Integrating Probabilistic Reasoning with Constraint Satisfaction

IJCAI Tutorial #7
Instructor: Eric I. Hsu

July 17, 2011
http://www.cs.toronto.edu/~eihsu/tutorial7
Getting Started

- Discursive Remarks.
- Organizational Details.
- Motivation.
- The Actual Tutorial.
Bottom Line Up Front (BLUF)

Thesis: Constraint satisfaction and probabilistic inference are closely related at a formal level, and this relationship motivates new ways of applying techniques from one area to the other.

- Block A: Probabilistic inference and constraint satisfaction in terms of graphical models, and their formal correspondence. (about 2 hours.)
- Block B: Concrete applications of the formal correspondence. (1 hour.)
- Block C: Conclusions, Future Research Opportunities, Discussion. (0.5 hours.)
CSP’s directly encode (uniform) probability distributions over their own solutions.

\[
P (v_1, \ldots, v_n) = \frac{1}{Z} \prod_{c} f_c(v_1, \ldots, v_n)
\]

\[
P (v_1=\text{‘+’}) = \frac{1}{Z} \sum_{V:v_1=+} \prod_{c} f_c(+, \ldots, v_n)
\]

\[
f_c(v_1, \ldots, v_n) = \begin{cases} 
1, & \text{if } v_1, \ldots, v_n \text{ satisfies } \text{constraint } c; \\
0 & \text{otherwise}.
\end{cases}
\]
BLUF: Main Idea

Quasigroup with Holes Problem

Boolean Satisfiability Problem
BLUF: Concrete Applications

Constraint Program

List of Solutions

Maximum Likelihood Estimator

“Survey”: Probability for each variable to be fixed a certain way when sampling a solution at random.

\[ P(\Theta_v | \text{SAT}) \]
Core formal principles are not new, but there is a terrific opportunity to operationalize them.

Natural next step at the formal level is to emphasize pure optimization over specific pre-packaged methodologies.

Natural emphasis at the operational level is results-oriented, problem driven, highly empirical.

Natural target for technical improvement is to find profitable places to sacrifice accuracy for speed.
Organization: Required Background

- Will build from the ground up.
- <Audience survey>.
My level of experience is:

A. Student.
B. “Recent” graduate.
C. “Established.”
I would describe my main area of research as:

A. Constraint Satisfaction / Constraint Programming.

B. Probabilistic Models / Machine Learning.

C. Some other related field.

D. Some other mostly unrelated field.

E. I do not have an area of research yet.
For those who answered ‘A’ or ‘B’:

A. I know almost nothing about the other field.

B. I get the main idea and already know the basic techniques, and want to know the current state of affairs.

C. I actually have a lot of experience with the other field and am more interested in applications.
Organization: Scope

- Will adopt a biased (but accurate) perspective.
- Will actually try to cover terminology comprehensively.
- Limitation to discrete, finite domains.
- Examples will focus on Boolean Satisfiability (SAT).
Questions encouraged at any time!

Overstatements, citations

Treating the two areas interchangeably.

Coming and going, laptop use are fine, please just minimize noise.
Motivation

• Understanding
• Problem-Solving
Part I. Basic principles of constraint satisfaction and probabilistic reasoning over graphical models.

Part II. Advanced concepts in CSP and probabilistic reasoning.
Basic Probabilistic Reasoning and Constraint Satisfaction; Integration
The Factor Graph representation allows us to link Probabilistic Reasoning and CSP through a common notation, clarifying the equivalence between basic methods from both areas.
What is the sum of function outputs, over all configurations?
A Constraint Satisfaction Problem (CSP) is a triple \( \langle X, D, C \rangle \), where:

- \( X \) is a set of variables.
- \( D \) is a set of domains for the variables.
  - Here, all assumed to be identical, discrete, and finite.
- \( C \) is a set of functions ("constraints") mapping subsets of \( X \) to \( \{0,1\} \).
Constraint Satisfaction: Queries

- **Decision Problem:**
  - Does there exist any configuration of X (assignment to all the variables with values from their respective domains) such that CSP \(<X, D, C>\) evaluates to 1?
  - Satisfiable versus Unsatisfiable instances give different flavors to the decision problem.

- **Model Counting (#CSP, #SAT):**
  - For how many of the exponentially numerous possible configurations does the CSP evaluate to 1?

