

8th Week



Constraint Satisfaction Problems and Counting Functions

Synopsis.

- Counting Functions and #P
- Counting Constraint Satisfaction problems
- Signatures
- Approximate #CSPs
- Dichotomy Theorems

May 28, 2018. 23:59

Course Schedule: 16 Weeks

Subject to Change

- **Week 1:** Basic Computation Models
- **Week 2:** NP-Completeness, Probabilistic and Counting Complexity Classes
- **Week 3:** Space Complexity and the Linear Space Hypothesis
- **Week 4:** Relativizations and Hierarchies
- **Week 5:** Structural Properties by Finite Automata
- **Week 6:** Type-2 Computability, Multi-Valued Functions, and State Complexity
- **Week 7:** Cryptographic Concepts for Finite Automata
- **Week 8:** Constraint Satisfaction Problems
- **Week 9:** Combinatorial Optimization Problems
- **Week 10:** Average-Case Complexity
- **Week 11:** Basics of Quantum Information
- **Week 12:** BQP, NQP, Quantum NP, and Quantum Finite Automata
- **Week 13:** Quantum State Complexity and Advice
- **Week 14:** Quantum Cryptographic Systems
- **Week 15:** Quantum Interactive Proofs
- **Week 16:** Final Evaluation Day (no lecture)

YouTube Videos

- This lecture series is based on numerous papers of **T. Yamakami**. He gave **conference talks (in English)** and **invited talks (in English)**, some of which were video-recorded and uploaded to YouTube.
- Use the following keywords to find a playlist of those videos.
- **YouTube search keywords:**
Tomoyuki Yamakami conference invited talk playlist



Conference talk video



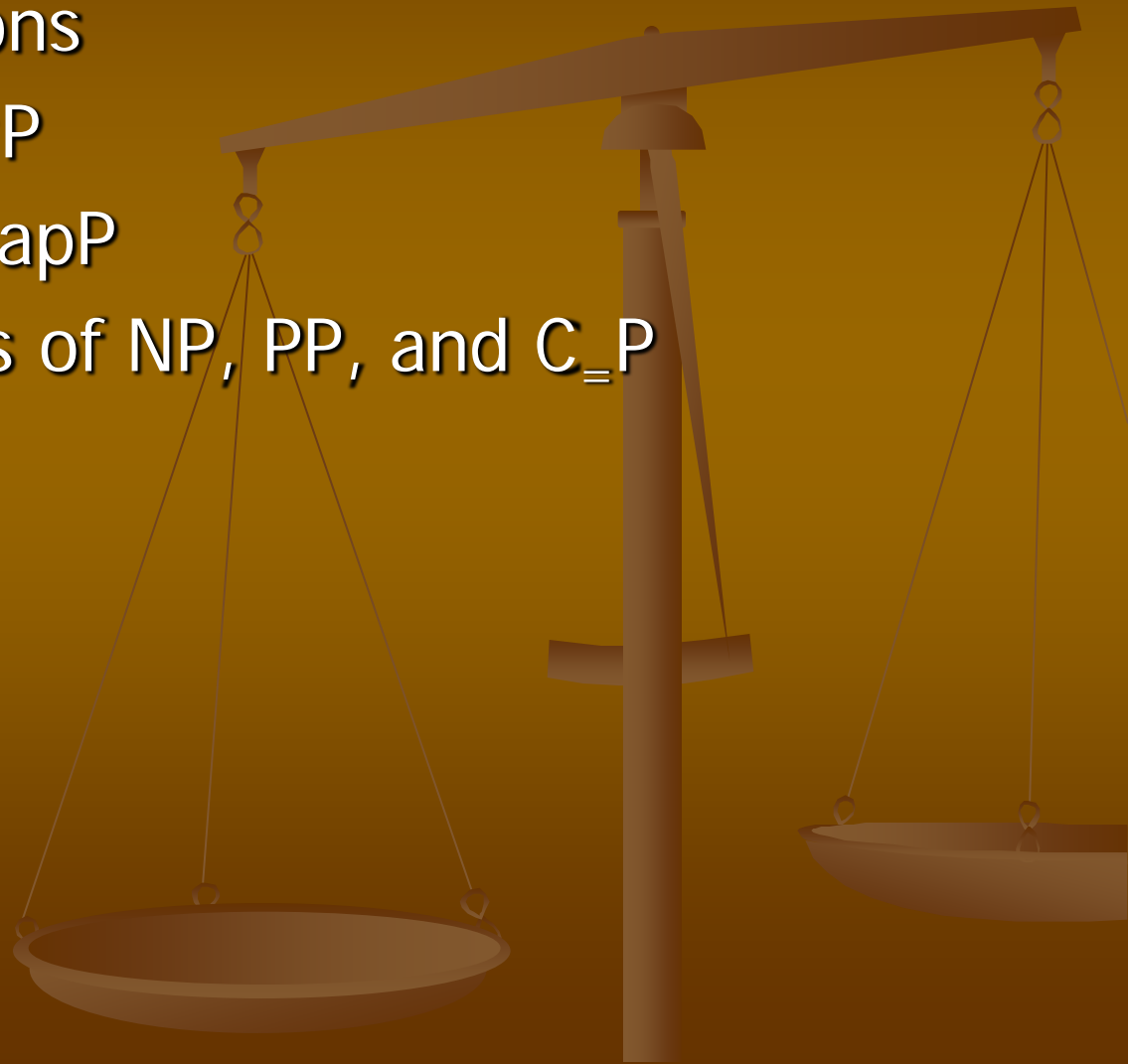
Main References by T. Yamakami



- ✎ **T. Yamakami.** Approximate counting for complex-weighted Boolean constraint satisfaction problems. Information and Computation 219, 17-38 (2012)
- ✎ **T. Yamakami.** A dichotomy theorem for the approximate counting of complex-weighted bounded-degree Boolean CSPs. Theoretical Computer Science 447, 120-135 (2012)
- ✎ **T. Yamakami.** Approximation complexity of complex-weighted degree-two counting constraint satisfaction problems. Theoretical Computer Science 461, 86-105 (2012)
- ✎ **T. Yamakami.** Constant unary constraints and symmetric real-weighted counting constraint satisfaction problems. Theory of Computing Systems 55, 170-201 (2014)

I. Counting Functions

1. Counting Functions
2. Function Class #P
3. Function Class GapP
4. Characterizations of NP, PP, and C=P



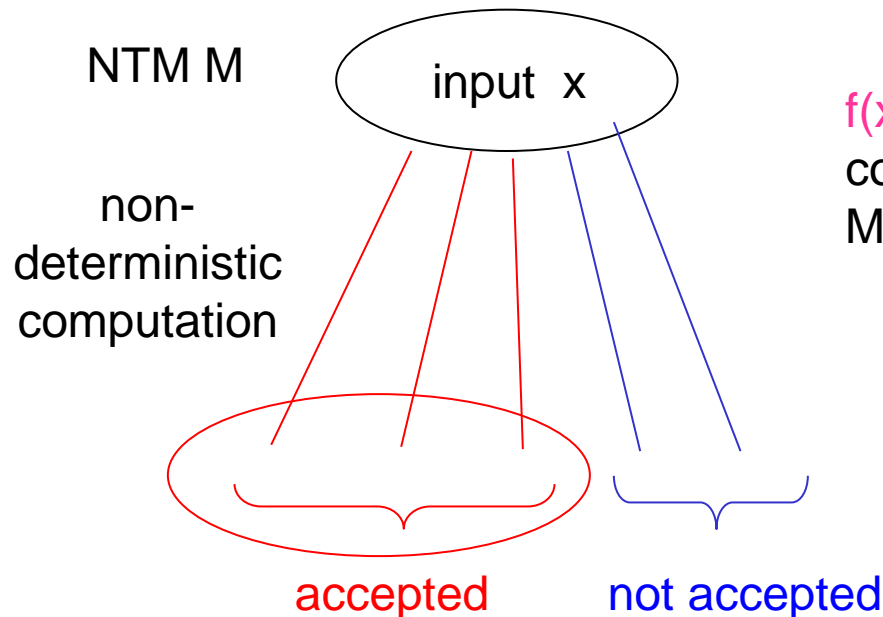
Counting Functions

- Like search and optimization problems, **counting problems** are quite important in practice.
- Those counting problems can be treated as functions.
- In general, a **counting function** is a function that solves a certain counting problem.
- (Example) **Permanent**
 - instance: a non-negative integer $n \times n$ matrix A
 - output: the permanent of A

$$\det \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = a_{11}a_{22} - a_{12}a_{21} \quad \text{perm} \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = a_{11}a_{22} + a_{12}a_{21}$$

Function Class #P

- To cope with counting functions, Valiant (1979) introduced a counting function class, called #P.
- A function $f: \Sigma^* \rightarrow \mathbb{N}$ is in #P \Leftrightarrow there exists a polynomial-time NTM M such that, for every input $x \in \Sigma^*$, $f(x)$ = the number of all accepting computation paths of M on x .



$f(x)$ = # of accepting computation paths of M on input x

Such an NTM is often called a counting machine.

