CS7480: Topics in Programming Languages: Probabilistic Programming

Lecture 7: Scaling Discrete Inference

Instructor: Steven Holtzen

Place: Northeastern University

Term: Fall 2021

Course webpage:
https://www.khoury.northeastern.edu/home/sholtzen/CS7480Fall21/

Course Updates

- Updated project online (dated Oct. 1 now)

Scaling Discrete Inference

- **Goal**: Scale exact inference to large discrete probabilistic programs

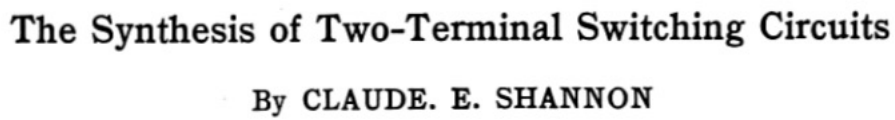
- **Why?**
  1. Discrete inference has lots of applications
  2. Scalability is important: practical systems are large
  3. Current existing systems struggle with discreteness

- **How?**
  - Inference via compilation to circuits

Circuits for Probabilistic Inference

History

- Circuits are a fundamental computing primitive
- Their relationship with complexity was introduced by Shannon

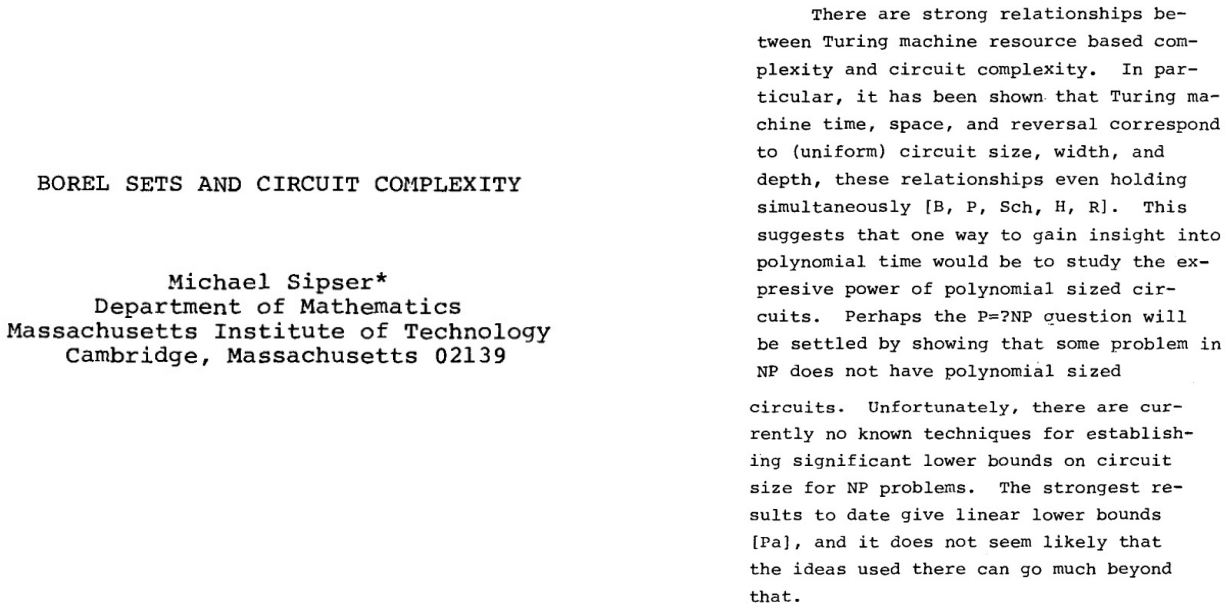

The Synthesis of Two-Terminal Switching Circuits

By CLAUDE. E. SHANNON

- Proved that almost all Boolean functions on n variables require circuits of size $\Theta(2^n/n)$, where n is #variables
  - First *circuit lower bound*

History: P vs NP

- Sipser, Michael. "Borel sets and circuit complexity." *Proceedings of the fifteenth annual ACM symposium on Theory of computing*. 1983.



BOREL SETS AND CIRCUIT COMPLEXITY

Michael Sipser*
Department of Mathematics
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

There are strong relationships between Turing machine resource based complexity and circuit complexity. In particular, it has been shown that Turing machine time, space, and reversal correspond to (uniform) circuit size, width, and depth, these relationships even holding simultaneously [B, P, Sch, H, R]. This suggests that one way to gain insight into polynomial time would be to study the expresive power of polynomial sized circuits. Perhaps the P=?NP question will be settled by showing that some problem in NP does not have polynomial sized circuits. Unfortunately, there are currently no known techniques for establishing significant lower bounds on circuit size for NP problems. The strongest results to date give linear lower bounds [Pa], and it does not seem likely that the ideas used there can go much beyond that.

