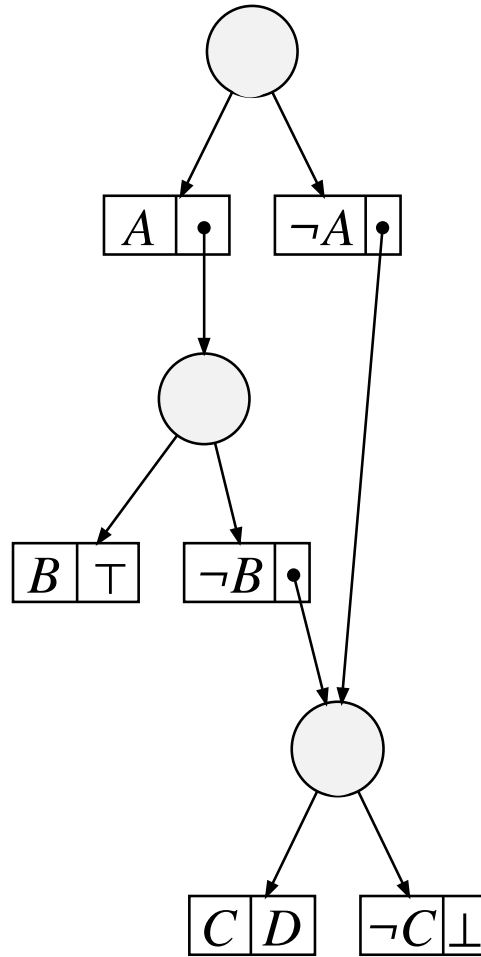
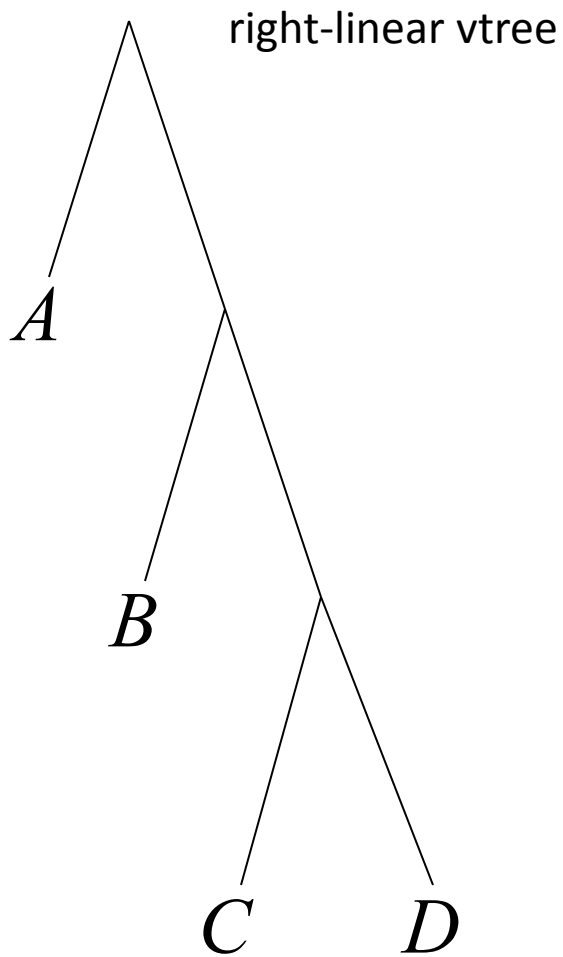
$$f = (A \wedge B) \vee (B \wedge C) \vee (C \wedge D)$$



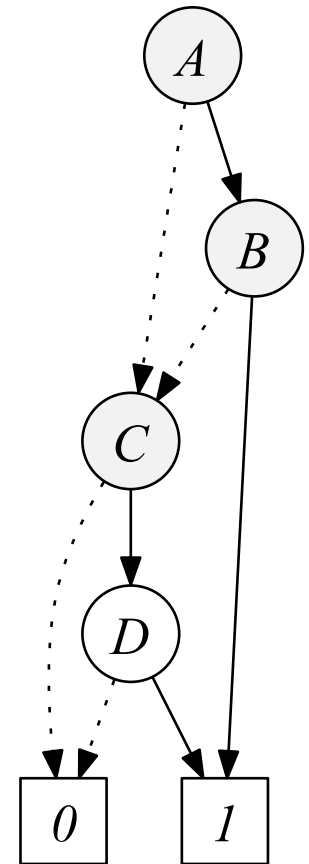
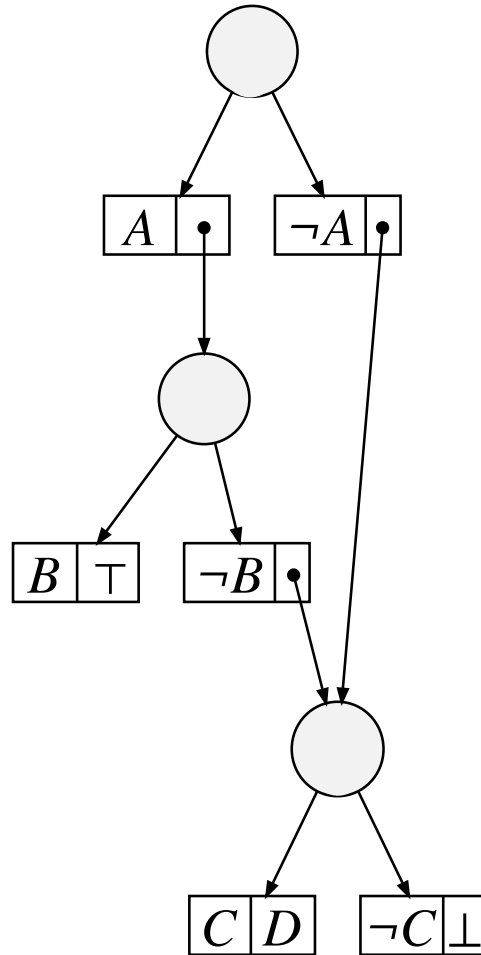
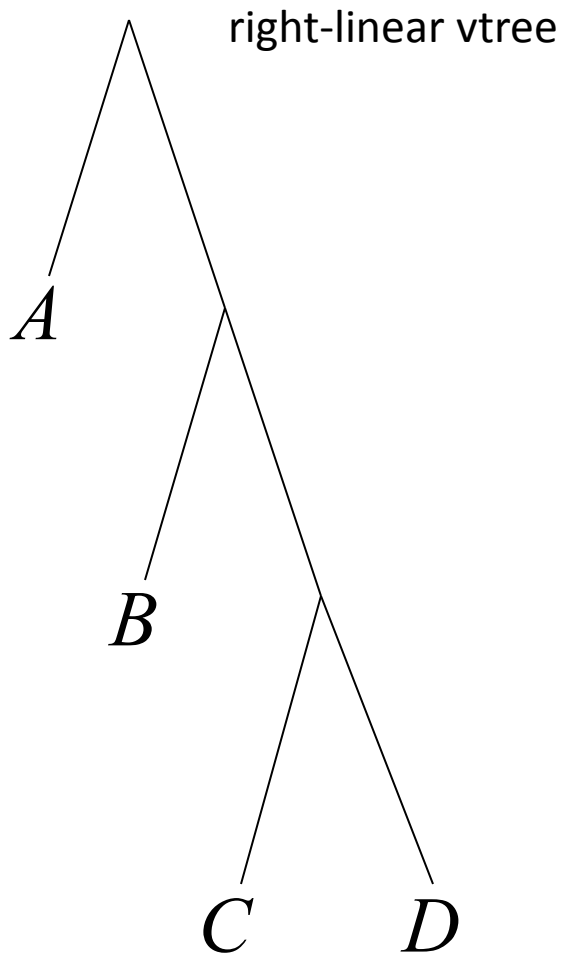
OBDDs are SDDs