#P Characterizations of NP, PP, C=P

- #P functions are quite useful to characterize many classes of decision problems.
- L be any decision problem.
- $L \in \mathbf{NP} \Leftrightarrow$ there is a function $f \in \#P$ such that, for every input x ,
 - $x \in L \Leftrightarrow f(x) > 0$
- $L \in \mathbf{PP} \Leftrightarrow$ there are two functions $f, g \in \#P$ such that, for every input x ,
 - $x \in L \Leftrightarrow f(x) > g(x)$
- $L \in \mathbf{C=P} \Leftrightarrow$ there are two functions $f \in \#P, g \in \mathbf{FP}$ such that, for every input x ,
 - $x \in L \Leftrightarrow f(x) = g(x)$

Function Class GapP

- A function $f: \Sigma^* \rightarrow \mathbb{N}$ is in **GapP** \Leftrightarrow there exists a polynomial-time NTM M such that, for every input $x \in \Sigma^*$, $f(x) =$ the number of all accepting computation paths of M on x **minus** the number of all rejecting computation paths of M on x .

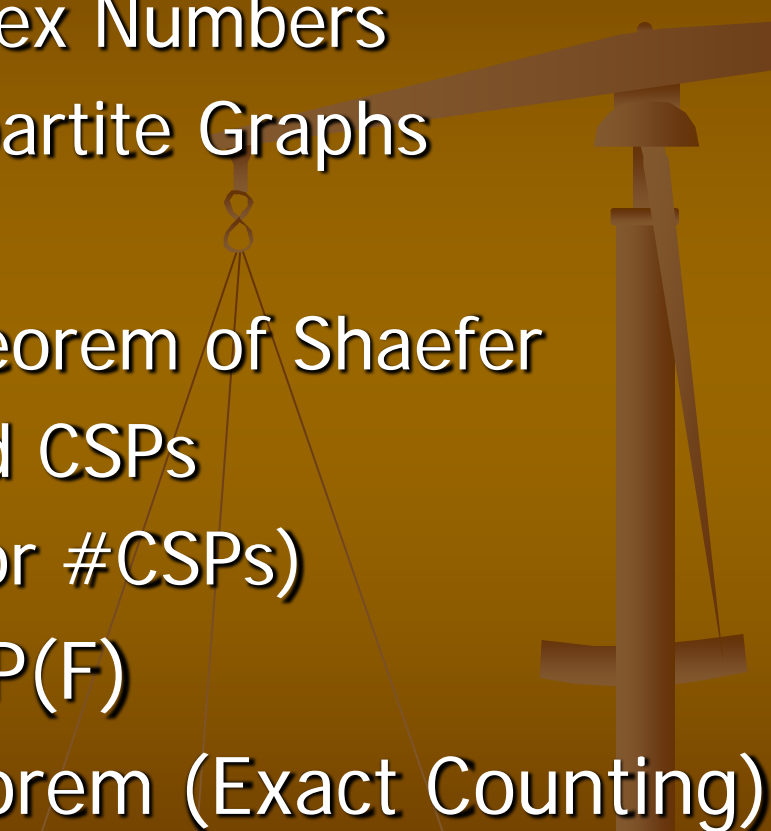
❖ An alternative definition

- A function $f: \Sigma^* \rightarrow \mathbb{Z}$ is in **GapP** \Leftrightarrow there are two functions $g, h \in \#P$ such that, for every input $x \in \Sigma^*$, $f(x) = g(x) - h(x)$.
- We express this definition as **GapP = #P - #P**.

GapP Characterizations of PP, C=P

- Here, we give GapP characterizations of the complexity classes PP and C=P.
- L be any decision problem.
- $L \in \text{PP} \Leftrightarrow$ there is a function $f \in \text{GapP}$ such that, for every input x ,
 - $x \in L \Leftrightarrow f(x) > 0$
- $L \in \text{C=P} \Leftrightarrow$ there is a function $f \in \text{GapP}$ such that, for every input x ,
 - $x \in L \Leftrightarrow f(x) = 0$
- (*) co-C=P will be discussed in Week 12 in relation to nondeterministic quantum computation.

II. Constraint Satisfaction Problems

1. CSPs with Complex Numbers
 2. Expressed by Bipartite Graphs
 3. Decision CSPs
 4. Classification Theorem of Shaefer
 5. Worlds of NP and CSPs
 6. Counting CSPs (or #CSPs)
 7. Visualizing #CSP(F)
 8. Dichotomy Theorem (Exact Counting)
- 

CSPs have Appeared in Many Fields

- Generally speaking, a **constraint satisfaction problem (CSP)** takes **two types** of items as an input:
 - **variables**: x_1, x_2, \dots, x_n
 - **constraints**: $f_1(x_6, x_3, x_5), f_2(x_3, x_6), \dots$
- Note that many existing real-life problems can be expressed in the forms of CSPs.
- CSPs have been studied extensively in many fields, including:
 - artificial intelligence
 - database query evaluation
 - type inference
 - scheduling
 - graph theory
 - statistical physics
- **(Open Problem)** Can we efficiently solve all CSPs?

CSPs with Complex Numbers

- Standard **CSPs** mostly deal with strings or integers.
- Here, we want to discuss CSPs dealing with arbitrary complex numbers.
- **How can we handle real numbers or even complex numbers?**
- There have been several ways to deal with complex numbers in computational complexity theory.
- Here, we treat complex numbers as “**objects**” rather than a pair of “**binary series.**”
- In this way, we perform natural operations (such as, $+$, \div , \times , etc.) on such objects. Each of those operations are assumed to take a **unit time** (i.e. one step).

Extension of FP and #P to Complex Numbers

- By allowing **arbitrary complex numbers** to use, we naturally extend the standard function classes **FP** and **#P** to function classes handling complex numbers.
- Let C denote the set of all complex numbers.
 1. FP_C = the set of all **complex** functions computable in polynomial time
 2. $\#P_C$ = the set of all **complex** functions computable by counting machines in polynomial time

Example: Boolean Variables and Formulas

- We begin with **SAT**, which is a simple example of CSP.
- Our **input instance**: a logical formula **F** in CNF

$$F = (x_1 \vee x_4 \vee x_3) \wedge (x_2 \vee x_4) \wedge (x_3 \vee x_2 \vee x_4)$$

$OR_3(x_1, x_4, x_3)$ $OR_2(x_2, x_4)$ $OR_3(x_3, x_2, x_4)$

Functional expression

Equivalently, $F = OR_3(x_1, x_4, x_3) \cdot OR_2(x_2, x_4) \cdot OR_3(x_3, x_2, x_4)$

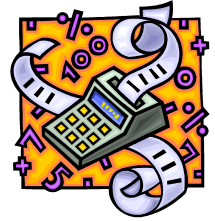
- **Our Possible Questions about the formula F:**
 - Is there any satisfying (variable) assignment for **F**?
 - How many satisfying (variable) assignments for **F** are there?
 - Can we approximate such a number?

Decision CSP

Exact counting CSP

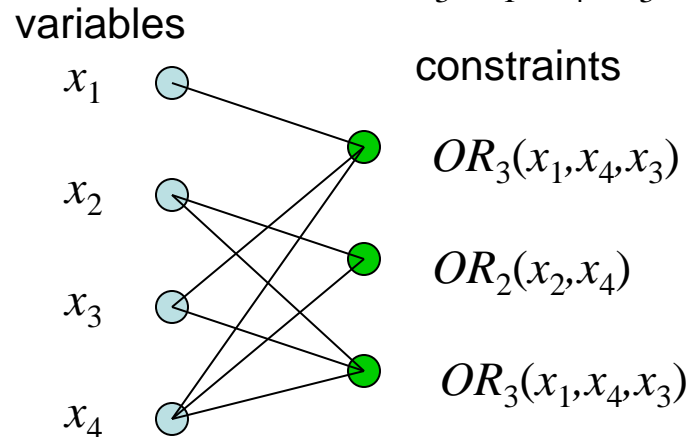
Approximate counting CSP

Expressed by Bipartite Graphs



- We can express the same F using a bipartite graph.

$$F = OR_3(x_1, x_4, x_3) \cdot OR_2(x_2, x_4) \cdot OR_3(x_3, x_2, x_4)$$



Boolean variables: $\{x_1, x_2, x_3, x_4\}$

Unweighted constraints:

$\{OR_3(x_1, x_4, x_3), OR_2(x_2, x_4), OR_3(x_1, x_4, x_3)\}$

- Is there any satisfying (variable) assignment for F ?

$$\exists x_1, x_2, x_3, x_4 \in \{0, 1\} \text{ s.t. } OR_3(x_1, x_4, x_3) OR_2(x_2, x_4) OR_3(x_3, x_2, x_4) \neq 0$$

- How many satisfying (variable) assignments for F ?

$$\# \text{ of such assignments} = \sum_{x_1, x_2, x_3, x_4 \in \{0, 1\}} OR_3(x_1, x_4, x_3) \cdot OR_2(x_2, x_4) \cdot OR_3(x_3, x_2, x_4)$$

Decision CSPs I

- The decision version of unweighted CSPs (or **decision CSPs**) are all **NP problems**.
- In the study of CSPs, one research direction is to **restrict a type of their constraints**, instead of allowing all constraints.
- Let **F** be any set of constraints. This set **F** is called a **constraint language** in some literature.
- **Decision CSP: CSP(F)**
 - **Instance:**
 - a set of variables
 - a set of constraints **in F**
 - **Question:**
 - Is there any truth assignment that make all constraints true?