Connection between conciseness and P/NP

Idea: Study Restricted Classes of Circuits

Докл. Акад. Наук СССР
Том 281 (1985), № 4

Soviet Math. Dokl.
Vol. 31 (1985), No. 2

LOWER BOUNDS FOR THE MONOTONE COMPLEXITY OF SOME BOOLEAN FUNCTIONS

UDC 519.95

A. A. RAZBOROV

The *combinatorial complexity* L_f *of a Boolean function* $f(x_1, \ldots, x_n)$ is the least number of logical elements AND, OR and NOT necessary for its realization in the form of a functional scheme. It is well known (see, for example, [1]) that there are Boolean functions whose combinatorial complexity is an exponential function of the number of variables. In a recent article [2], a natural sequence of Boolean functions

$$f_1(x_1, \ldots, x_{n_1}), f_2(x_1, \ldots, x_{n_2}), \ldots, f_m(x_1, \ldots, x_{n_m}), \ldots \tag{1}$$

was constructed, with $L_{f_m} \geq C^{n_m}$, where $C > 1$ is a universal constant.

In this note we will restrict ourselves to the consideration of sequences of the form (1) satisfying the following condition: the language $\{(\varepsilon_1 \cdots \varepsilon_{n_m}) | m \in \mathbf{N}, f_m(\varepsilon_1, \ldots, \varepsilon_{n_m}) = 1\}$ in the alphabet $\{0, 1\}$ can be recognized by a nondeterministic Turing machine in time which is polynomial in the length of the input n_m (i.e. it is an NP-language). Such sequences will be called *constructive*.

It is interesting to obtain lower bounds on the combinatorial complexity of functions from the constructive sequence (1), for example, in connection with the following remark (derivable from the results of [3]): if there is a constructive sequence of the form (1) such that

$$\overline{\lim_{m \to \infty}} \frac{\log L_{f_m}}{\log n_m} = \infty,$$

then $P \neq NP$. Apparently the strongest result obtained in this direction is found in [4], where an example of a constructive sequence (1) is constructed with $L_{f_m} \geq 2.5 n_m$.

Complexity: Modern Times

- Circuit complexity still hasn't resolved P vs NP ☹
  - Proving lower bounds is extremely difficult
  - Restricted circuit models easier to prove lower bounds in, but become *too* restricted
  - Much success in *other* domains (derandomization, parallel algorithms, etc.)

Circuits in AI

- Circuit lower bounds proved unfruitful in complexity, but the *idea* of circuits as a computation model remained influential

- The AI community transitioned to studying these objects in the context of *knowledge representation* in the early 90s

Tractability vs. Expressiveness

Expressiveness and tractability in knowledge representation and reasoning[1]

HECTOR J. LEVESQUE[2]
Department of Computer Science, University of Toronto, Toronto, Ont., Canada M5S 1A4

AND

RONALD J. BRACHMAN
AT&T Bell Laboratories, 600 Mountain Avenue, 3C-439, Murray Hill, NJ 07974, U.S.A.

Received November 3, 1986
Revision accepted April 8, 1987

A fundamental computational limit on automated reasoning and its effect on knowledge representation is examined. Basically, the problem is that it can be more difficult to reason correctly with one representational language than with another and, moreover, that this difficulty increases dramatically as the expressive power of the language increases. This leads to a tradeoff between the expressiveness of a representational language and its computational tractability. Here we show that this tradeoff can be seen to underlie the differences among a number of existing representational formalisms, in addition to motivating many of the current research issues in knowledge representation.

Key words: knowledge representation, description subsumption, complexity of reasoning, first-order logic, frames, semantic networks, databases.

Levesque, Hector J., and Ronald J. Brachman. "Expressiveness and tractability in knowledge representation and reasoning 1." Computational intelligence 3.1 (1987): 78-93.

Exploring the Landscape

Knowledge Compilation and Theory Approximation

BART SELMAN AND HENRY KAUTZ

AT&T Bell Laboratories, Murray Hill, New Jersey

Abstract. Computational efficiency is a central concern in the design of knowledge representation systems. In order to obtain efficient systems, it has been suggested that one should limit the form of the statements in the knowledge base or use an incomplete inference mechanism. The former approach is often too restrictive for practical applications, whereas the latter leads to uncertainty about exactly what can and cannot be inferred from the knowledge base. We present a third alternative, in which knowledge given in a general representation language is translated (compiled) into a tractable form—allowing for efficient subsequent query answering.

We show how propositional logical theories can be compiled into Horn theories that approximate the original information. The approximations bound the original theory from below and above in terms of logical strength. The procedures are extended to other tractable languages (for example, binary clauses) and to the first-order case. Finally, we demonstrate the generality of our approach by compiling concept descriptions in a general frame-based language into a tractable form.

Selman, Bart, and Henry Kautz. "Knowledge compilation and theory approximation." *Journal of the ACM (JACM)* 43.2 (1996): 193-224.