- **Maximization (WCSP, MAXSAT):**
  - Assign weights to constraints. Instead of evaluating to 0/1, CSP assigns score to a configuration consisting of sum of weights of satisfying constraints. Find configuration with maximum score.
Probabilistic Inference

Joint Probability Distribution

\[ [0,1] \]
(If more familiar with logical representations, you can view variables as propositions about the world.)

- **Marginalization**: what is the probability that variable $x_i$ holds value $v$? (i.e. what is the sum of outputs for all configurations containing this assignment?)

- **MAP (a.k.a. MPE)**: what is the most likely configuration of the variables (i.e., which input gives the greatest output?)
Explicitly iterate over all $O(2^n)$ configurations, check for output of 1.
Fix given variable to a given value; iterate over all $O(2^{n-1})$ configurations of the other variables, summing outputs.
Graphical Models: Localized Structure

Factor Graph: Global structure factors into local functions.
**Factor Graph Representation**

$$W_G(\vec{x}) = \prod_{f_a \in F} f_a(\vec{x})$$

**Factor** ("function") nodes connect to nodes for **variables** that appear in their scopes.

Generalizes other graphical models (Constraint Graph, MRF, Bayes Net, Kalman Filter, HMM…)
Factor Graphs: Extensions ("Tuples")

Notation for $f_2(x_2, x_3, x_4)$:

- **Scope** of $f_2$: $\{x_2, x_3, x_4\}$
- **Extension** to $f_2$: assignment to all the variables in its scope, i.e. $<0, 2, 1>$. Also known as a "tuple".
- **Projection** of a configuration onto $f_2$’s scope: yields assignments relevant to $f_2$. 
Nodes with no parents are associated with prior probability distributions, remaining nodes hold distributions that are conditioned on their parents.
Example: Markov Random Field

Each clique of nodes is associated with a potential function.
**Probabilistic Factor Graphs and Marginal Computation: The Partition Function**

The Probability of a Configuration:

\[ P(\vec{x}) = \frac{1}{\mathcal{N}} W_G(\vec{x}), \quad \text{where} \quad \mathcal{N} = \sum_{\vec{x}} W_G(\vec{x}) \]

The Marginal Computation:

\[ \theta_i(v_j) = \frac{1}{\mathcal{N}} \sum_{x_1} \cdots \sum_{x_{i-1}} \sum_{x_{i+1}} \cdots \sum_{x_n} W_G(\langle x_1, \ldots, x_{i-1}, v_j, x_{i+1}, \ldots, x_n \rangle) \]

Probability of particular variable holding particular value.
(a.k.a. proportion of total mass contributed by vectors containing this assignment)
(a.k.a. computing the \textit{partition function} \( \mathcal{N} \))
Returning to the Main Idea...

Input Configuration

\[\vec{x} \in \begin{cases} 42351 \\
12453 \\
12354 \end{cases} \ldots\]

Assignment to variables

Function

Constraint Satisfaction Problem

Output Range

\[\{0, 1\}\]

Whether problem is satisfied

\[\vec{x} \in \begin{cases} 42351 \\
12453 \\
12354 \end{cases} \ldots\]

Assignment to (random) variables

Probabilistic Model

\[\mathbb{R}^+ \cup \{0\}\]

Joint probability of assignment
Main Idea: Product Decomposition

Function Type

- Constraint Program
- Probabilistic Graphical Model

Internal Function Structure

\[ \bigoplus_{x_1} \bigoplus_{x_2} \cdots \bigoplus_{x_n} \bigotimes_{f_a \in F} f_a(x) \]

Algebraic Semiring

\[ \bigoplus : \lor \]
\[ \bigotimes : \land \]
\[ \bigoplus : + \]
\[ \bigotimes : \times \]
Factor Graph Representation

\[ W_G(\vec{x}) = \begin{bmatrix} \prod_{f \alpha \in F} f_\alpha(\vec{x}) \end{bmatrix} \]

\[ G : \]
Basic Operations: Search and Inference

- **Search:**
  Try different combinations of values for the variables.