Decision CSPs II

- There are a number of decision CSPs.
- Here, we give a few examples.
- **Examples of CSP(F)'s**
 - SAT (satisfiability problem)
 - COLORABILITY (colorability problem)
 - VERTEX-COVER (vertex cover problem)
- **3-Colorability Problem**
 - **instance:**
 - **variables:** a set V of vertices in a graph $G=(V,E)$
 - **constraints:** $NEQ(x,y)$ (inequality)
 - **question:**
 - is there any coloring $f: V \rightarrow \{1,2,3\}$ s.t. $NEQ(f(v),f(w))$ for all $(v,w) \in E$?

Classification Theorem of Schaefer I

- There has been a large volume of work on CSPs.
- We pick a notable result on the complexity of the CSPs.
- Let F be any set of constraints used for CSPs.
- **Schaefer** (1978) proved a dichotomy theorem for
 - unweighted Boolean CSPs,
(that is, $F = \text{Boolean constraints}$)
- A **dichotomy theorem** says that every $\text{CSP}(F)$ is classified into only **two** categories:
 1. problems in P , and
 2. problems that are NP-complete.

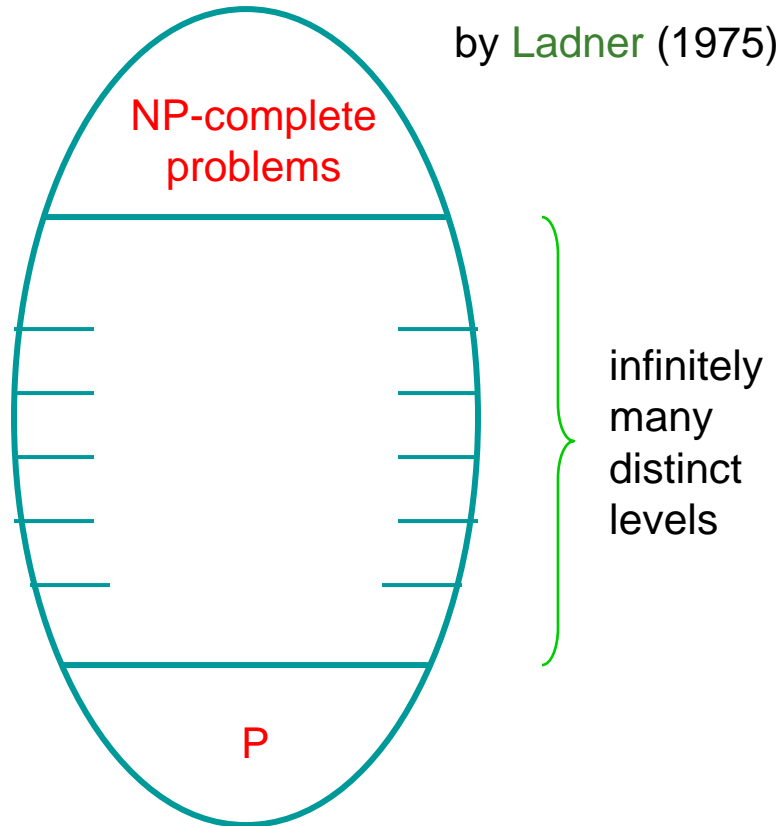
Classification Theorem of Schaefer II

- We formally state the result of Schaefer.
- **Dichotomy Theorem:** [Schaefer (1978)]
 - If F is in one of six sets of constraints described below, then $\text{CSP}(F)$ is in P .
 - Otherwise, $\text{CSP}(F)$ is NP-complete.
- Six sets of unweighted Boolean constraints:
 1. 0-valid
 2. 1-valid
 3. weakly positive
 4. weakly negative
 5. affine
 6. bijection

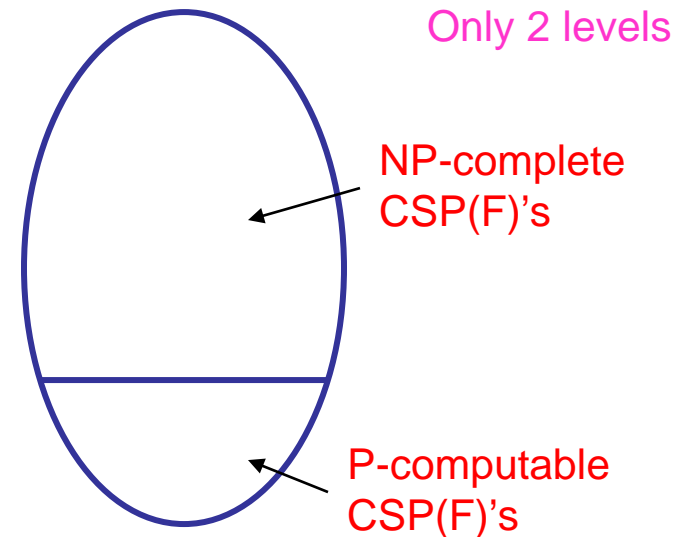
E.g., an **affine** relation is a collection of solutions of a certain set of linear equations over Galois field $\text{GF}(2)$.

Worlds of NP and CSP(F)'s (if $P \neq NP$)

World of NP-problems



World of CSP(F)'s



NP has infinitely many levels induced by \leq_m^P , assuming $P \neq NP$.

CSP(F) has only 2 levels induced by \leq_m^P , assuming $P \neq NP$.

Counting CSPs (or #CSPs)



- Next, we study a counting version of CSPs (called counting CSPs or #CSPs).
- Let F be any set of constraints.
- **Counting CSP: $\#CSP(F)$**
 - **Instance:**
 - a set of Boolean variables
 - a set of constraints in F
 - **Question:**
 - How many (variable) assignments satisfy all the given constraints?
- **NOTE:** all #CSPs (counting CSPs) are #P problems.

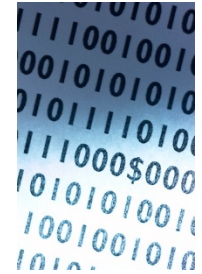
Here, we consider only the “Boolean” case.

Examples of #CSPs

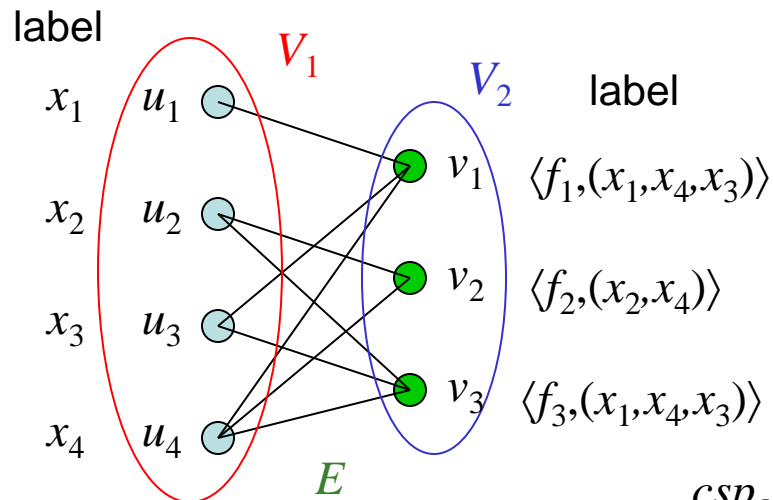
- Here are a few examples of #CSPs.
- **Examples:**
 - #SAT (counting satisfiability problem)
 - #BIS (counting bipartite independent set problem)
 - #DOWNSET (counting downset problems)



Visualizing #CSP(F) I



- A **constraint frame** Ω is a tuple $(G, X | F', \pi)$, where
 1. a undirected bipartite graph $G = (V_1/V_2, E)$,
 2. a variable set $X = \{x_1, x_2, \dots, x_n\}$,
 3. $F' \subseteq F$, a finite subset, and
 4. $\pi: V_1 \cup V_2 \rightarrow X \cup F'$ (a **labeling function**) s.t.
 $\pi(V_1) \subseteq X$ and $\pi(V_2) \subseteq F'$.



Boolean variables: $\{x_1, x_2, x_3, x_4\}$
 $\pi(u_1) = x_1, \pi(u_2) = x_2, \pi(u_3) = x_3,$
 $\pi(u_4) = x_4$

(Boolean) constraints:
 $\pi(v_1) = \langle f_1, (x_1, x_4, x_3) \rangle,$
 $\pi(v_2) = \langle f_2, (x_2, x_4) \rangle,$
 $\pi(v_3) = \langle f_3, (x_1, x_4, x_3) \rangle$

$$csp_{\Omega} = \sum_{x_1, x_2, x_3, x_4 \in \{0,1\}} f_1(x_1, x_4, x_3) \cdot f_2(x_2, x_4) \cdot f_3(x_3, x_2, x_4)$$

Visualizing #CSP(F) II



- Each counting problem #CSP(F) is defined as follows.
- #CSP(F)
 - instance: a constraint frame Ω
 - task: compute the value csp_Ω
- When we consider the computational complexity of #CSP(F), we expand FP and #P to handle arbitrary complex numbers.