OBDDs are SDDs



OBDDs are SDDs



Ingredients for *Delicious Decision Diagrams*



Minimization



Apply Function



Succinctness



Queries



Ingredients for *Delicious Decision Diagrams*



Minimization



Apply Function



Succinctness



Queries





Compression

- An (\mathbf{X}, \mathbf{Y}) -partition: $f(\mathbf{X}, \mathbf{Y}) = p_1(\mathbf{X})s_1(\mathbf{Y}) \vee \dots \vee p_n(\mathbf{X})s_n(\mathbf{Y})$
is **compressed** when subs are distinct: $s_i(\mathbf{Y}) \neq s_j(\mathbf{Y})$ if $i \neq j$
- $f(\mathbf{X}, \mathbf{Y})$ has a **unique** compressed (\mathbf{X}, \mathbf{Y}) -partition

Compression

- An (\mathbf{X}, \mathbf{Y}) -partition: $f(\mathbf{X}, \mathbf{Y}) = p_1(\mathbf{X})s_1(\mathbf{Y}) \vee \dots \vee p_n(\mathbf{X})s_n(\mathbf{Y})$
is **compressed** when subs are distinct: $s_i(\mathbf{Y}) \neq s_j(\mathbf{Y})$ if $i \neq j$
- $f(\mathbf{X}, \mathbf{Y})$ has a **unique** compressed (\mathbf{X}, \mathbf{Y}) -partition

prime	sub
$A \wedge B$	true
$A \wedge \overline{B}$	$C \wedge D$
$\overline{A} \wedge B$	C
$\overline{A} \wedge \overline{B}$	$C \wedge D$

Compression

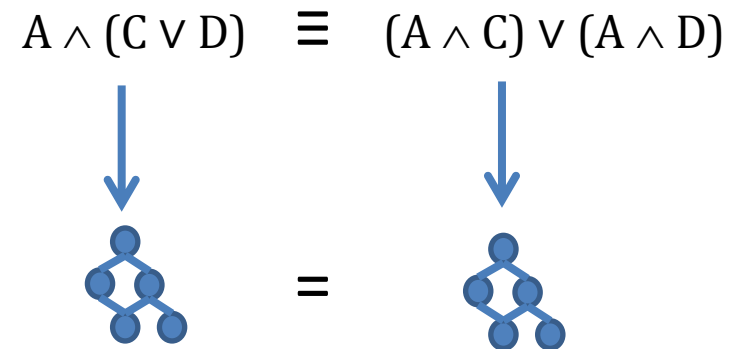
- An (\mathbf{X}, \mathbf{Y}) -partition: $f(\mathbf{X}, \mathbf{Y}) = p_1(\mathbf{X})s_1(\mathbf{Y}) \vee \dots \vee p_n(\mathbf{X})s_n(\mathbf{Y})$ is **compressed** when subs are distinct: $s_i(\mathbf{Y}) \neq s_j(\mathbf{Y})$ if $i \neq j$
- $f(\mathbf{X}, \mathbf{Y})$ has a **unique** compressed (\mathbf{X}, \mathbf{Y}) -partition

prime	sub		prime	sub
$A \wedge B$	true		$A \wedge B$	true
$A \wedge \overline{B}$	$C \wedge D$		$\overline{A} \wedge B$	C
$\overline{A} \wedge B$	C	\vee	\overline{B}	$C \wedge D$
$\overline{A} \wedge \overline{B}$	$C \wedge D$			

SDDs are Canonical

For a fixed vtree (fixing **X, Y** throughout the SDD), compressed SDDs are **canonical!**

Equivalent sentences
have identical circuits.