- **Inference:**
  Propagate consequences of a variable choice; reason over constraint extensions.
Basic Operation: Distributing over Sum

\[ \theta_2(0) \propto \sum_{x_1} \sum_{x_3} \sum_{x_4} \sum_{x_5} (f_1(x_1) \cdot f_2(0) \cdot f_3(x_1, 0, x_3) \cdot f_4(x_3, x_4) \cdot f_5(x_3, x_5)) \]

\[ = f_2(0) \cdot \left( \sum_{x_1} f_1(x_1) \cdot \left( \sum_{x_3} f_3(x_1, 0, x_3) \cdot \left( \sum_{x_4} f_4(x_3, x_4) \right) \cdot \left( \sum_{x_5} f_5(x_3, x_5) \right) \right) \right) \]

\[ \theta_3(0) \propto \left( \sum_{x_1} \sum_{x_2} f_3(x_1, x_2, 0) \cdot f_1(x_1) \cdot f_2(x_2) \right) \cdot \left( \sum_{x_4} f_4(0, x_4) \right) \cdot \left( \sum_{x_5} f_5(0, x_5) \right) \]

Note the structure of the parenthesized expressions—dynamic programming!
Basic Operation: Node Merging

“Node Merging”
Basic Operation: Dualization

Factors become variables defined over their possible extensions.
Variables become 0/1-factors enforcing consistency between extensions.
Scoring factors introduced to simulate factors from the primal graph.

(“Extension”: assignment to all the variables in a function’s scope)
Algorithm: Variable Elimination on a Tree

We will send **messages** between functions and variables, and variables and functions.

Messages are themselves *functions* defined over the possible values of the participating variable.

1. Initialize arbitrary node as the root of the tree. Choose any ordering that begins with leaves and ends at the root.
2. Initialize messages at leaves.
   - Variable leaf nodes: identity message.
   - Function leaf nodes: send their own functions as messages.
Algorithm: Variable Elimination on a Tree

- **Variable-to-function message** represents totality of “downstream influence” on the variable by all other functions.

- **Function-to-variable message** represents totality of “downstream influence” on the variable by summarizing all downstream functions and variables as a single function.

$$\mu_{x_i \to f_a}(v) = \prod_{f_b \in \eta_i \setminus \{f_a\}} \mu_{f_b \to x_i}(v)$$ (8)

$$\mu_{f_a \to x_i}(v) = \sum_{z_a: z_a(x_i) = v} \left( f_a(z_a) \cdot \prod_{x_j \in \sigma_a \setminus \{x_i\}} \mu_{x_j \to f_a}(z_a(x_j)) \right)$$ (9)
Algorithm: Variable Elimination on a Tree

3. Apply update rules according to ordering from leaves to root.
4. Apply update rules according to reverse ordering from root to leaves.
5. Calculate final marginals according to formula very similar to variable-to-function update rule; normalize.

Calculates all marginals at once.
Example: Variable Elimination on a Tree

Consider marginal probability that $x_3$ has value 0:

$$
\mu_{f_3 \rightarrow x_3}(0) = \sum_{x_1} \sum_{x_2} (f_3(x_1, x_2, 0) \cdot \mu_{x_1 \rightarrow f_3}(x_1) \cdot \mu_{x_2 \rightarrow f_3}(x_2))
$$

$$
\mu_{f_4 \rightarrow x_3}(0) = \sum_{x_4} (f_4(0, x_4) \cdot \mu_{x_4 \rightarrow f_4}(x_4))
$$

$$
\mu_{f_5 \rightarrow x_3}(0) = \sum_{x_5} (f_5(0, x_5) \cdot \mu_{x_5 \rightarrow f_5}(x_5))
$$
Example: Variable Elimination on a Tree

\[
\mu_{f_3 \rightarrow x_3}(0) = \sum_{x_1} \sum_{x_2} (f_3(x_1, x_2, 0) \cdot f_1(x_1) \cdot f_2(x_2))
\]

\[
\mu_{f_4 \rightarrow x_3}(0) = \sum_{x_4} (f_4(0, x_4) \cdot 1)
\]

\[
\mu_{f_5 \rightarrow x_3}(0) = \sum_{x_5} (f_5(0, x_5) \cdot 1)
\]

\[
\theta_3(0) = \frac{1}{N} \cdot \left( \sum_{x_1} \sum_{x_2} f_3(x_1, x_2, 0) \cdot f_1(x_1) \cdot f_2(x_2) \right) \cdot \left( \sum_{x_4} f_4(0, x_4) \right) \cdot \left( \sum_{x_5} f_5(0, x_5) \right)
\]
Basic Operation: Distributing over Sum

\[ \theta_3(0) = \frac{1}{N} \cdot \left( \sum_{x_1} \sum_{x_2} f_3(x_1, x_2, 0) \cdot f_1(x_1) \cdot f_2(x_2) \right) \cdot \left( \sum_{x_4} f_4(0, x_4) \right) \cdot \left( \sum_{x_5} f_5(0, x_5) \right) \]

Note the structure of the parenthesized expressions—dynamic programming!
What if the factor graph is not a tree?