How to Describe a Constraint

- A **constraint** f of **arity** k is a function from $\{0,1\}^k$ to C .
- We assume a lexicographic order on $\{0,1\}^k$.
- Let $f: \{0,1\}^2 \rightarrow C$ be any constraint of arity 2. This f is expressed as:

$$f = [f(00), f(01), f(10), f(11)]$$

- Let $f: \{0,1\}^3 \rightarrow C$ be any constraint of arity 3.

$$f = [f(000), f(001), f(010), f(011), f(100), f(101), f(110), f(111)]$$

- **Examples:**

1. $OR_2 = [0, 1, 1, 1]$, $OR_3 = [0, 1, 1, 1, 1, 1, 1, 1]$

2. $Implies = [1, 1, 0, 1]$

3. $EQ_2 = [1, 0, 0, 1]$, $EQ_3 = [1, 0, 0, 0, 0, 0, 0, 1]$

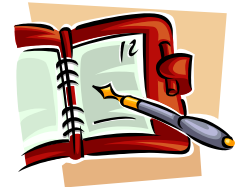
4. $NAND_2 = [1, 1, 1, 0]$, $XOR_2 = [0, 1, 1, 0]$

5. $\Delta_0 = [1, 0]$, $\Delta_1 = [0, 1]$



Simple Results on Unweighted Constraints

- An **unweighted (Boolean) constraint** is a constraint mapping $\{0,1\}^k$ to $\{0,1\}$ for a certain constant $k \in \mathbb{N}^+$.
- CSPs with unweighted constraints have been studied for a long time.
- **(Claim)** Here are simple known results about unweighted (Boolean) constraints:
 - $\#CSP(EQ_2) \in FP$
 - $\#CSP(AND_2) \in FP$
- **(Open Problem)** Is $\#CSP(OR_2) \in FP$?



Dichotomy Theorems (Exact Counting) I

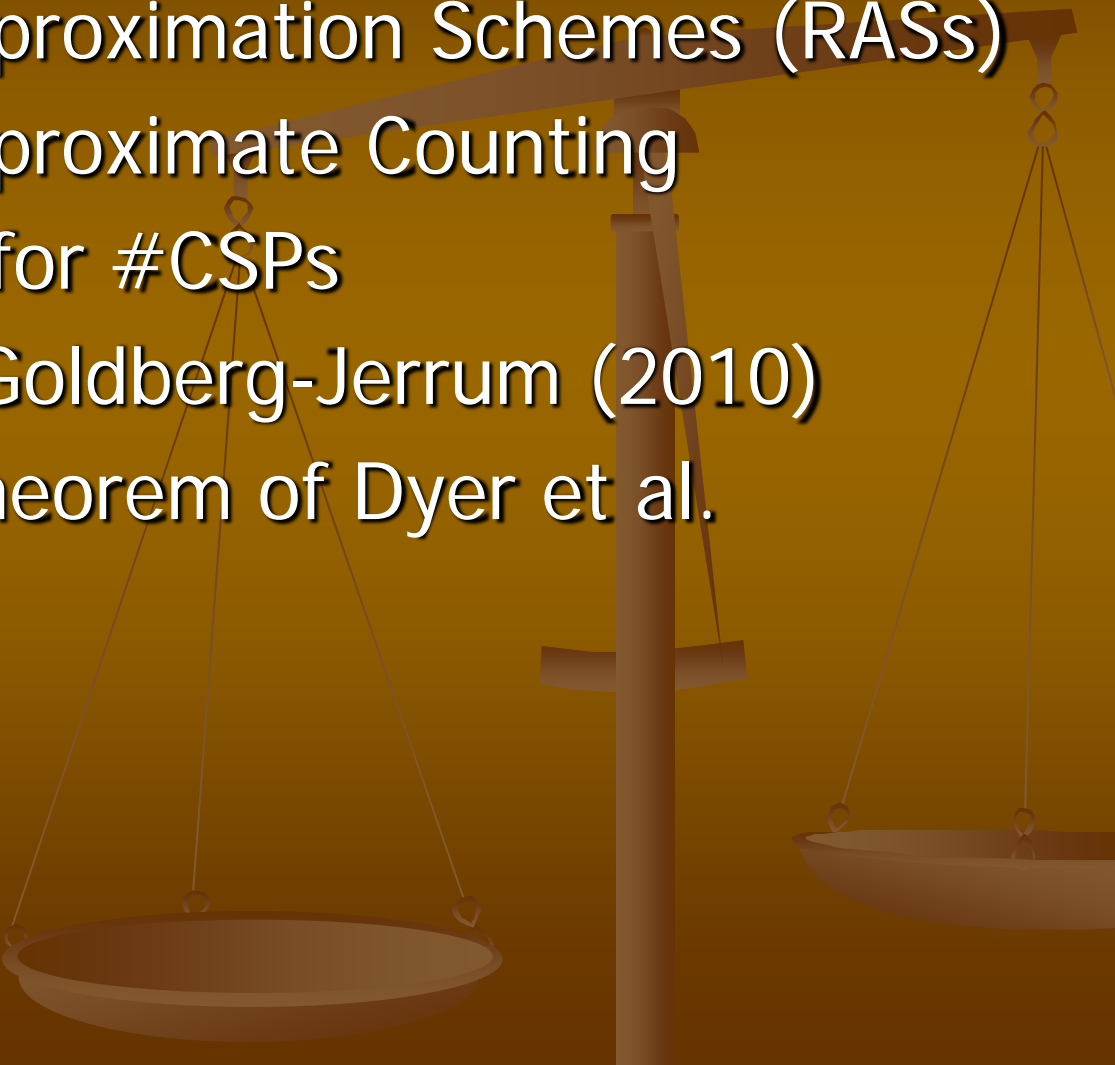
- Exact Counting: $\#CSP(F)$
- We review some known results of $CSP(F)$'s.
 - If F consists of affine constraints, then $\#CSP(F)$ is in FP. Otherwise, $\#CSP(F)$ is $\#P$ -complete.
- Creignou and Herman (1996) proved a dichotomy theorem about
 - unweighted Boolean $\#CSP$ s.
- Dyer, Goldberg, and Jerrum (2009) proved a dichotomy theorem about
 - non-negative-weighted Boolean $\#CSP$ s.
- Cai, Lu, and Xia (2009) proved a dichotomy theorem about
 - complex-weighted Boolean $\#CSP$ s.

Classification Theorems (Exact Counting) II

- We quickly review the result of [Cai, Lu, and Xia \(2009\)](#).
- Let F be any complex-weighted constraint set.
- **(Claim)** [Cai-Lu-Xia (2009)]
 1. If all constraints in F are affine, then $\#\text{CSP}(F) \in \text{FP}_C$.
 2. Otherwise, $\#\text{CSP}(F)$ is $\#\text{P}$ -complete under polynomial-time Turing reductions.

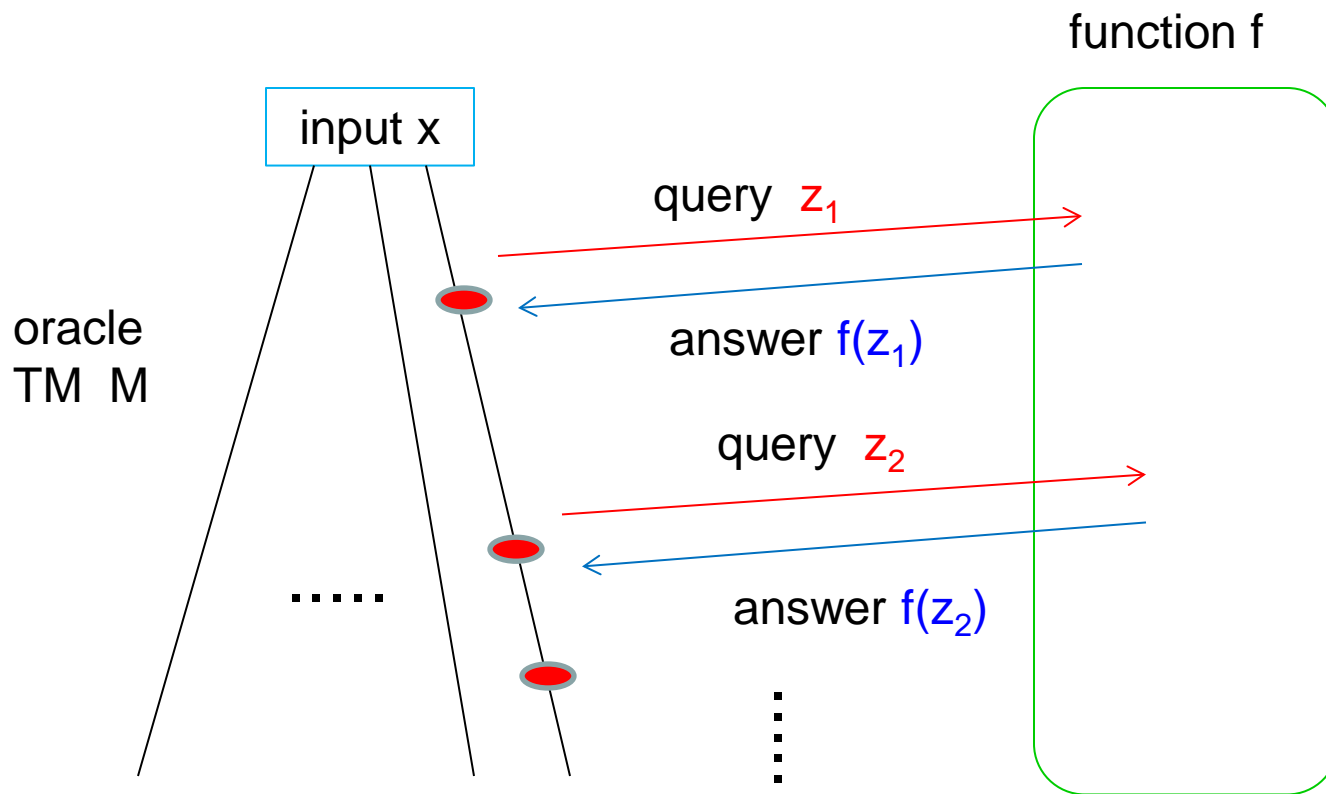


III. Approximate Counting of CSPs

1. Randomized Approximation Schemes (RASs)
 2. Randomized Approximate Counting
 3. AP-Reducibility for #CSPs
 4. Result of Dyer-Goldberg-Jerrum (2010)
 5. Classification Theorem of Dyer et al.
- 

Function-Oracle PTMs (revisited)

- Recall **function-oracle Turing machines** from Week 6.