OBDD Minimization

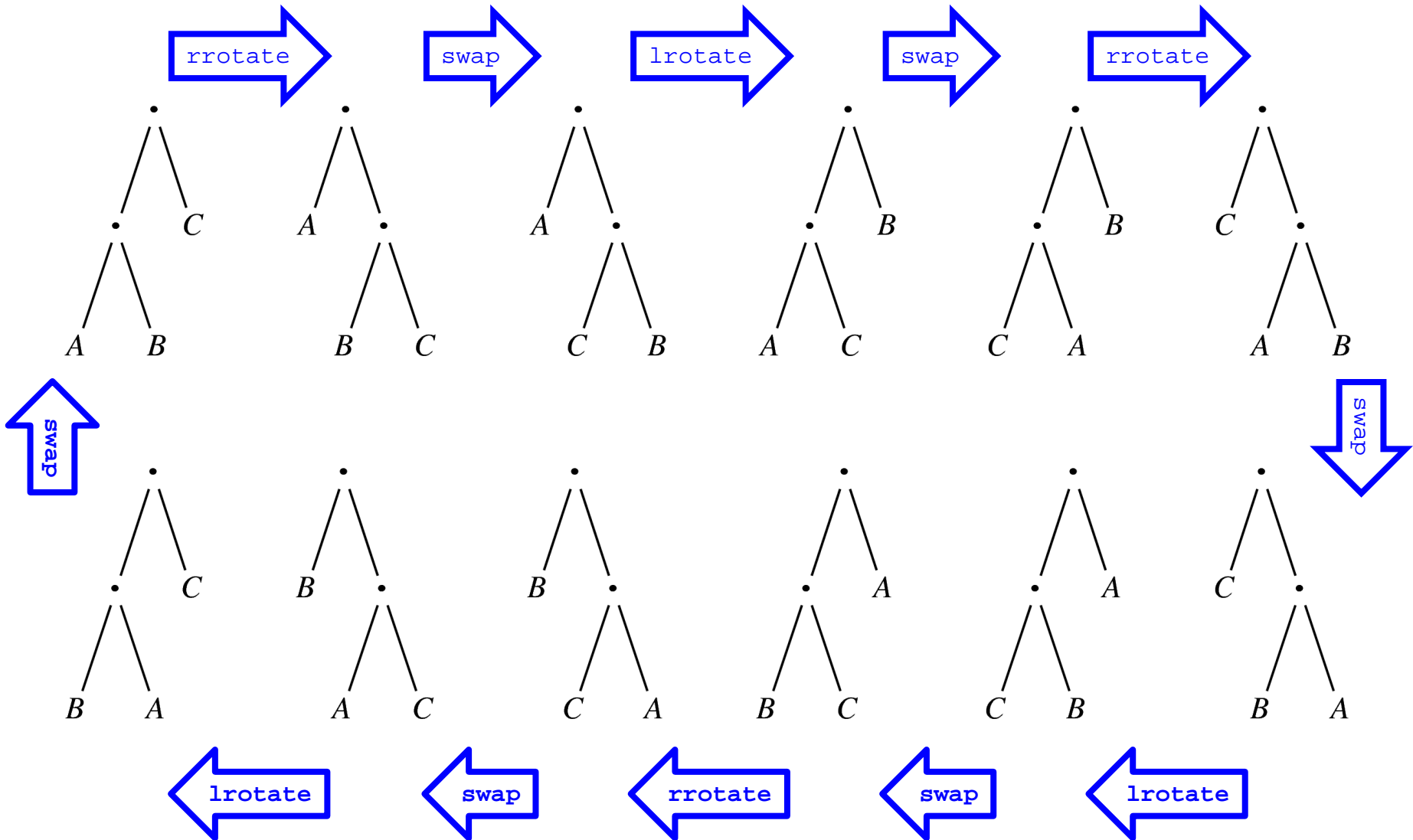
- **24 ordering** of 4 variables

$ABCD \Rightarrow ABDC \Rightarrow ADBC \Rightarrow DABC \Rightarrow DACB \Rightarrow ADCB \Rightarrow$
 $ACDB \Rightarrow ACBD \Rightarrow CABD \Rightarrow CADB \Rightarrow CDAB \Rightarrow DCAB \Rightarrow$
 $DCBA \Rightarrow CDBA \Rightarrow CBDA \Rightarrow CBAD \Rightarrow BCAD \Rightarrow BCDA \Rightarrow$
 $BDCA \Rightarrow DBCA \Rightarrow DBAC \Rightarrow BDAC \Rightarrow BADC \Rightarrow BACD$

- 24 OBDDs for every function over 4 variables
- Searching for an optimal OBDD is searching for an **optimal variable order**



SDD Minimization



Ingredients for *Delicious Decision Diagrams*



Minimization



Apply Function



Succinctness

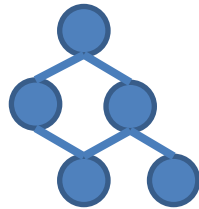
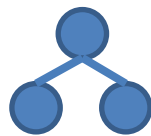
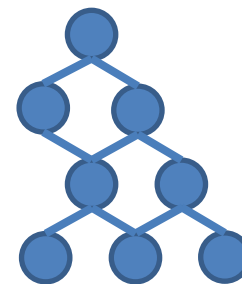


Queries



Efficient Apply Function

- Build Boolean **combinations** of existing circuits
- Compile **arbitrary** sentence incrementally

 $(A \oplus (B \wedge D))$
 $(C \vee D)$
 $(A \oplus (B \wedge D)) \wedge (C \vee D)$

 \wedge

 $=$


- **Polytime** Apply: one Apply cannot blow up size

$$\left| \left(\begin{array}{c} \text{Tree 1} \\ \wedge \\ \text{Tree 2} \end{array} \right) \right| = O\left(\left| \begin{array}{c} \text{Tree 1} \\ \text{Tree 2} \end{array} \right| \times \left| \begin{array}{c} \text{Tree 2} \end{array} \right| \right)$$

Is Apply for SDDs Polytime?

Algorithm 1 $\text{Apply}(\alpha, \beta, \circ)$

```

1: if  $\alpha$  and  $\beta$  are constants or literals then
2:   return  $\alpha \circ \beta$            // result is a constant or literal
3: else if  $\text{Cache}(\alpha, \beta, \circ) \neq \text{nil}$  then
4:   return  $\text{Cache}(\alpha, \beta, \circ)$  // has been computed before
5: else
6:    $\gamma \leftarrow \{\}$ 
7:   for all elements  $(p_i, s_i)$  in  $\alpha$  do
8:     for all elements  $(q_j, r_j)$  in  $\beta$  do
9:        $p \leftarrow \text{Apply}(p_i, q_j, \wedge)$ 
10:      if  $p$  is consistent then
11:         $s \leftarrow \text{Apply}(s_i, r_j, \circ)$ 
12:        add element  $(p, s)$  to  $\gamma$ 

14: return  $\text{Cache}(\alpha, \beta, \circ) \leftarrow \text{UniqueD}(\gamma)$ 

```

- $|\alpha| \times |\beta|$
recursive calls
- **Polytime!**

Ingredients for *Delicious Decision Diagrams*



Minimization



Apply Function



Succinctness



Queries





Succinctness

- Theory

- OBDD \subset SDD thus SDD **never larger** than OBDD
- Quasi-polynomial separation with OBDD
*OBDD can be **much larger** than SDD*
- **Treewidth** upper bounds (important in AI!)

- Practice

- SDD Compiler available and effective  
- SDD Package: <http://reasoning.cs.ucla.edu/sdd/>
- Can obtain orders of magnitude improvements

Ingredients for *Delicious Decision Diagrams*



Minimization



Apply Function



Succinctness



Queries




Queries

- OBDDs are Swiss army knife of supported queries
- SDDs are **equally powerful**

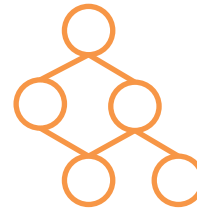
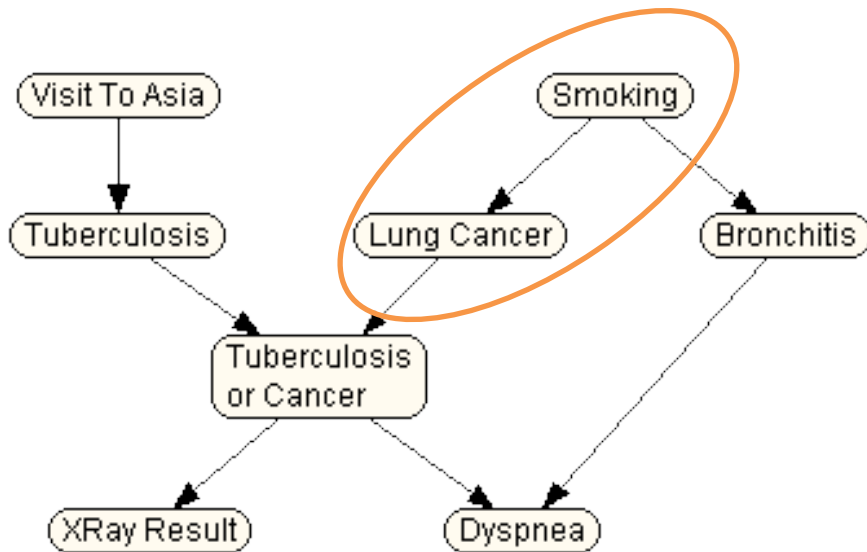
Query	Description	OBDD	SDD
CO	consistency	✓	✓
VA	validity	✓	✓
CE	clausal entailment	✓	✓
IM	implicant check	✓	✓
EQ	equivalence check	✓	✓
CT	model counting	✓	✓
SE	sentential entailment	✓	✓
ME	model enumeration	✓	✓