Mapping the Landscape

Journal of Artificial Intelligence Research 17 (2002) 229-264 Submitted 12/01; published 9/02

A Knowledge Compilation Map

Adnan Darwiche DARWICHE@CS.UCLA.EDU
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095, USA

Pierre Marquis MARQUIS@CRIL.UNIV-ARTOIS.FR
Université d'Artois
F-62307, Lens Cedex, France

- Introduced a formal notion of conciseness (aka succinctness)

Definition 3.1 (Succinctness) *Let* $\mathbf{L}_1$ *and* $\mathbf{L}_2$ *be two subsets of* NNF. $\mathbf{L}_1$ *is at least as succinct as* $\mathbf{L}_2$, *denoted* $\mathbf{L}_1 \leq \mathbf{L}_2$, *iff there exists a polynomial* p *such that for every sentence* $\alpha \in \mathbf{L}_2$, *there exists an equivalent sentence* $\beta \in \mathbf{L}_1$ *where* $|\beta| \leq p(|\alpha|)$. *Here,* $|\alpha|$ *and* $|\beta|$ *are the sizes of* α *and* β, *respectively.*

From KR to Probabilistic Inference

A Logical Approach to Factoring Belief Networks

Adnan Darwiche
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095
darwiche@cs.ucla.edu

Darwiche, Adnan. "A logical approach to factoring belief networks." *KR* 2 (2002): 409-420.

From search to circuits

Weighted Model Counting

- Recall the definition of a weighted Boolean formula, a pair (φ, w):
  - φ is a Boolean sentence
  - w: $L \rightarrow R$ a weight function that maps literals (variable assignments) to real values
  - Valuation : $$[\![(\varphi, w)]\!]v = \begin{cases} \prod_{l \in v} w(l) & \text{if } v \models \varphi \\ 0 & \text{otherwise} \end{cases}$$
  - The *weighted model count* of a weighted Boolean formula is:

$$WMC(\varphi, w) = \sum_{v} [\![(\varphi, w)]\!]v$$

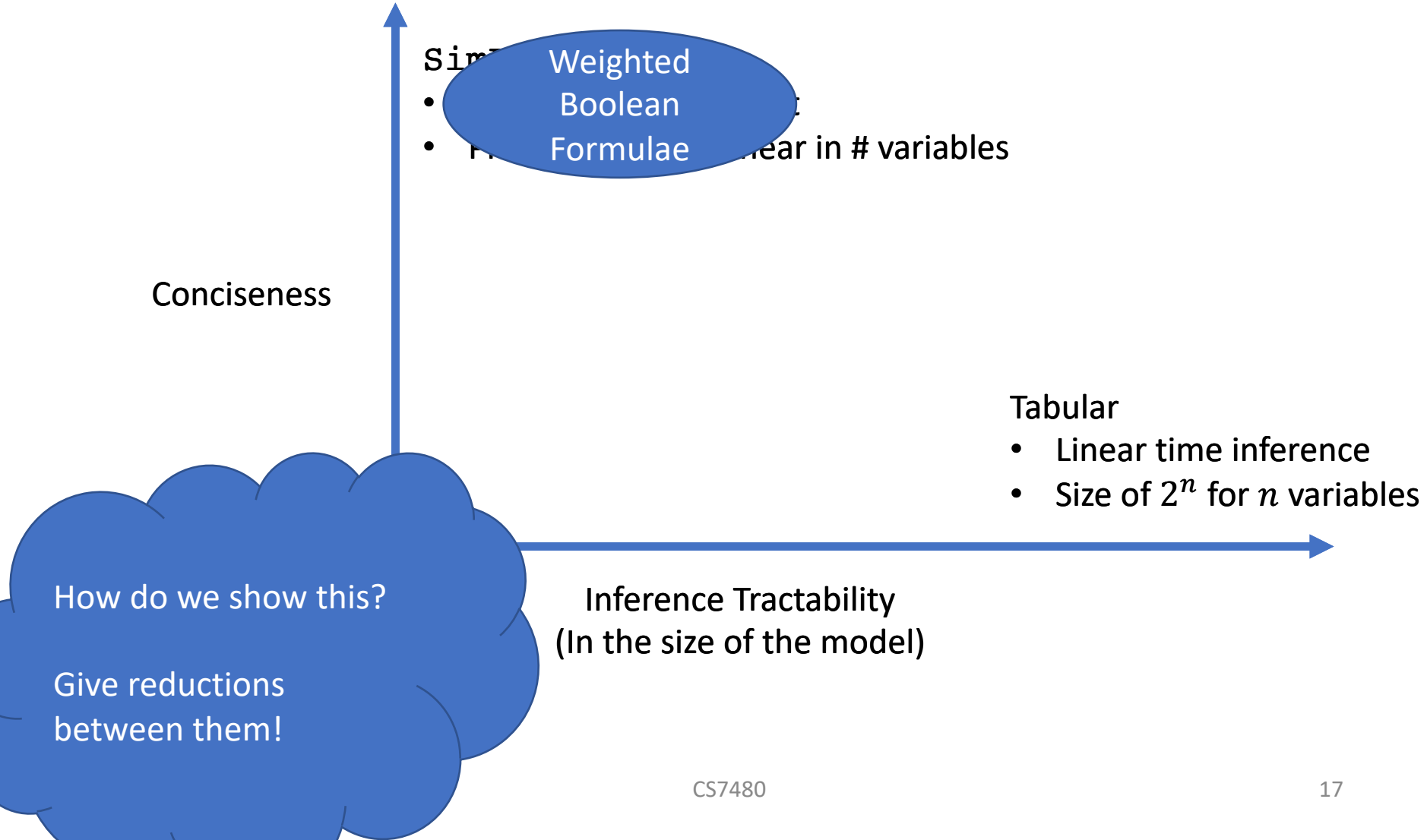
Solving WMC: Tables

$$w = \{f_1 \mapsto 0.1, \bar{f}_1 \mapsto 0.9, f_2 \mapsto 0.3, \bar{f}_3 \mapsto 0.7\}$$
$$\varphi = f_1 \vee f_2$$

f_1	f_2	$[\![(\varphi, w)]\!]v$
T	T	$0.1 * 0.3$
T	F	$0.1 * 0.7$
F	T	$0.9 * 0.3$
F	F	0

- Then, WMC is ***sum of the rows***

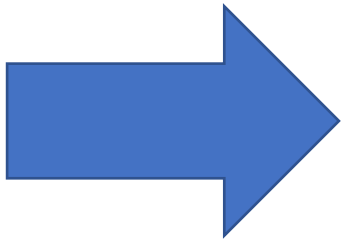
Weighted Boolean Formulae as a PPL



Reducing WMC to SimPPL Inference

Assume $w(A) = 1 - w(\bar{A})$

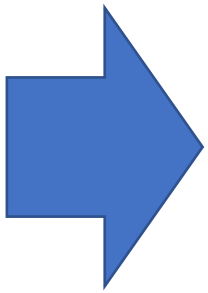
$WMC(f_1 \wedge f_2, w)$



```
f1 ~ flip w(f1);
f2 ~ flip w(f2);
return (&& f1 f2)
```

Reducing SimPPL Inference to WMC

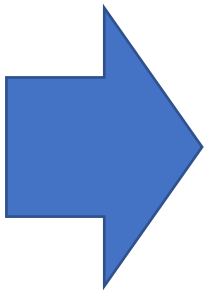
```
f1 ~ flip 0.1;
f2 ~ flip 0.3;
return (|| f1 f2)
```



$$WMC\left(f_1 \vee f_2, \left\{f_1 \mapsto 0.1, \bar{f_1} \mapsto 0.9, f_2 \mapsto 0.3, \bar{f_3} \mapsto 0.7\right\}\right)$$

Handling Observations

```
f1 ~ flip 0.1;
f2 ~ flip 0.3;
observe (|| f1 f2)
return f1
```

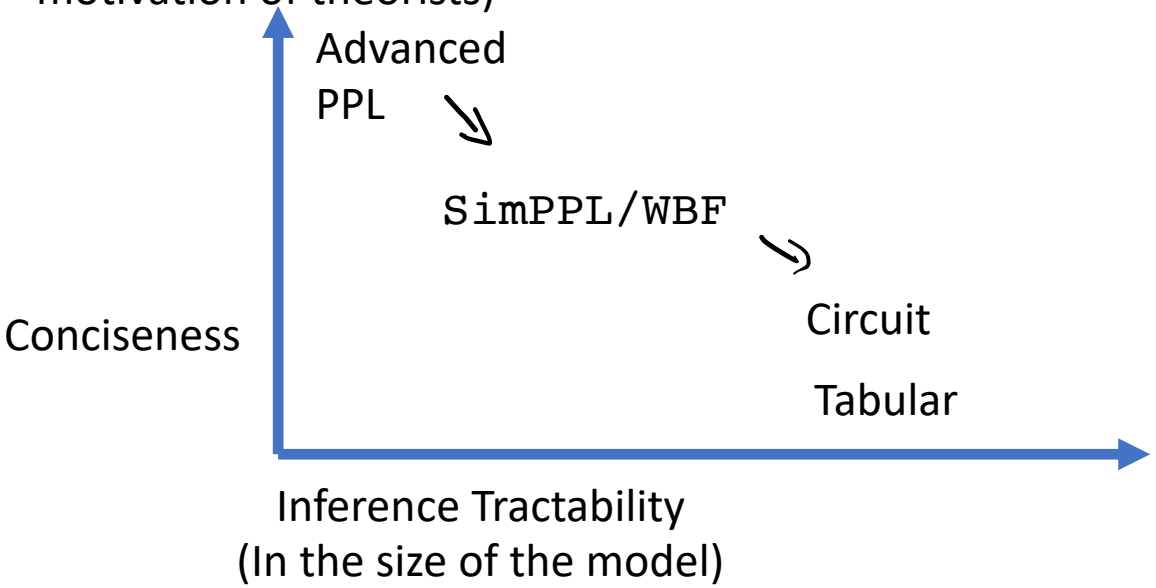


$$\frac{WMC(f_1, w)}{WMC(f_1 \vee f_2, w)}, w = \{f_1 \mapsto 0.1, \bar{f_1} \mapsto 0.9, f_2 \mapsto 0.3, \bar{f_3} \mapsto 0.7\}$$