Randomized Approximation Schemes (RASs)

- We are focused on the **approximate counting** of CSPs instead of the exact counting of CSPs.
- We explain what type of approximation we are using.
- Let F be a function from $\Sigma^* \rightarrow \mathbb{C}$.
- A **randomized approximation scheme** is a probabilistic algorithm that
 - takes $(x, \varepsilon) \in \Sigma^* \times \mathbb{R}^{\geq 0}$ as an input.
 - outputs a number w such that
 - $F(x)$ are **approximated** by w with **relative error** of 2^ε with high probability.
- A **fully polynomial-time randomized approximation scheme** (or **FPRAS**) is a RAS that runs in time polynomial in $(|x|, 1/\varepsilon)$.

NOTE: in this model, even if α and β are approximated, $\alpha + \beta$ may not be approximated properly for complex numbers α, β .

$$2^{-\varepsilon} \leq \left| \frac{w}{F(x)} \right| \leq 2^\varepsilon \quad \text{and} \quad \left| \arg \left(\frac{w}{F(x)} \right) \right| \leq \varepsilon$$

Randomized Approximate Counting

- It is practical to approximate $\#CSPs$ rather than compute $\#CSPs$ exactly.
- Randomized Approximate Counting: $\#CSP(F)$
 - Let $\#CSP^*(F) = \#CSP(F, U)$, where U is the set of all unary constraints.
- Here, we overview a result of [Dyer, Goldberg, and Jerrum \(2010\)](#) regarding the approximation complexity of $\#CSP(F)$'s.

“*” means that we use any unary constraint for free of charge.

AP-Reducibility for #CSPs



- Dyer, Goldberg, Greenhill, and Jerrum (2003) introduced a notion of (**randomized**) **approximation-preserving reduction** (or **AP-reduction**).
- Let F and G be any two counting problems.

- F is **AP-reducible** to G by an AP-reduction $M \iff$
 - M is a function-oracle PTM working on input (x, ϵ) with an oracle,
 - M is a RAS for F and the oracle is also a RAS for G ,
 - every oracle call made by M is of the form (w, δ) with $1/\delta \leq \text{poly}(|x|, 1/\epsilon)$,
 - the running time of M is bounded by a polynomial in $(|x|, 1/\epsilon)$.

Notational Convention

- **Notation:**

1. $F \leq_{AP} G \Leftrightarrow F$ is AP-reducible to G .

2. $F \equiv_{AP} G \Leftrightarrow F \leq_{AP} G$ and $G \leq_{AP} F$.

- We list some simple known results about unweighted (Boolean) constraints:

- $\#CSP(OR_2) \equiv_{AP} \#CSP(NAND_2) \equiv_{AP} \#SAT$

- $\#CSP(Implies) \equiv_{AP} \#BIS$

- **(Claim)** [Dyer-Goldberg-Jerrum (2003)]

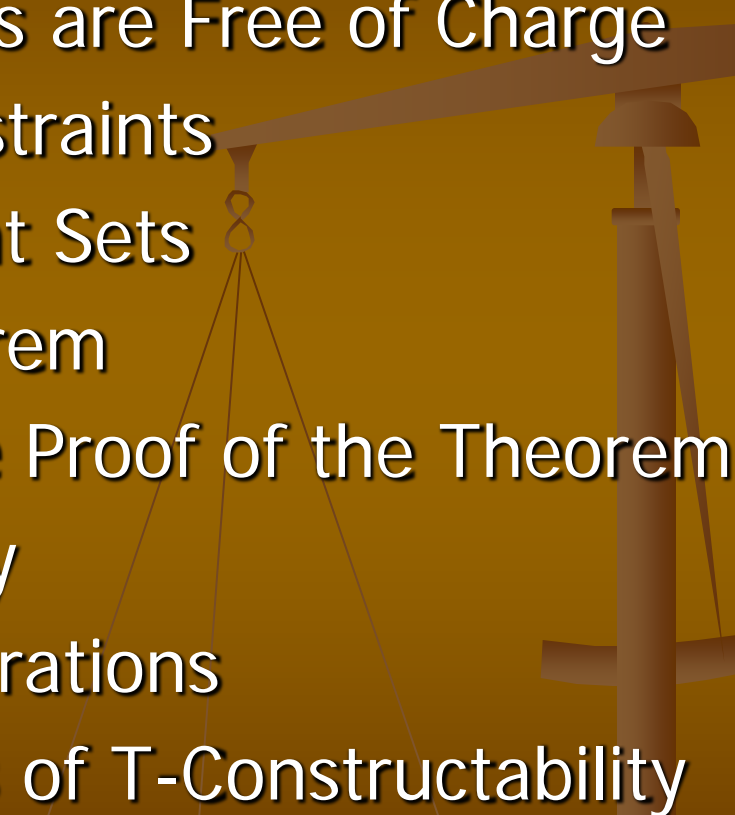
$\#DOWNSET \equiv_{AP} \#BIS \leq_{AP} \#SAT$



Result of Dyer-Goldberg-Jerrum (2010)

- Before describing our result, we quickly go over the aforementioned result of [Dyer, Goldberg, and Jerrum \(2010\)](#).
- They proved the next theorem on **unweighted** constraints.
- Let F be any set of unweighted constraints.
- **Trichotomy Theorem:** [Dyer-Goldberg-Jerrum (2010)].
 - If F consists of affine constraints, then $\#\text{CSP}(F) \in \text{FP}$.
 - Otherwise, if $F \subseteq \text{IM-conj}$, then $\#\text{CSP}(F) \equiv_{\text{AP}} \#\text{BIS}$.
 - Otherwise, $\#\text{CSP}(F) \equiv_{\text{AP}} \#\text{SAT}$.
- **IM-conj** = set of Boolean constraints logically equivalent to products of **Implies**, Δ_0 , and Δ_1 , where
 - $\Delta_0(x) = \text{False}$, $\Delta_1(x) = \text{True}$, and
 - “**Implies**” means a logical connective “ \rightarrow ” (implies); i.e., $\text{Implies}(x,y) = \text{OR}_2(\text{NOT}(x), y)$

IV. Classification Theorem

1. Unary Constraints are Free of Charge
 2. Degenerate Constraints
 3. Special Constraint Sets
 4. Dichotomy Theorem
 5. An Outline of the Proof of the Theorem
 6. T-Constructability
 7. Examples of Operations
 8. Useful Properties of T-Constructability
- 

Unary Constraints are Free of Charge

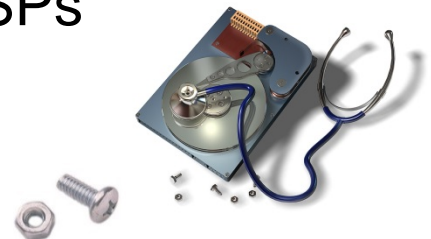
- Here, any **unary constraint** is used for free of charge.
- Let **U** be the set of all unary constraints.
- Constant unary constraints are Δ_0 , and Δ_1 .
- Such a use of free unary constraints has been made elsewhere.

- Feder (2001) for Boolean CSPs
- Dalmau and Ford (2003) for Boolean CSPs
- Cai, Huang, and Lu (2010) for Holant problems
- Cai, Lu, and Xia (2009) for Holant problems
- Dyer, Goldberg, Jalsenius, and Richerby (2010) for bounded-degree #CSPs
- Yamakami (2010) for bounded-degree #CSPs

- Notational convention:

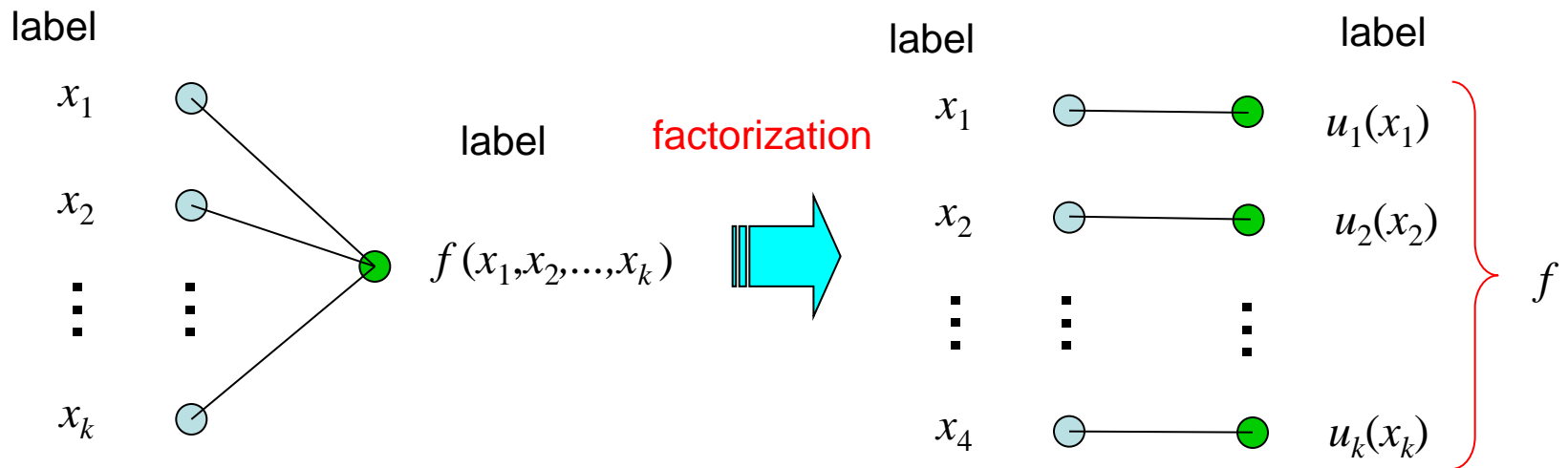
$$\blacktriangleright \#CSP^*(F) =_{\text{def}} \#CSP(F, U)$$

$$\begin{aligned}\Delta_0(x) &= \text{False}(x) \\ \Delta_1(x) &= \text{True}(x)\end{aligned}$$



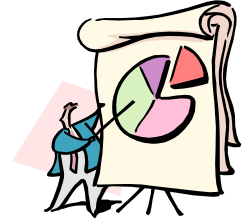
Degenerate Constraints

- **Degenerate constraints** are ones that are expressed as products of unary constraints.
- To be more precise, let us consider a constraint f of arity k that can be factorized into k unary constraints u_1, u_2, \dots, u_k .



- In this case, f is called **degenerate**.
- csp_{Ω} is simply calculated as:
$$csp_{\Omega} = \prod_{i=1}^k (u_i(0) + u_i(1))$$

Special Constraint Sets



- Let each h_i denote a unary constraint.
- **NZ** = set of constraints f of arity ≥ 1 such that $f(x_1, x_2, \dots, x_k) \neq 0$ (non-zero) for all $(x_1, x_2, \dots, x_k) \in \{0, 1\}^k$
- **DG** = set of all degenerate constraints
- **ED** = set of constraints f of arity ≥ 1 such that

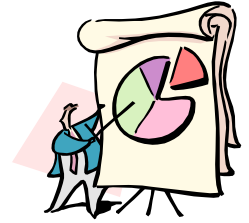
$$f(x_1, x_2, \dots, x_k) = \left(\prod_{i=1}^{\ell_1} h_i(x_{j_i}) \right) \left(\prod_{i=1}^{\ell_2} g_i(x_{m_i}, x_{n_i}) \right)$$

with $\ell_1, \ell_2 \geq 0$, $\ell_1 + \ell_2 \geq 1$, and $1 \leq j_i, m_i, n_i \leq k$, where each g_i is either binary EQ_2 or XOR_2 .

- **IM** = set of constraints $f \notin \text{NZ}$ of arity ≥ 1 such that

$$f(x_1, x_2, \dots, x_k) = \left(\prod_{i=1}^{\ell_1} h_i(x_{j_i}) \right) \left(\prod_{i=1}^{\ell_2} \text{Implies}(x_{m_i}, x_{n_i}) \right)$$

Dichotomy Theorem



- Let $\#SAT_C$ be a complex-weighted version of the counting satisfiability problem.
- Yamakami's (2012) dichotomy theorem says:
- **Dichotomy Theorem:** [Yamakami (2012)]
 1. If $F \subseteq ED$, then $\#CSP^*(F) \in FP_C$.
 2. Otherwise, $\#SAT_C \leq_{AP} \#CSP^*(F)$.
- A key is the following proposition on signatures f .
- **Proposition:** [Yamakami (2012)]

Assume that $f \notin AF \cup ED$. Let F be any signature set.

 1. $\#CSP^*(\text{Implies}, F) \equiv_{AP} \#CSP^*(OR_2, F)$.
 2. If $f \notin ED \cup NZ$, then $\#CSP^*(OR_2, F) \leq_{AP} \#CSP^*(f, F)$.

An Outline of the Proof of the Proposition

- Let us give an outline of the proof of **Item (2)** of the previous proposition.
- The proof proceeds **by induction** on the arity of f .
 1. Basis case: $k = 1$. ← Trivial from the definitions.
 2. Next basis case: $k = 2$. ← By constructing AP-reductions.
 3. Induction case: $k \geq 3$. ← A core of the proof.
- Since **Basis Case** ($k=1$) is trivial, we are focused on **Next Basis Case**.
- In the next slide, we will present a bit of flavor of the proof.

Next Basis Case: $k = 2$



- The next basis case is handled case by case.

CASES

WANTS TO PROVE

- Let $f = [a, 0, 0, b]$ with $ab \neq 0$.

$$\#CSP^*(EQ_2, F) \leq_{AP} \#CSP^*(f, F)$$

- Let $f = [0, a, b, 0]$ with $ab \neq 0$.

$$\#CSP^*(XOR, F) \leq_{AP} \#CSP^*(f, F)$$

- Let $f = [a, b, c, 0]$ with $abc \neq 0$.

$$\#CSP^*(OR_2, F) \leq_{AP} \#CSP^*(f, F)$$

- Let $f = [1, a, b, c]$
with $abc \neq 0$ and $ab \neq c$.

$$\#CSP^*(OR_2, F) \leq_{AP} \#CSP^*(f, F)$$

- Let $f = [a, b, 0, c]$ with $abc \neq 0$.

We will prove this claim in the subsequent slides.

T-Constructability I

- For the case of $k \geq 3$, we need to introduce a useful notion of “T-constructability.”
 - A signature f is **T-constructable** from a set G of signatures if f is obtained from G by recursively applying the following operations.
 1. permutation
 2. pinning
 3. projection
 4. linking
 5. multiplication
 6. expansion
 7. normalization
- } See the subsequent slides
- We write $f \leq_{con} G$ if f is T-constructable from G .
 - If $G = \{g\}$, we simply write $f \leq_{con} g$.

T-Constructability II

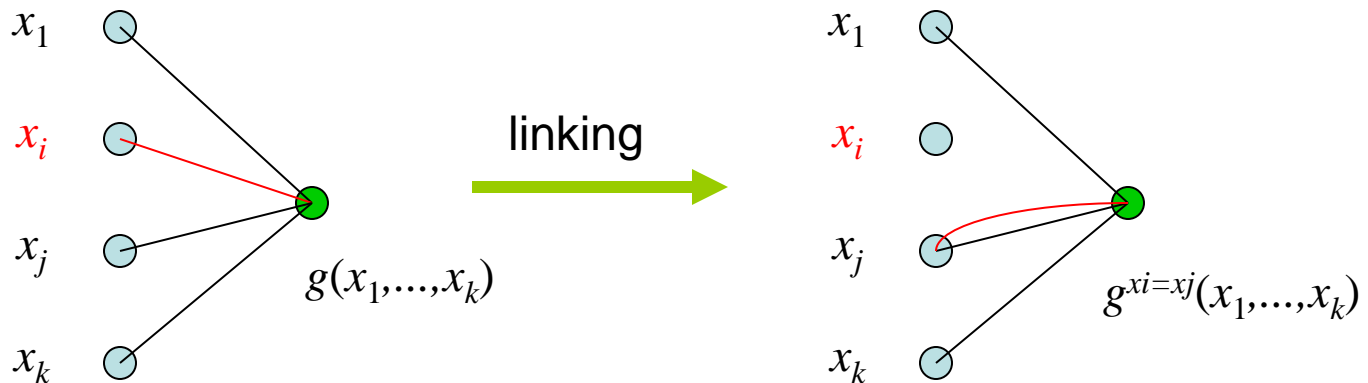


- We explain the operation “linking.”
- **linking**

$g \mapsto g^{x_i=x_j}$, where

$$g^{x_i=x_j}(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_k) = g(x_1, \dots, x_{i-1}, x_j, x_{i+1}, \dots, x_k)$$

- That is, “linking” is a replacement of variable x_i by x_j .



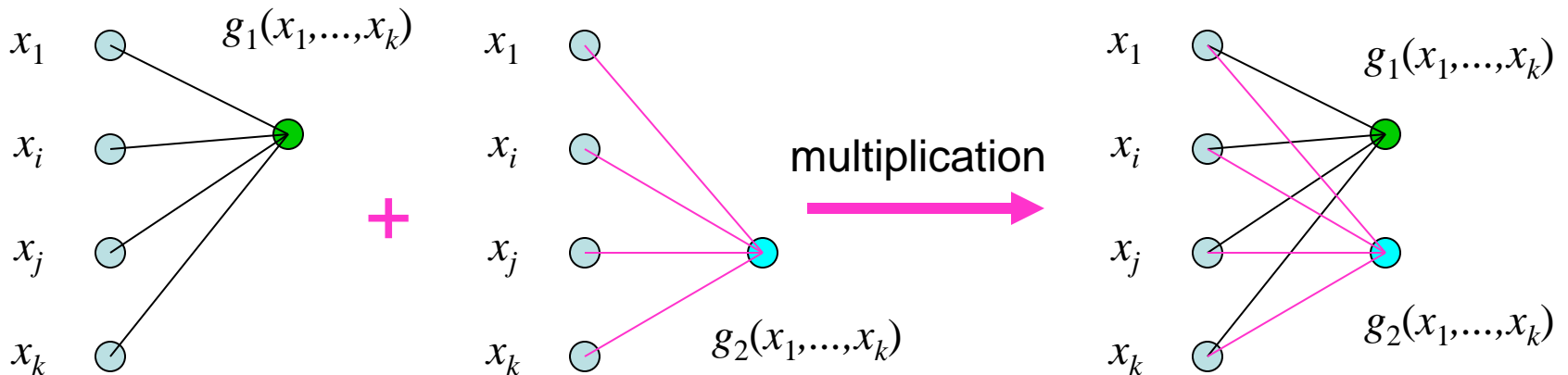
T-Constructability III

- We explain another operation “multiplication.”
- **multiplication**

$g \mapsto g_1 \bullet g_2$, where

$$g_1 \bullet g_2(x_1, \dots, x_k) = g_1(x_1, \dots, x_k) g_2(x_1, \dots, x_k)$$

- “Multiplication” is a multiplication of two signatures.