- Some enabled by canonicity + apply 
- E.g., (Weighted) Model Counting for Probabilistic reasoning (E.g., $\Pr(\text{bill passes} \mid \text{Vote1=Yea})$)

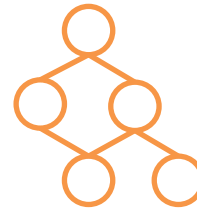
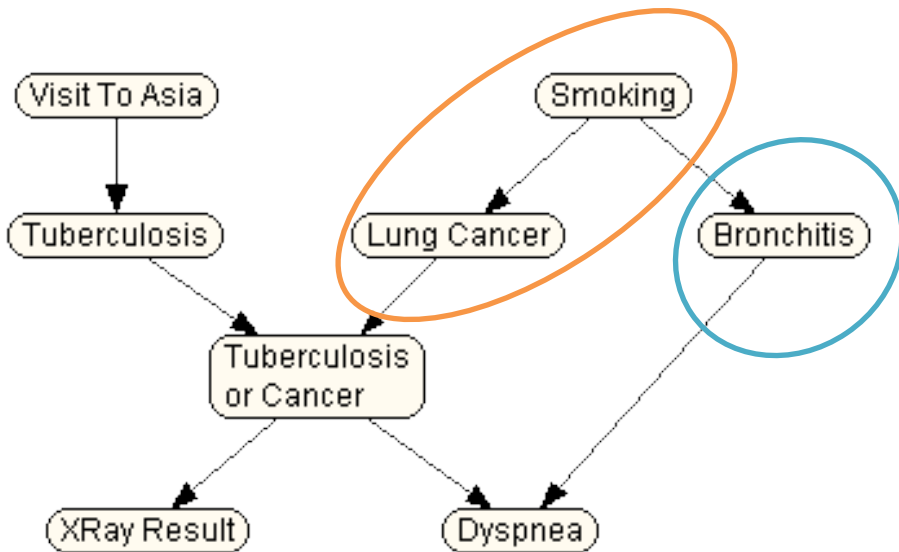
Application: Bayesian Networks

- Incrementally compile network  



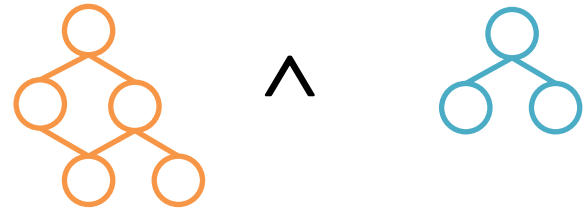
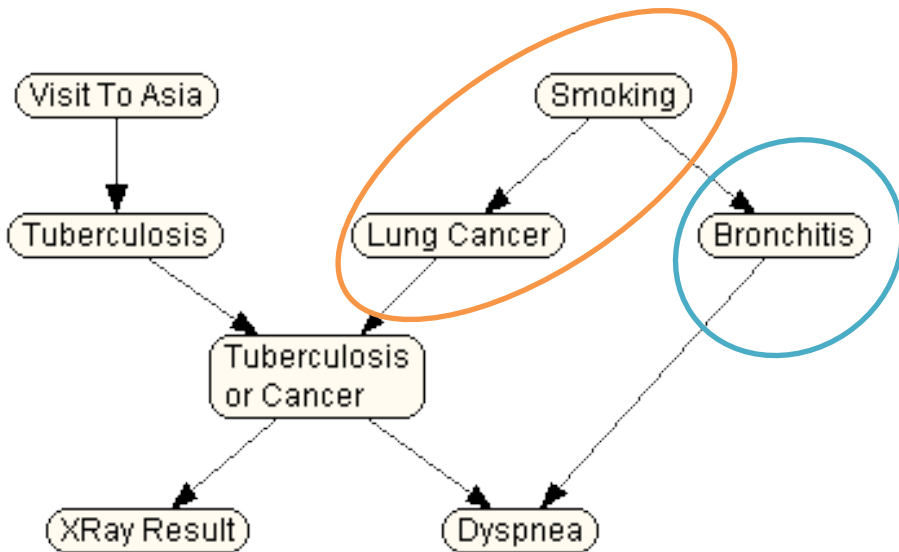
Application: Bayesian Networks

- Incrementally compile network  



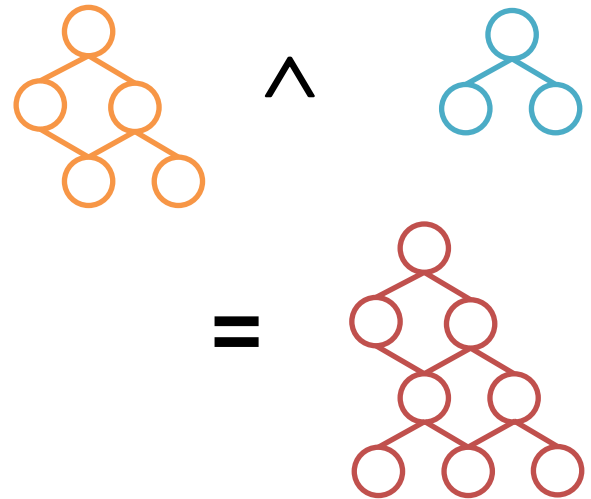
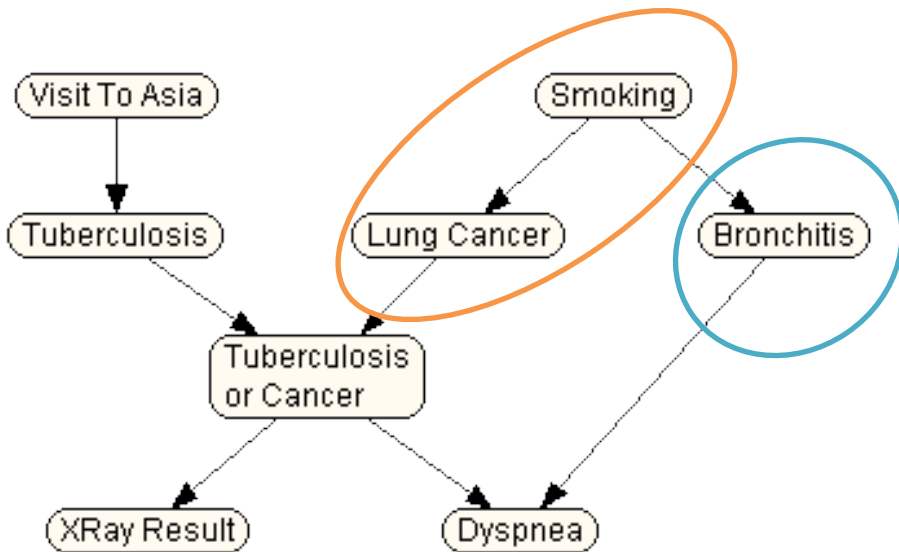
Application: Bayesian Networks