- What did this buy us so far ? *Nothing!*
  - Inference for arbitrary WBF is *still hard*

Popping Up

- So, WMC and SimPPL are basically the same probabilistic programming language
- Why do this translation?
  - `SimPPL` is *very simple*; this reduction may not hold for programming languages with more features
  - We can design inference algorithms on this "assembly language" (similar to motivation of theorists)



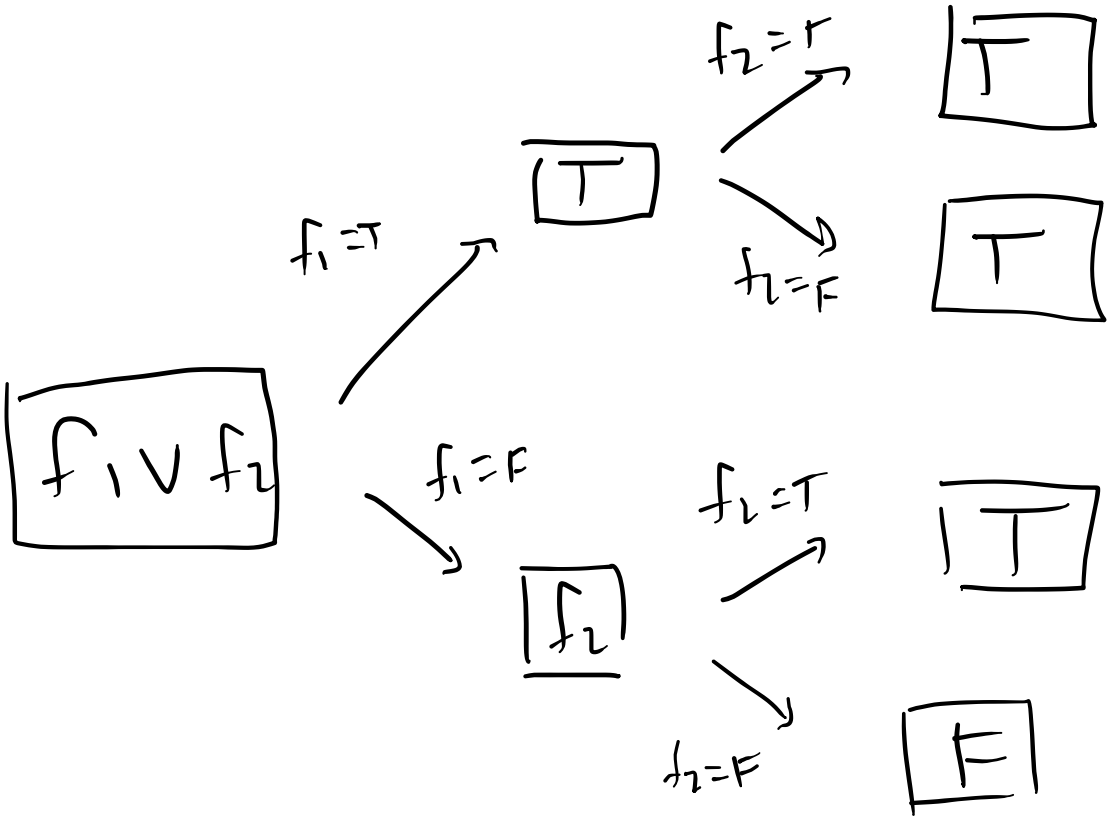
Decomposing WMC

- We can decompose a big WMC problem into two smaller ones (for any variable A)