- Make it into a tree. (Exact, Convergent).
  - Full **Variable Elimination** (Belief Propagation)
    - Cluster-Tree Elimination.
What if the factor graph is not a tree?

- Make it into a tree. (Exact, Convergent).
  - Full **Variable Elimination** (Belief Propagation)
  - **Cluster-Tree Elimination**.
What if the factor graph is not a tree?

- Make it into a tree. (Exact, Convergent)
  - Full Variable Elimination (Belief Propagation)
    - Cluster-Tree Elimination.
  - Cycle-Cutset.
  - Recursive Conditioning.
  - Bucket Elimination, Node-Splitting.

- Pretend that the graph is a tree, even if it isn’t.
  (Approximate, Non-Convergent)
  - Loopy Belief Propagation.
  - (Variations).
Alternative Method: Sampling

- Choose an arbitrary initial configuration.
- For some number of iterations:
  - For each variable in the graph (according to some ordering):
    - Sample a new value for the variable according to the current values of its neighbors.
- On conclusion, marginal probability of a variable assignment is the fraction of iterations under which given variable holds given value.

“Ergodicity” ensures that we can explore entire configuration space.
Summary: Marginals over Graphical Models

- **Dualization:**

- **Node-Merging:**

- Variable elimination as node-merging.
- Belief propagation as variable elimination.
- Loopy belief propagation as “wrongful” belief propagation.

(Same goes for inference in the sense of CSP!)
Basic Constraint Satisfaction

- Search (Sum):
  - Backtracking (depth-first) search to build up a satisfying configuration.
  - Failure to build such a configuration upon exhausting entire space indicates unsatisfiability.

- Inference (Product):
  - During backtracking search, propagate consequences of an individual variable assignment.
  - (Can also add constraints during search).
  - Can also just solve a problem entirely by inference.
Basic Backtracking Search

Data: constraint satisfaction problem \( G = (X, F) \).
Result: solution, or ‘\( \perp \)’ (“unsatisfiable”).

1. \textbf{return} \textsc{Recursive-Backtracking}(\emptyset, G).

2. \textbf{subroutine} \textsc{Recursive-Backtracking}(assignments, csp)

3. \textbf{begin}

4.  \hspace{1em} \textbf{if} (assignments violate csp) \textbf{then return} ‘\( \perp \)’.

5.  \hspace{1em} \textbf{if} (\text{\#}assignments = \( n \)) \textbf{then return} assignments.

6.  \hspace{1em} var \leftarrow \textsc{Choose-Variable}(assignments, csp).

7.  \hspace{1em} \textbf{foreach} \textit{val} \textbf{in} \textsc{Order-Values}(var, assignments, csp) \textbf{do}

8.  \hspace{2em} result \leftarrow \textsc{Recursive-Backtracking}(assignments \cup \{\text{var = val}\}, csp).

9.  \hspace{2em} \textbf{if} result = ‘\( \perp \)’ \textbf{then continue} \textbf{else return} assignments.

10. \textbf{end}

11. return ‘\( \perp \)’.

12. \textbf{end}
Quasigroup with Holes (QWH) Problem

- Latin Square:
  - $d$ rows by $d$ columns
  - Cell variables $\in \{1, \ldots, d\}$
  - No repeats in any row or column: “alldiff” constraint

$$d = 5$$
Quasigroup with Holes (QWH) Problem

- **Latin Square:**
  - $d$ rows by $d$ columns
  - Cell variables $\in \{1, \ldots, d\}$
  - No repeats in any row or column: “alldiff” constraint

- **QWH Problem:**
  - Remove a percentage of the holes from a valid Latin Square.