- Incrementally compile network  



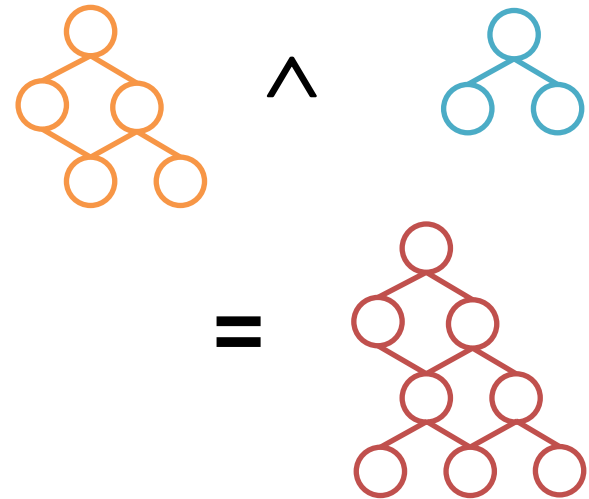
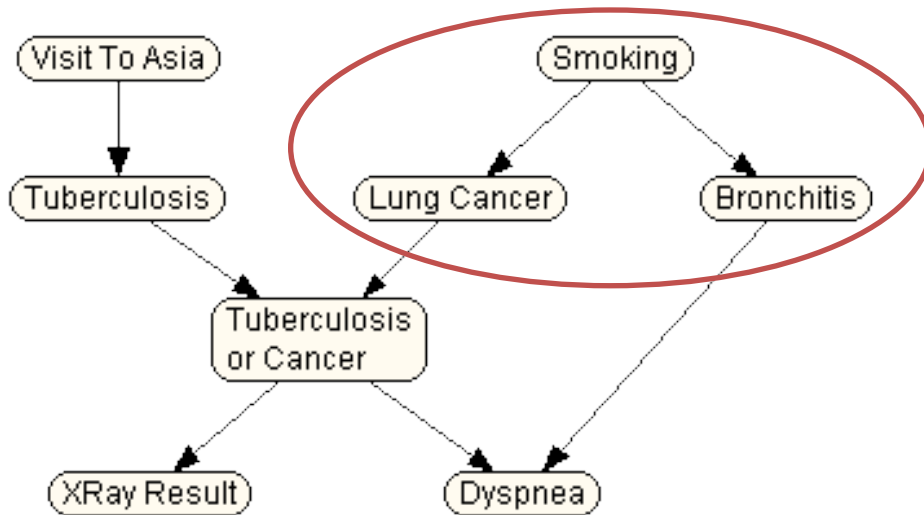
Application: Bayesian Networks

- Incrementally compile network  






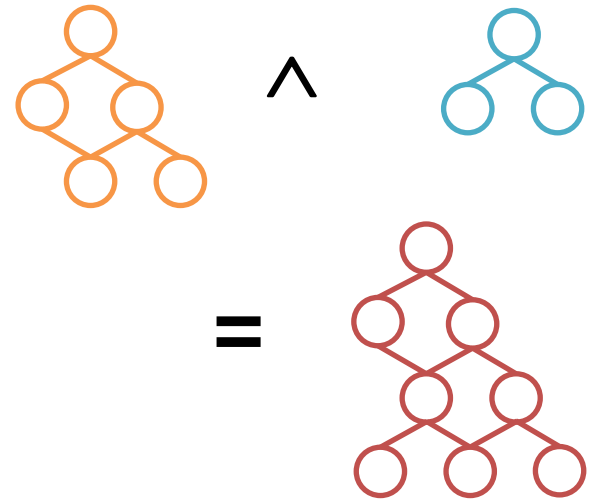
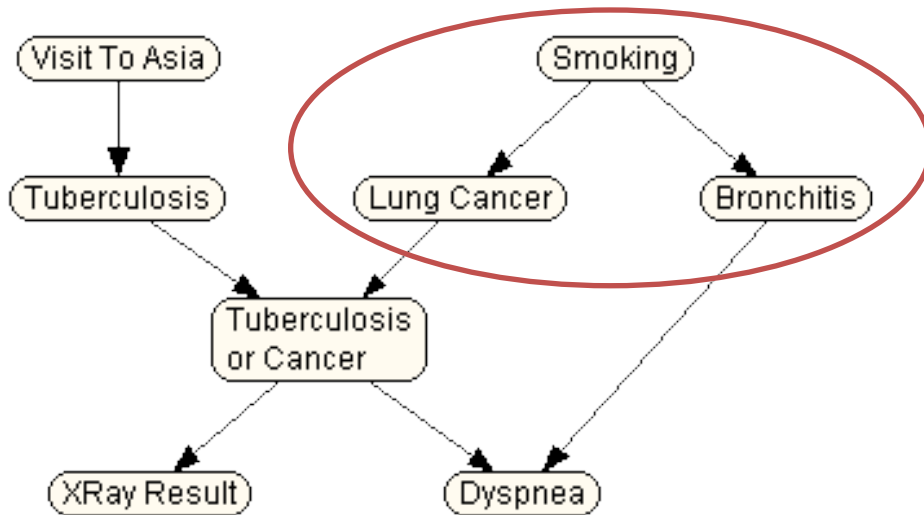
Application: Bayesian Networks

- Incrementally compile network  







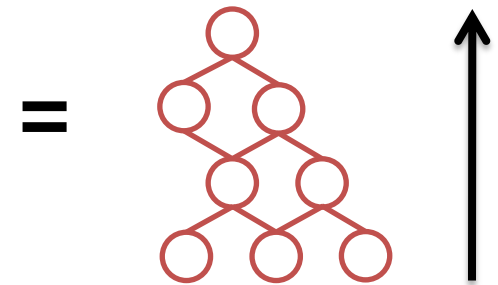
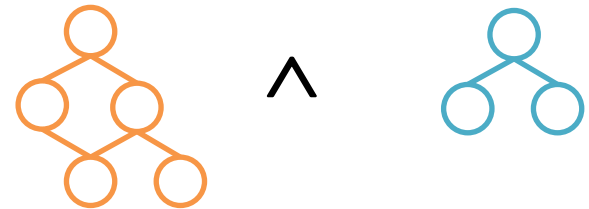
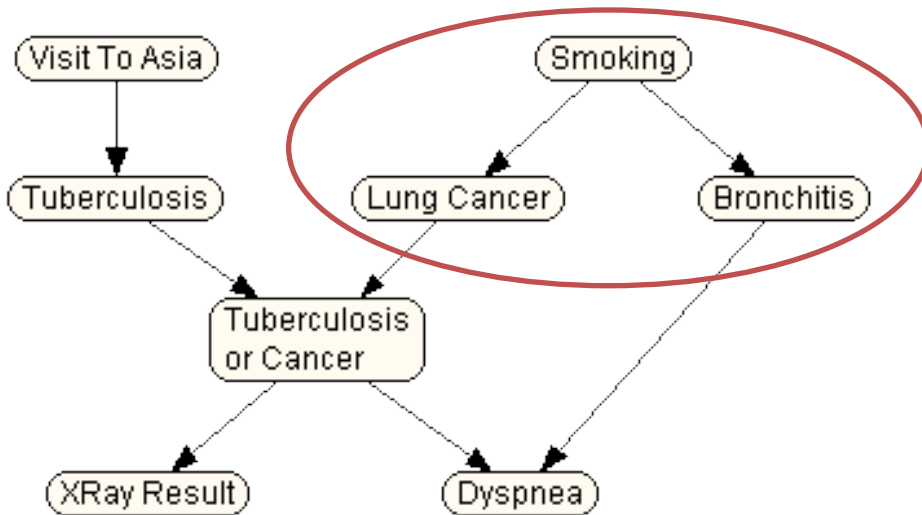
Application: Bayesian Networks