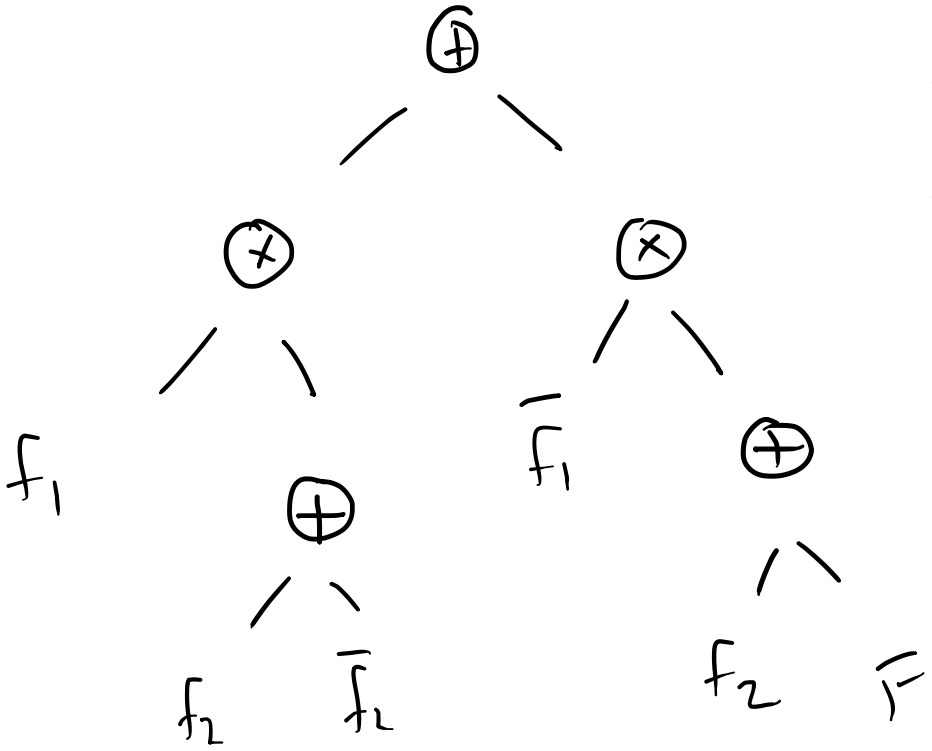
$$WMC(\varphi, w) = WMC(\varphi \mid A, w) \times w(A) + WMC(\varphi \mid \bar{A}, w) \times w(\bar{A})$$

f_1	f_2	$[\![(\varphi, w)]\!]v$
T	T	$0.1 * 0.3$
T	F	$0.1 * 0.7$
F	T	$0.9 * 0.3$
F	F	0

Solving WMC: Search



Solving WMC: Search Circuit

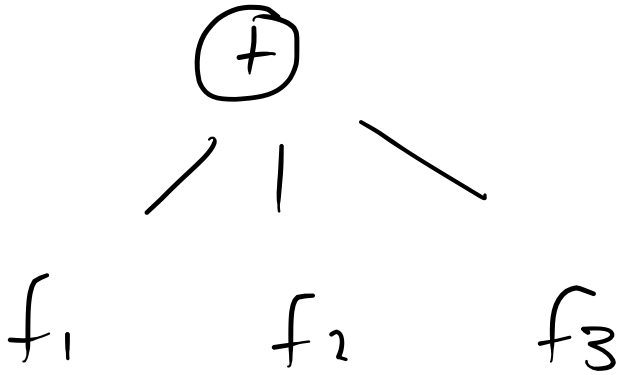


- To compute WMC, plug in weights for f1 and f2 to evaluate
- Inference complexity: size of circuit

Search circuit for $f_1 \vee f_2$

Not All Circuits Compute WMC

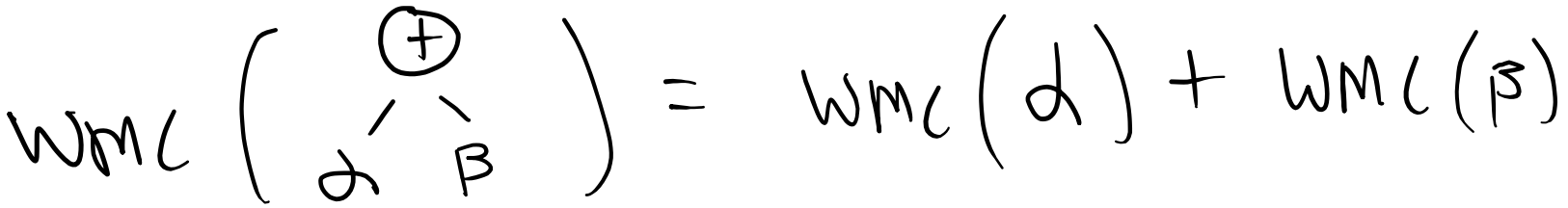
- Consider the following circuit for $f_1 \vee f_2 \vee f_3$