Useful Properties of T-Constructability

- We list a few useful properties of T-constructability.
- **(Claim)** \leq_{con} is a partial order; that is,
 - (reflexivity) $g \leq_{\text{con}} g$.
 - (transitivity) $g \leq_{\text{con}} h$ and $h \leq_{\text{con}} k$ imply $g \leq_{\text{con}} k$.
- **(Claim)** T-constructability is invariant under AP-reductions; that is,

$$g \leq_{\text{con}} f \Rightarrow \forall F \left[\#CSP^*(g, F) \leq_{AP} \#CSP^*(f, F) \right]$$

- This last claim helps us prove that

$$\#CSP^*(\text{Implies}, F) \leq_{AP} \#CSP^*(f, F)$$



Case of $f = [1, a, 0, b]$ with $ab \neq 0$.

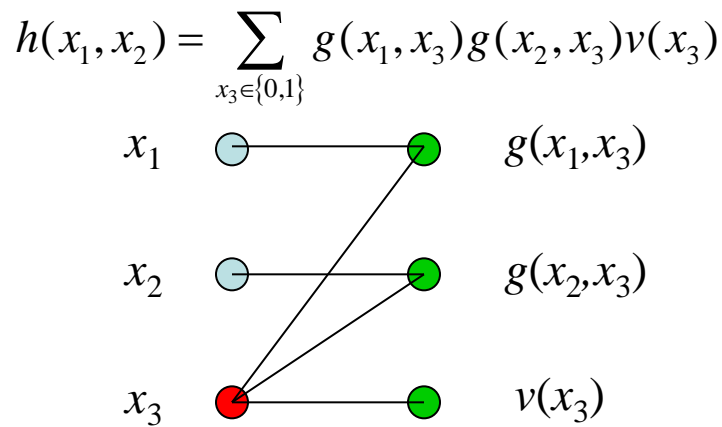
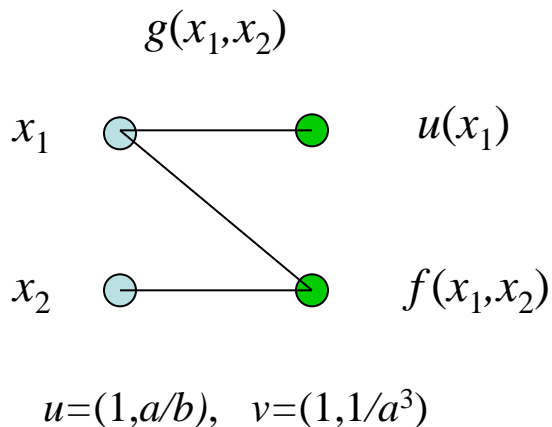


- In the case of $f = [a, b, 0, c]$ with $abc \neq 0$, we may set $a=1$.
- Let $f = [1, a, 0, b]$ with $ab \neq 0$. Here, we want to show that

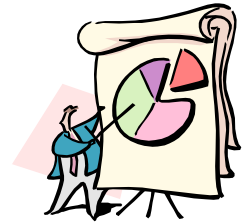
$$\forall F \left[\#CSP^*(\text{Implies}, F) \leq_{AP} \#CSP^*(f, F) \right]$$

- To prove this, it suffices to show that $\text{Implies} \leq_{\text{con}} \{f, u, v\}$ for some unary signatures $u, v \in \mathbf{U} \cap \mathbf{NZ}$.

□ **Proof Sketch:** We express “Implies” by the following h .



Dichotomy Theorem (again)



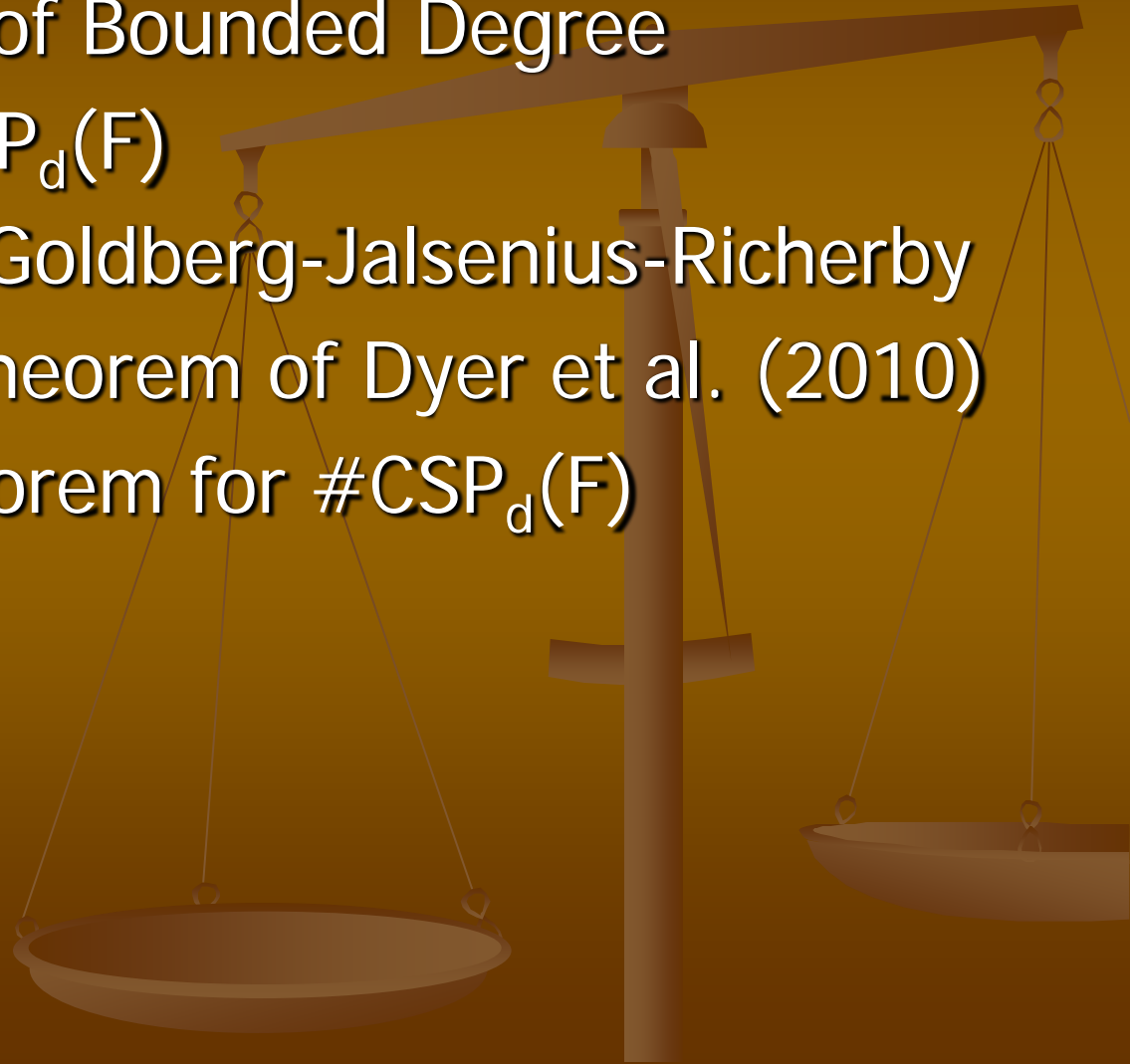
- Let $\#SAT_C$ be a complex-weighted version of the counting satisfiability problem.
- Yamakami's (2012) dichotomy theorem says:
- **Dichotomy Theorem:** [Yamakami (2012)]
 1. If $F \subseteq ED$, then $\#CSP^*(F) \in FP_C$.
 2. Otherwise, $\#SAT_C \leq_{AP} \#CSP^*(F)$.
- A key is the following proposition on signatures f .
- **Proposition:** [Yamakami (2010)]

Assume that $f \notin AF \cup ED$. Let F be any signature set.

 1. $\#CSP^*(\text{Implies}, F) \equiv_{AP} \#CSP^*(OR_2, F)$.
 2. If $f \notin ED \cup NZ$, then $\#CSP^*(OR_2, F) \leq_{AP} \#CSP^*(f, F)$.