- Incrementally compile network  
- Compute probability of any query 



Application: Bayesian Networks

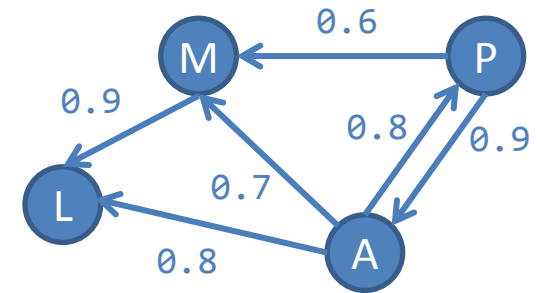
- Incrementally compile network  
- Compute probability of any query 
- Better than state of the art (treewidth) 



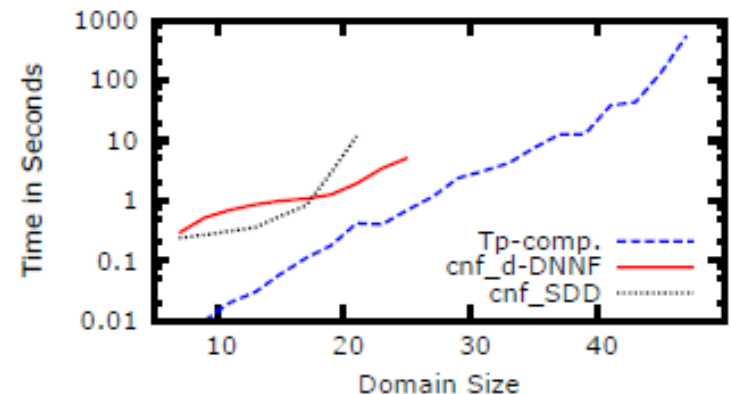
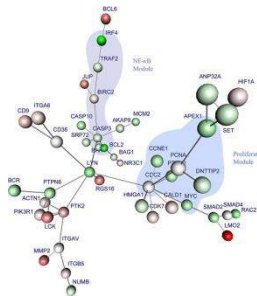
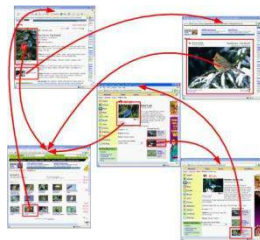
Application: Probabilistic Programming

Model = program with random numbers

```
reach(X,Y) :- flight(X,Y).  
reach(X,Y) :- flight(X,Z), reach(Z,Y).
```






State of the art inference: SDDs

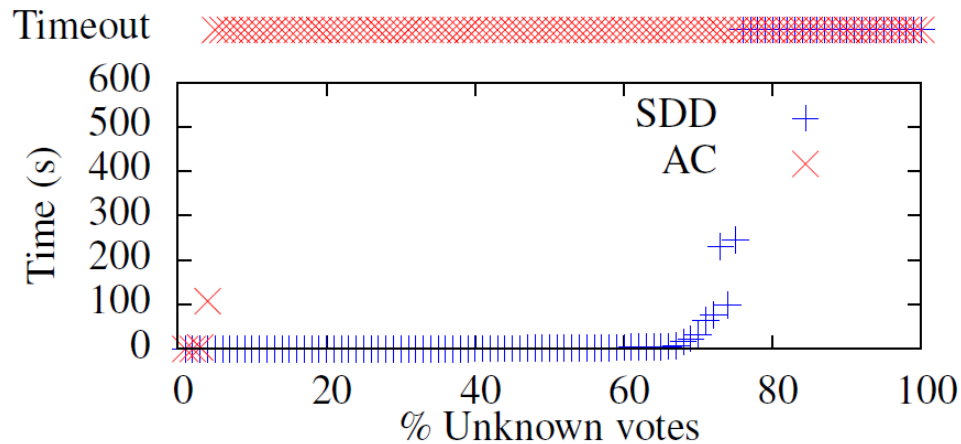


Application: Tractable Learning

- Given: data
- Objective:
 - learn a probability distribution
 - ensure distribution is tractable for querying
- Unstructured space: Voting data
- Structured space: Movie recommendation

Learning in Unstructured Spaces

- Voting data from US House
1764 votes of 453 congressmen
- Learn distribution (Markov network) 
- Represent as SDD to ensure tractability 
- Query efficiency 



Learning in Structured Spaces

Student enrollment constraints:

- Must take at least one of Probability or Logic.

$$P \vee L$$

- Probability is a prerequisite for AI.

$$A \Rightarrow P$$

- The prerequisites for KR is either AI or Logic.

$$K \Rightarrow (P \vee L)$$

$$w = A \wedge K \wedge L \wedge \neg P \text{ *impossible*}$$

L	K	P	A	Students
0	0	1	0	6
0	0	1	1	54
0	1	1	1	10
1	0	0	0	5
1	0	1	0	1
1	0	1	1	0
1	1	0	0	17
1	1	1	0	4
1	1	1	1	3

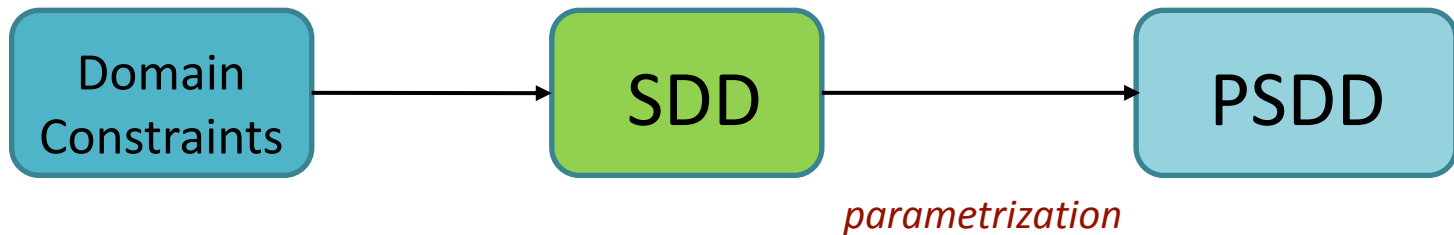
Table 1: Student enrollment data.