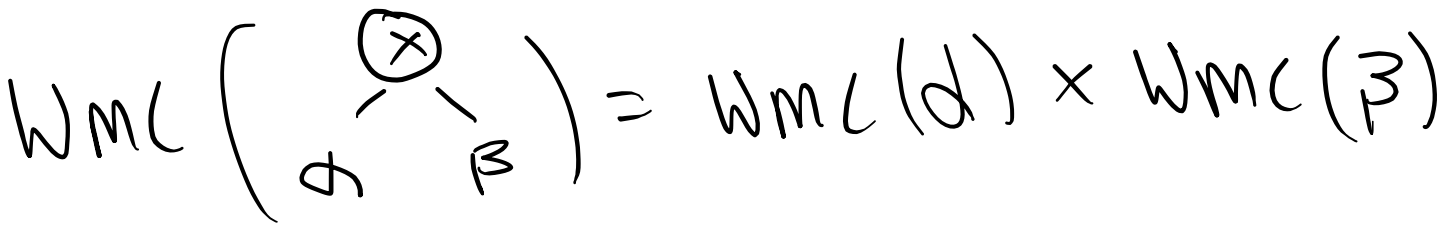
- What's wrong? It's *over counting*

Requirements for WMC Circuits

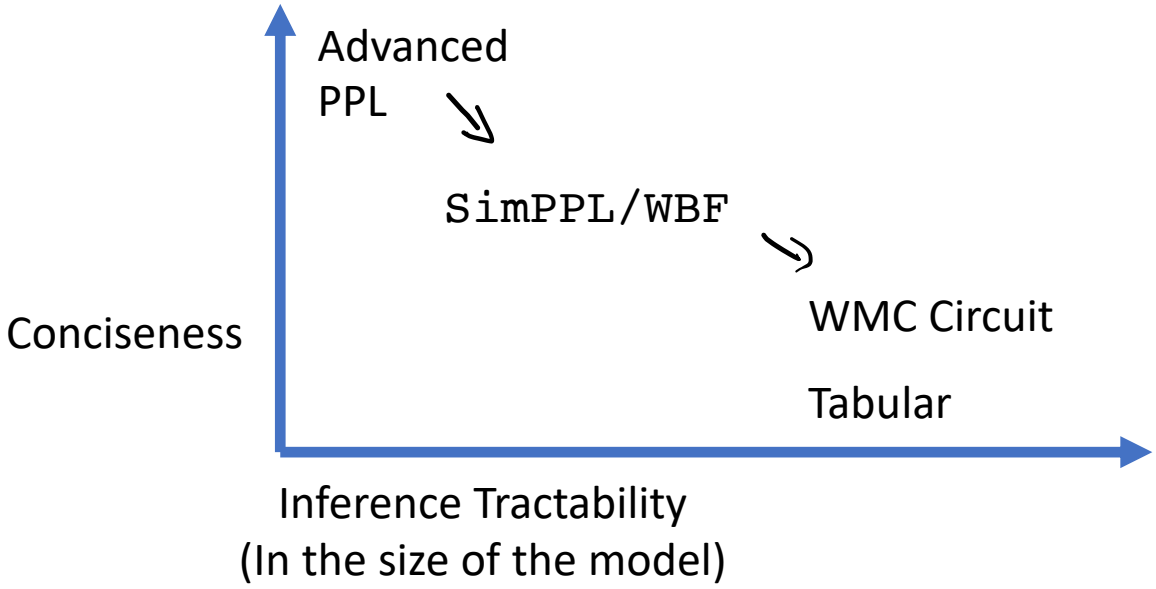
1. Sum nodes must respect:



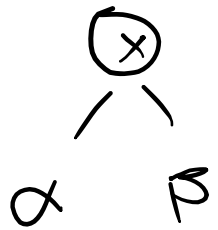
2. Product nodes must respect:



Popping Up


Advanced
PPL
SimPPL/WBF
WMC Circuit
Tabular
Conciseness
Inference Tractability
(In the size of the model)

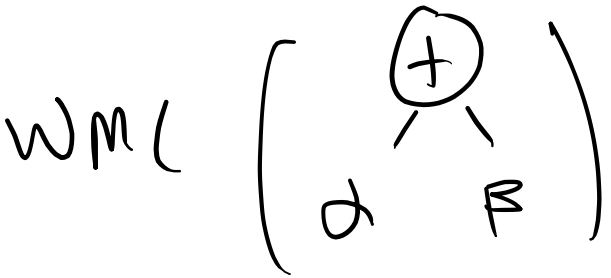
Sufficient Conditions for Product Decomposition