(Kautz et al., ’01)
Factor Graph for QWH

(Unary constraints represent the fixed values.)
Boolean Satisfiability (SAT) in CNF

- Variables are Boolean (domain \{0,1\}).
- **Literals** represent the assertion or negation of a variable.
  - i.e. \(X_1\), or else \(\neg X_1\); “positive” or “negative” literal for \(X_1\).
- Constraints consist of **clauses**, disjunctions of literals.
  - i.e. \((X_1 \lor \neg X_2 \lor X_3)\). At least one literals must be satisfied.
- **CNF** = “Conjunctive Normal Form”.

\[
\begin{align*}
C_1 &= (x_1 \lor x_2 \lor \neg x_3) \\
C_2 &= (\neg x_1 \lor \neg x_2 \lor \neg x_4) \\
C_3 &= (x_1 \lor \neg x_2 \lor \neg x_5) \\
C_4 &= (\neg x_1 \lor x_3 \lor \neg x_4) \\
C_5 &= (x_1 \lor \neg x_3 \lor x_5) \\
C_6 &= (x_1 \lor \neg x_4 \lor x_5) \\
C_7 &= (x_2 \lor x_4 \lor x_5) \\
C_8 &= (\neg x_3 \lor x_4 \lor \neg x_5)
\end{align*}
\]

\[
\bigwedge_{a=1}^{8} C_a = \bigvee_{x_1 x_2 x_3} C_1 \wedge \left( \bigvee_{x_4} C_2 \wedge C_4 \wedge \left( \bigvee_{x_5} C_3 \wedge C_5 \wedge C_6 \wedge C_7 \wedge C_8 \right) \right)
\]
Factor Graph for SAT

(Solid or dotted lines indicate that variable appears positively or negatively in clause.)
Basic Backtracking Search

Two most important issues: order variables/values, perform propagation (inference)

Data: constraint satisfaction problem $G = (X, F)$.
Result: solution, or ‘⊥’ (“unsatisfiable”).

1. return RECURSIVE-BACKTRACKING($\emptyset$, $G$).

2. subroutine RECURSIVE-BACKTRACKING(assignments, csp)
3. begin
4. if (assignments violate csp) then return ‘⊥’.
5. if ($|assignments| = n$) then return assignments.
6. var $\leftarrow$ CHOOSE-VARIABLE(assignments, csp).
7. foreach val in ORDER-VALUES(var, assignments, csp) do
8.   result $\leftarrow$ RECURSIVE-BACKTRACKING(assignments $\cup \{var = val\}$, csp).
9.   if result = ‘⊥’ then continue else return assignments.
10. end
11. return ‘⊥’.
12. end
Prune the domains of variables to include only those that are supportable by some assignment to the other variables.

- **Node consistency.** Check variable assignment against zero other variables. ("Individual consistency").

- **(Generalized) Arc consistency.** Check variable assignment against any one other variable. ("Pairwise consistency.")
  - Special case for SAT: "unit propagation".

- Many more...
II. Advanced Concepts from Probabilistic Reasoning and Constraint Satisfaction
Overview of CSP/Prob. Inference

- Contemporary research in constraint satisfaction:
  - Random problems: backbones and backdoor.
  - Constraint propagators.
  - Data structures and engineering.
  - Clause learning / No-Good learning.

- Contemporary research in probabilistic graphical models:
  - Marginal polytope.
  - Emphasis on pure (numerical) optimization.
  - Function propagators.

- Comparison.
Motivation: Discovering Structure

- Certain variables are much more important than the rest.
  - Combinatorial cores can be small.
  - Remainder is extraneous.
Motivation: Discovering Structure

- Certain variables are much more important than the rest.
  - Combinatorial cores can be small.
  - Remainder is extraneous.

- Can’t just rely on graph structure in the general case.
Interesting Features of Random CSPs

- Combinatorial abstraction for real problems
- Small, tightly constrained cores
  - Backdoors/Backbones (Williams et al., ‘03)
  - Restarts
- Heavy-Tailed Behavior
  - High sensitivity to variable ordering
    (Gomes et al., ‘00; Hulubei et al., ‘04)

Runtimes over Multiple Trials

Frequency
Survey Propagation and the Phase Transition in Satisfiability

- Random problems are of interest to statistical physics and other applications.
- Hard random problems can be viewed in terms of backbone and backdoor variables.
Contemporary Research in CSP

- Propagators.
- Modeling /Applications.
- Clause Learning/No-Good Learning.
- Restarts.
- Data Structures and Algorithms.
Prob. Inference as Pure Optimization

- Computing sets of marginals can be viewed as an optimization problem constrained by marginal polytope.
- "Survey": set of marginal estimates.
Gibbs Free Energy

\[ G(q(\cdot), p(\cdot)) = - \sum_{\bar{x}} q(\bar{x}) \log p(\bar{x}) + \sum_{\bar{x}} q(\bar{x}) \log q(\bar{x}) + \log \mathcal{N} \]