Example: Rankings and Permutations

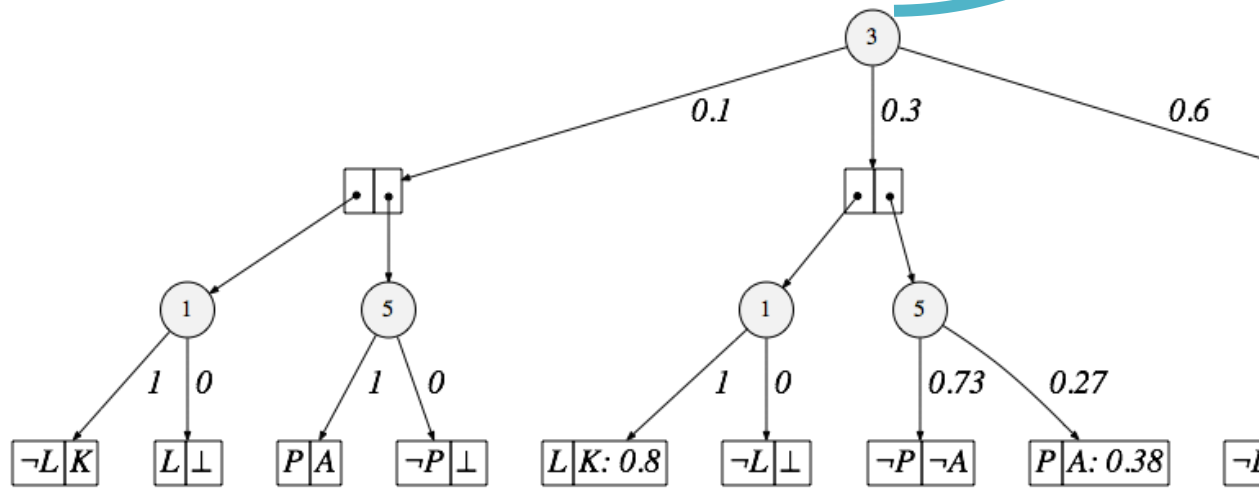
rank	user 1	rank	user 2	rank	user 3
1	The Godfather	1	Star Wars V: The Empire Strikes Back	1	The Usual Suspects
2	Raiders of the Lost Ark	2	Star Wars IV: A New Hope	2	One Flew over the Cuckoo's Nest
3	Casablanca	3	The Godfather	3	The Godfather: Part II
4	The Shawshank Redemption	4	The Shawshank Redemption	4	Monty Python and the Holy Grail
5	Schindler's List	5	The Usual Suspects	5	Star Wars IV: A New Hope
⋮	⋮	⋮	⋮	⋮	⋮