- To decompose  require that α and β do not

  reference the same variables
  - called *decomposability*

Sufficient Conditions for Sum Decomposition

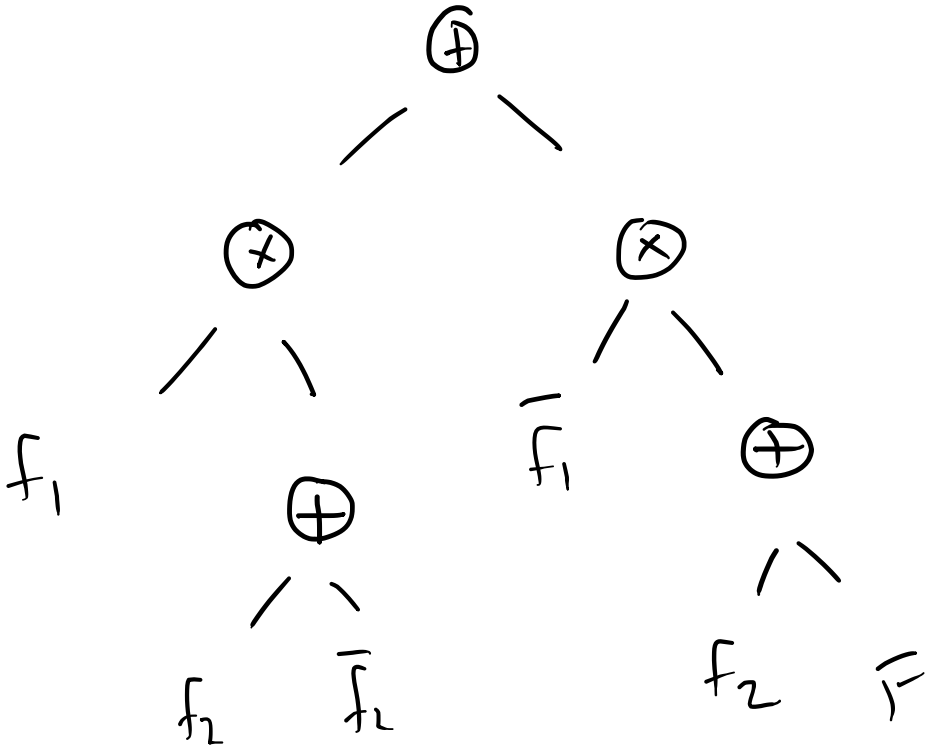
- To decompose



require that $\alpha \wedge \beta \models F$ (no overlapping worlds)

- Called *determinism*
  - Not a great name; much better name is *unambiguous*

WMC Circuit



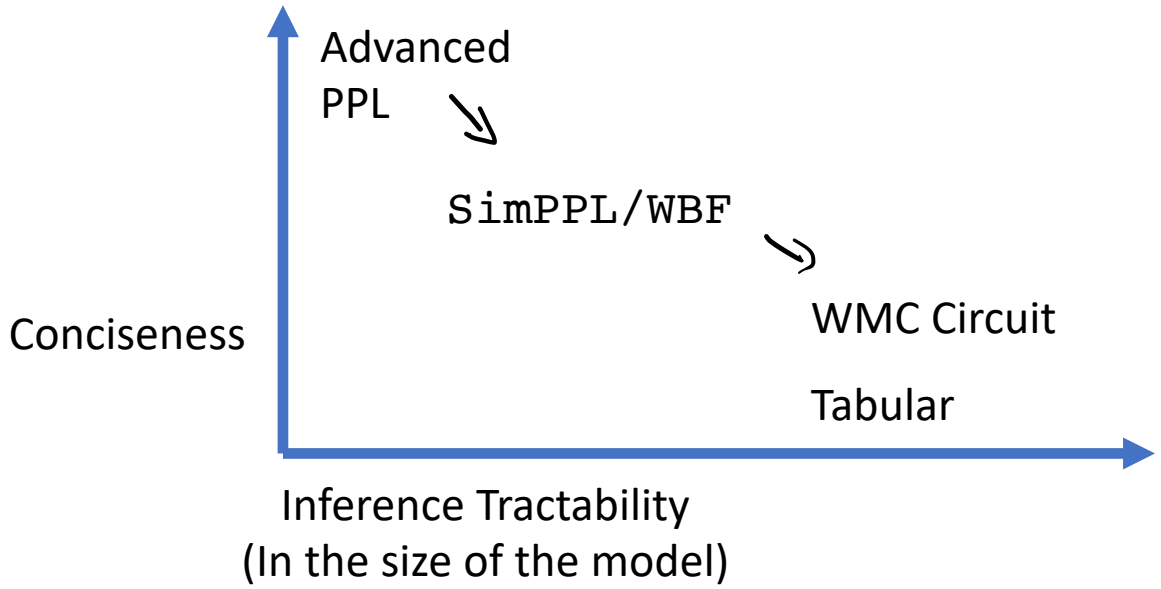
- Check Determinism
- Check decomposability

Sharing Structure

- Suppose we have a search circuit with *redundant sub-circuits*

Conclusion

- Inference via compilation is a powerful framework for PPL inference



Extra slides

Applications of Discrete Inference

- Systems reliability and diagnosis
  - Networks
  - Hardware
  - Supply chains
  - Etc…
- Text models
- Hybrid models
  - Many programs contain *discrete sub-programs*

Hardware Diagnosis

Scalability Challenge in Hardware

Network Reliability & Verification with PPLs

- Smolka, Steffen, et al. "Scalable verification of probabilistic networks." *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*. 2019.

- Gehr, Timon, et al. "Bayonet: probabilistic inference for networks." *ACM SIGPLAN Notices* 53.4 (2018): 586-602.

Discreteness is Hard

- Many advanced approximate inference methods rely on *program differentiability*