- We want “hypothesized” \textbf{q-distribution} to match “true” \textbf{p-distribution}.
- First term: \textbf{Mean Free Energy}; make them match.
- Second term: \textbf{Entropy}; don’t be too extreme.
- (Third term: Helmholtz term, for normalization.)
Mean Field Approximation

\[ q_{\mathcal{MF}}(\vec{x}) = \prod_i q_i(x_i) \]

\[
G_{\mathcal{MF}}(q_{\mathcal{MF}}(\cdot), p(\cdot)) = -\sum_{\vec{x}} \prod_i q_i(x_i) \log(\prod_a f_a(\vec{z}_a)) + \sum_{\vec{x}} \prod_i q_i(x_i) \log(\prod_i q_i(x_i)) + \log N
\]

\[
= -\sum_a \sum_{\vec{z}_a} \left( \prod_i q_i(z_{a}(x_i)) \right) \log f_a(\vec{z}_a) + \sum_i \sum_{x_i} \left( \prod_i q_i(x_i) \right) \log q_i(x_i) + \log N
\]

- No correlation: probability of a given extension is the product of the individual marginal probabilities of its constituent variable assignments.

- (Node consistency!)
Bethe Approximation (BP)

- Pairwise correlation: hypothesize distributions over extensions, and distributions over individual variables.
- Extension distributions must marginalize to individual variable distributions, one variable at a time.
- (Generalized Arc Consistency!)

\[
G_B(q_B(.), p(.)) = - \sum_{\bar{z}} \prod_a q_a(z_a) \log(\prod_a f_a(z_a)) + \sum_{\bar{z}} \prod_a q_a(z_a) \log(\prod_i q_a(z_a)) + \log \mathcal{N} \\
= - \sum_a \sum_{z_a} q_a(z_a) \log f_a(z_a) + \sum_a \sum_{z_a} q_a(z_a) \log q_a(z_a) + \log \mathcal{N}
\]
CSP and Marginalization as Optimization

CSP:

\[
\max_{\mathcal{I}} \prod_{f_a \in \mathcal{F}} \left[ \sum_{z_a} \left( \prod_{x_i \in \sigma_a} I_i[z_a(x_i)] \right) \cdot f_a(z_a) \right]
\]

such that \( \forall i, j : I_i[v_j] \in \{0, 1\} \) and \( \forall i : \sum_{v_j} I_i[v_j] = 1. \)
CSP and Marginalization as Optimization

Probabilistic Marginalization:

\[
\max_{\Theta} \prod_{f_a \in F} \left[ \sum_{z_a} \left( \theta_a(z_a) \cdot f_a(z_a) \right) \right]
\]

such that \( \forall i, j : \theta_i[v_j] \in [0, 1] \), \( \forall i : \sum_{v_j} \theta_i[v_j] = 1 \),

\( \forall a, z_a : \theta_a[z_a] \in [0, 1] \), \( \forall a : \sum_{z_a} \theta_a[z_a] = 1 \),

and \( \text{CONSISTENT}(\Theta) \).
CSP and Marginalization as Optimization

CSP:

$$\max_{I} \prod_{f_a \in F} \left[ \sum_{z_a} \left( \left( \prod_{x_i \in \sigma_a} I_i[z_a(x_i)] \right) \cdot f_a(z_a) \right) \right]$$

such that \( \forall i, j : I_i[v_{ij}] \in \{0, 1\} \) and \( \forall i : \sum_{v_{ij}} I_i[v_{ij}] = 1 \).

Marginalization:

$$\max_{\Theta} \prod_{f_a \in F} \left[ \sum_{z_a} (\theta_a(z_a) \cdot f_a(z_a)) \right]$$

such that \( \forall i, j : \theta_i[v_{ij}] \in [0, 1], \ \forall i : \sum_{v_{ij}} \theta_i[v_{ij}] = 1 \), \( \forall a, z_a : \theta_a[z_a] \in [0, 1], \ \forall a : \sum_{z_a} \theta_a[z_a] = 1 \),

and \( \text{CONSISTENT}(\Theta) \).
Contemporary Research in Prob. Inference

- Optimization.
- Modeling / Applications.
- Algorithms.
- “Propagators”.
- (Learning? a.k.a. column generation, cutting planes)
Marginalization is not just “similar” to constraint satisfaction, it just is constraint satisfaction under a relaxed space.

Duality, node-merging, and adding constraints during solving, yield corresponding techniques.