Learn rankings of movies (permutations):
Predict new movies given preferences

Distributions over Structured Spaces: PSDDs

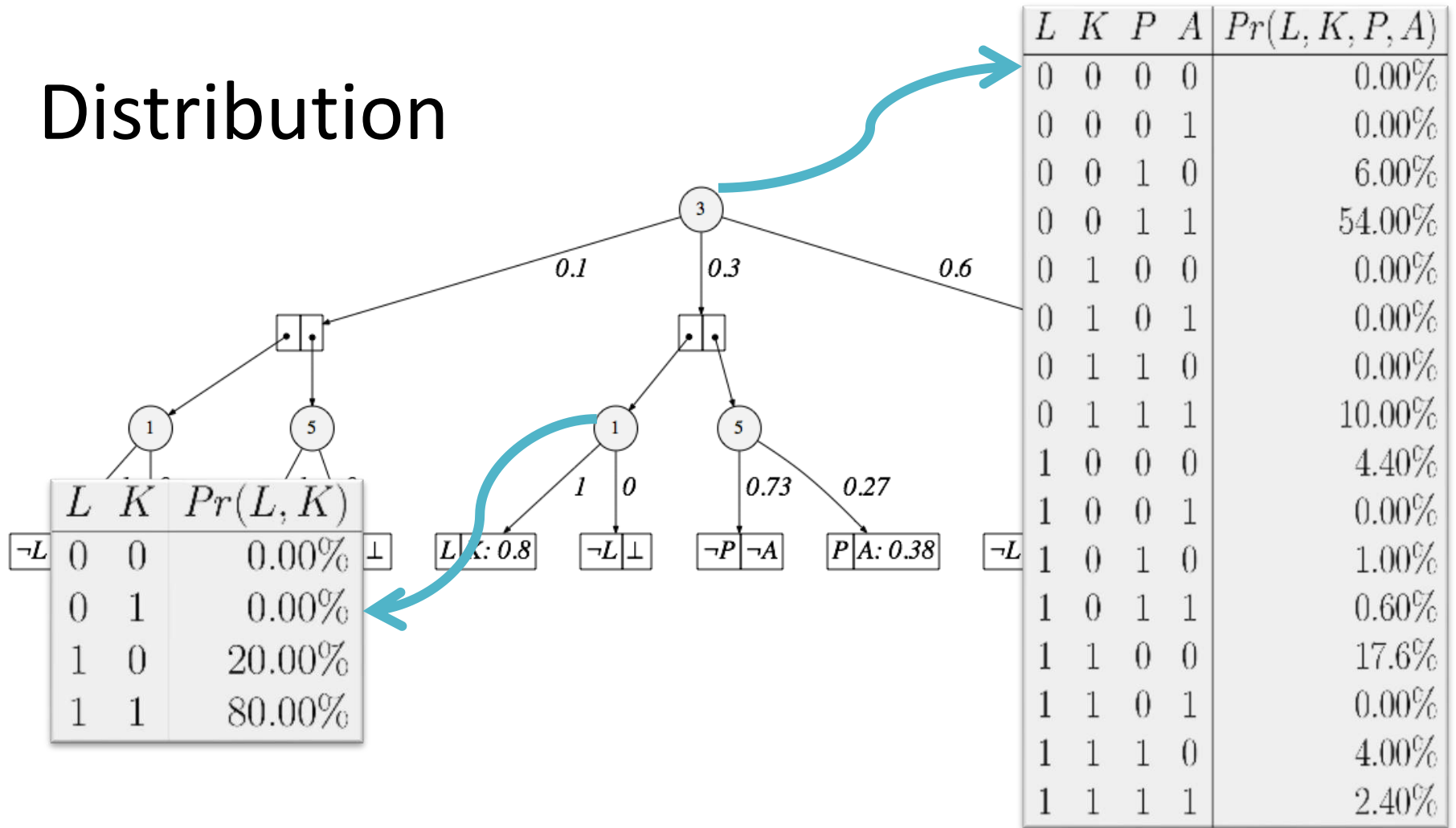


Distribution

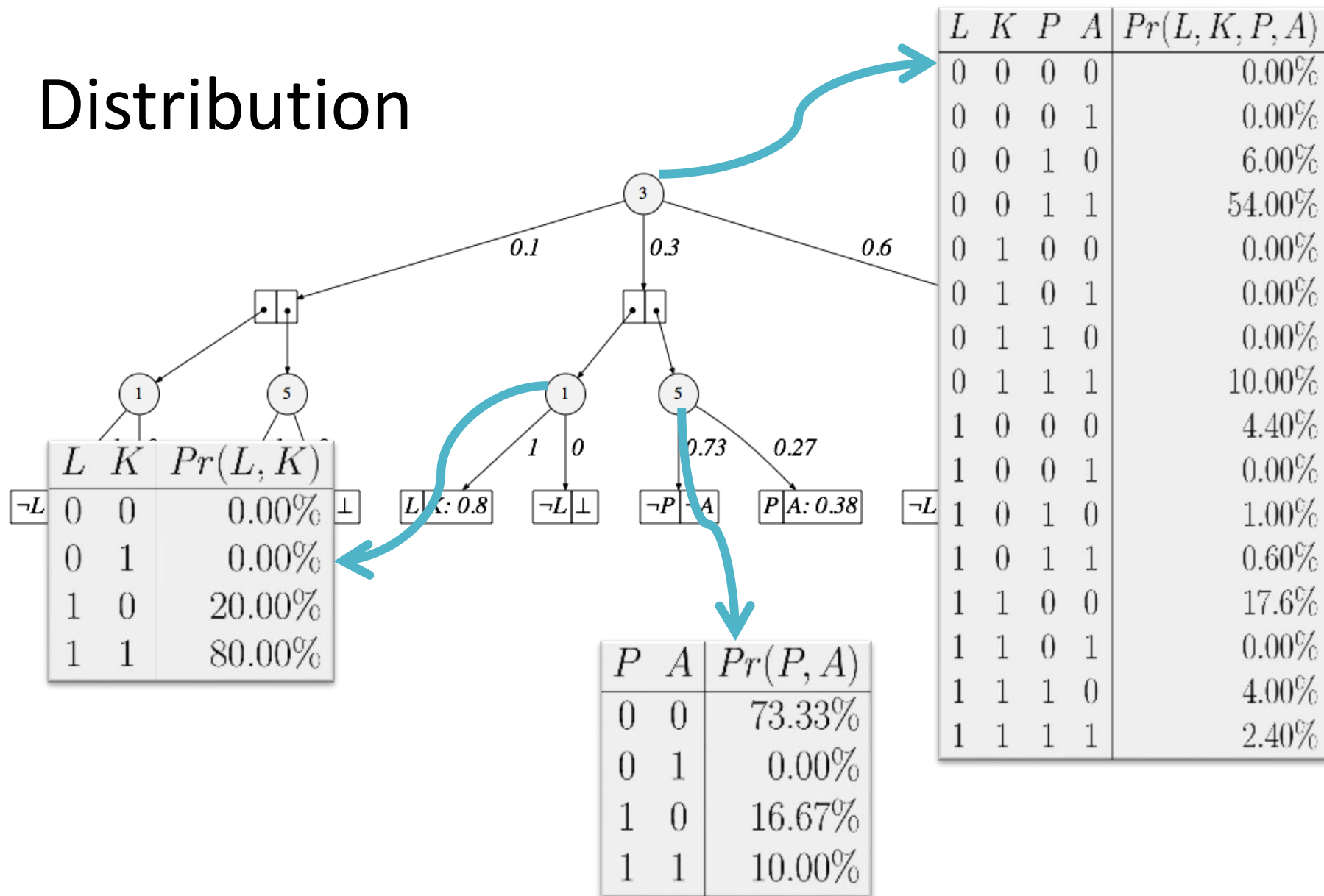


<i>L</i>	<i>K</i>	<i>P</i>	<i>A</i>	$Pr(L, K, P, A)$
0	0	0	0	0.00%
0	0	0	1	0.00%
0	0	1	0	6.00%
0	0	1	1	54.00%
0	1	0	0	0.00%
0	1	0	1	0.00%
0	1	1	0	0.00%
0	1	1	1	10.00%
1	0	0	0	4.40%
1	0	0	1	0.00%
1	0	1	0	1.00%
1	0	1	1	0.60%
1	1	0	0	17.6%
1	1	0	1	0.00%
1	1	1	0	4.00%
1	1	1	1	2.40%

Distribution



Distribution



Reasoning with PSDDs

Example: Preference Distributions

observe:

- favorite movie is Star Wars V

rank	movie
1	Star Wars V: The Empire Strikes Back
2	Star Wars IV: A New Hope
3	The Godfather
4	The Shawshank Redemption
5	The Usual Suspects

observe:

- favorite movie is Star Wars V
- no other Star Wars movie in top-5
- at least one **comedy** in top-5

rank	movie
1	Star Wars V: The Empire Strikes Back
2	American Beauty
3	The Godfather
4	The Usual Suspects
5	The Shawshank Redemption

Conclusions

- SDD a strict **superset** of OBDD:
 - Characterized by trees, which include orders
 - Branch over sentences, which include literals
- SDDs maintain key **properties** of OBDDs:
 - Canonical, Polytime* Apply, Queries, etc.
- SDDs are more **succinct**
 - Treewidth instead of pathwidth
- Lots of **applications** in probabilistic AI and ML



References

- Darwiche, Adnan. "SDD: A new canonical representation of propositional knowledge bases." Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI). Vol. 22. No. 1. 2011.
- Xue, Yexiang, Arthur Choi, and Adnan Darwiche. "Basing decisions on sentences in decision diagrams." Twenty-Sixth AAAI Conference on Artificial Intelligence. 2012.
- Choi, Arthur, and Adnan Darwiche. "Dynamic minimization of sentential decision diagrams." In Twenty-Seventh AAAI Conference on Artificial Intelligence. 2013.
- Razgon, Igor. "On OBDDs for CNFs of bounded treewidth." *arXiv preprint arXiv:1308.3829* (2013).
- Choi, Arthur, Doga Kisa, and Adnan Darwiche. "Compiling probabilistic graphical models using sentential decision diagrams." Symbolic and Quantitative Approaches to Reasoning with Uncertainty. Springer Berlin Heidelberg, 2013. 121-132
- Kisa, Doga, et al. "Probabilistic sentential decision diagrams." Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR). 2014.

References

- Vlasselaer, Jonas, et al. "Compiling probabilistic logic programs into sentential decision diagrams." Workshop on Probabilistic Logic Programming (PLP), Vienna. 2014.
- Kisa, Doga, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche. "Probabilistic sentential decision diagrams: Learning with massive logical constraints.". 2014
- Oztok, Umut, and Adnan Darwiche. "On compiling cnf into decision-dnnf." In Principles and Practice of Constraint Programming, pp. 42-57. Springer International Publishing, 2014.
- Van den Broeck, Guy, and Adnan Darwiche. "On the role of canonicity in knowledge compilation." Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence. 2015.
- Choi, Arthur, Guy Van den Broeck, and Adnan Darwiche. "Tractable learning for structured probability spaces: a case study in learning preference distributions." Proceedings of 24th International Joint Conference on Artificial Intelligence (IJCAI). 2015.

References

- Choi, Arthur, Guy Van den Broeck, and Adnan Darwiche. "Tractable learning for structured probability spaces: a case study in learning preference distributions." Proceedings of 24th International Joint Conference on Artificial Intelligence (IJCAI). 2015.
- Oztok, Umut, and Adnan Darwiche. "A top-down compiler for sentential decision diagrams." Proceedings of the 24th International Conference on Artificial Intelligence. AAAI Press, 2015.
- Vlasselaer , Jonas, Guy Van den Broeck, Angelika Kimmig, Wannes Meert, and Luc De Raedt. Anytime Inference in Probabilistic Logic Programs with Tp-compilation, In Proceedings of 24th International Joint Conference on Artificial Intelligence (IJCAI), 2015.
- Bekker, Jessa, Jesse Davis, Arthur Choi, Adnan Darwiche, and Guy Van den Broeck. Tractable Learning for Complex Probability Queries, In Advances in Neural Information Processing Systems 28 (NIPS), 2015.