V. Counting CSPs of Bounded Degree


1. Counting CSPs of Bounded Degree
2. Visualizing $\#CSP_d(F)$
3. Result of Dyer-Goldberg-Jalsenius-Richerby
4. Classification Theorem of Dyer et al. (2010)
5. Dichotomy Theorem for $\#CSP_d(F)$



Counting CSPs (or #CSPs) of Bounded Degree

- We study #CSP whose constraints have bounded degrees.
- Let F be any set of constraints and d b
- Degree-d Counting CSP: $\#CSP_d(F)$
 - Instance:
 - a set of Boolean variables
 - a set of constraints in F satisfying the degree-d condition
 - Question:
 - How many (variable) assignments satisfy all the given constraints?
- We write $\#CSP_d^*(F)$ for $\#CSP_d(F, U)$, where U is the set of all unary constraints.

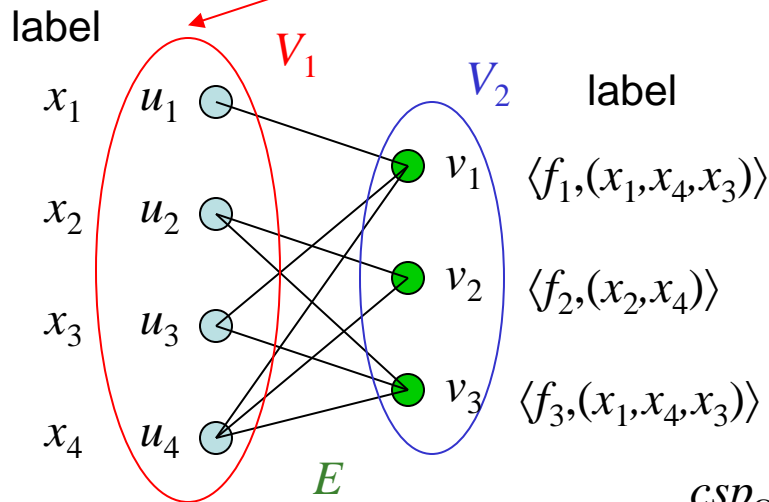
The **degree** is the maximal number of times that any variable appears among its constraints.



Visualizing $\#CSP_d(F)$

- $\#CSP_d(F)$
 - Instance Ω with the degree-d condition
 - Task: compute csp_Ω .

The degree-d condition means that the degree of each node in V_1 is at most d.



Boolean variables: $\{x_1, x_2, x_3, x_4\}$

$\pi(u_1) = x_1, \pi(u_2) = x_2, \pi(u_3) = x_3,$
 $\pi(u_4) = x_4$

(Boolean) constraints:

$\pi(v_1) = \langle f_1, (x_1, x_4, x_3) \rangle,$
 $\pi(v_2) = \langle f_2, (x_2, x_4) \rangle,$
 $\pi(v_3) = \langle f_3, (x_1, x_4, x_3) \rangle$

$$csp_\Omega = \sum_{x_1, x_2, x_3, x_4 \in \{0,1\}} f_1(x_1, x_4, x_3) \cdot f_2(x_2, x_4) \cdot f_3(x_3, x_2, x_4)$$

Approximating Bounded-Degree #CSPs

- It is practical to approximate #CSPs rather than compute #CSPs exactly.
- Here, we overview a known result about the approximation complexity of $\#CSP_d(F)$'s degree d .
 - **Approximate Counting: $\#CSP_d^*(F)$**
 - **Dyer, Goldberg, Jalsenius, and Richerby (2010)** proved a classification theorem on
 - **unweighted** Boolean #CSPs of **degree ≥ 3** .
 - **Yamakami (2010)** proved a classification theorem on
 - **complex-weighted** Boolean #CSPs of **degree ≥ 3** .

“*” means that we use any **unary** constraint for free of charge.

Result of Dyer-Goldberg-Jalsenius-Richerby I

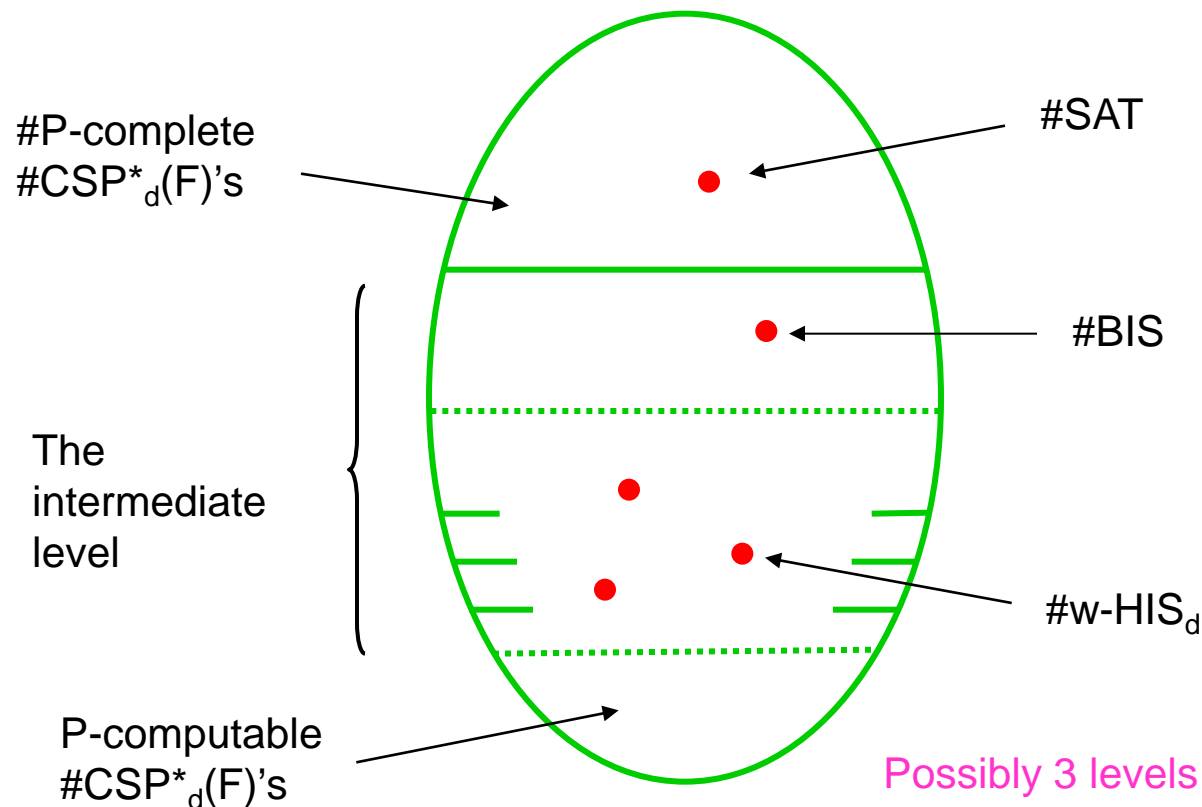
- Dyer, Goldberg, Jalsenius, and Richerby (2010) proved the following classification theorem on **unweighted Boolean constraints**.
- To describe their result, we need the following definitions.
 - **IM-conj** = set of Boolean constraints logically equivalent to products of **Implies**, Δ_0 , and Δ_1
 - **OR-conj**, **NAND-conj** = similarly defined
 - **#BIS** = problem of counting the independent sets in a bipartite graph
 - **#w-HIS_d** = problem of counting the independent sets in a width-w **hypergraph** H of degree at most d

Result of Dyer-Goldberg-Jalsenius-Richerby II

- Let F be a set of unweighted Boolean constraints and set $d \geq 3$.
- **Classification Theorem:** [Dyer-Goldberg-Jalsenius-Richerby (2010)]
 1. If F consists of affine constraints, then $\#\text{CSP}_d^*(F)$ is in FP.
 2. Otherwise, if $F \subseteq \text{IM-conj}$, then $\#\text{CSP}_d^*(F) \equiv_{\text{AP}} \#\text{BIS}$.
 3. Otherwise, if $F \subseteq \text{OR-conj} \cup \text{NAND-conj}$, then $\#\text{w-HIS}_d \leq_{\text{AP}} \#\text{CSP}_d^*(F) \leq_{\text{AP}} \#\text{w-HIS}_{kd}$ (where k is a constant).
 4. Otherwise, $\#\text{CSP}_d^*(F) \equiv_{\text{AP}} \#\text{SAT}$.

Classification Theorem of Dyer et al. (2010)

- Dyer, Goldberg, Jalsenius, and Richerby (2010) proved the following classification theorem on **unweighted** $\#CSP^*_d(F)$'s.



Dichotomy Theorems for $\#\text{CSP}_d(F)$

- The dichotomy theorem of **Yamakami** (2012) says:

- **Dichotomy Theorem:** Let $d \geq 3$.

1. If $F \subseteq \text{ED}$, then $\#\text{CSP}_d^*(F)$ is in FP_C .

2. Otherwise, $\#\text{SAT}_C^* \leq_{\text{AP}} \#\text{CSP}_d^*(F)$

- There has been an open problem of whether this theorem holds for **degree-2** Boolean

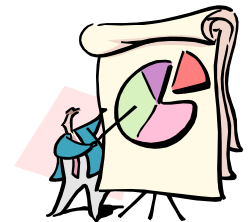
Holant*(F) uses **any graph**, not necessarily limited to bipartite graphs. [Cai-Lu-Xia (2009)]

- when $d = 1$ or 2 , we have the following result.

- **Theorem:** [Yamakami (2012)]

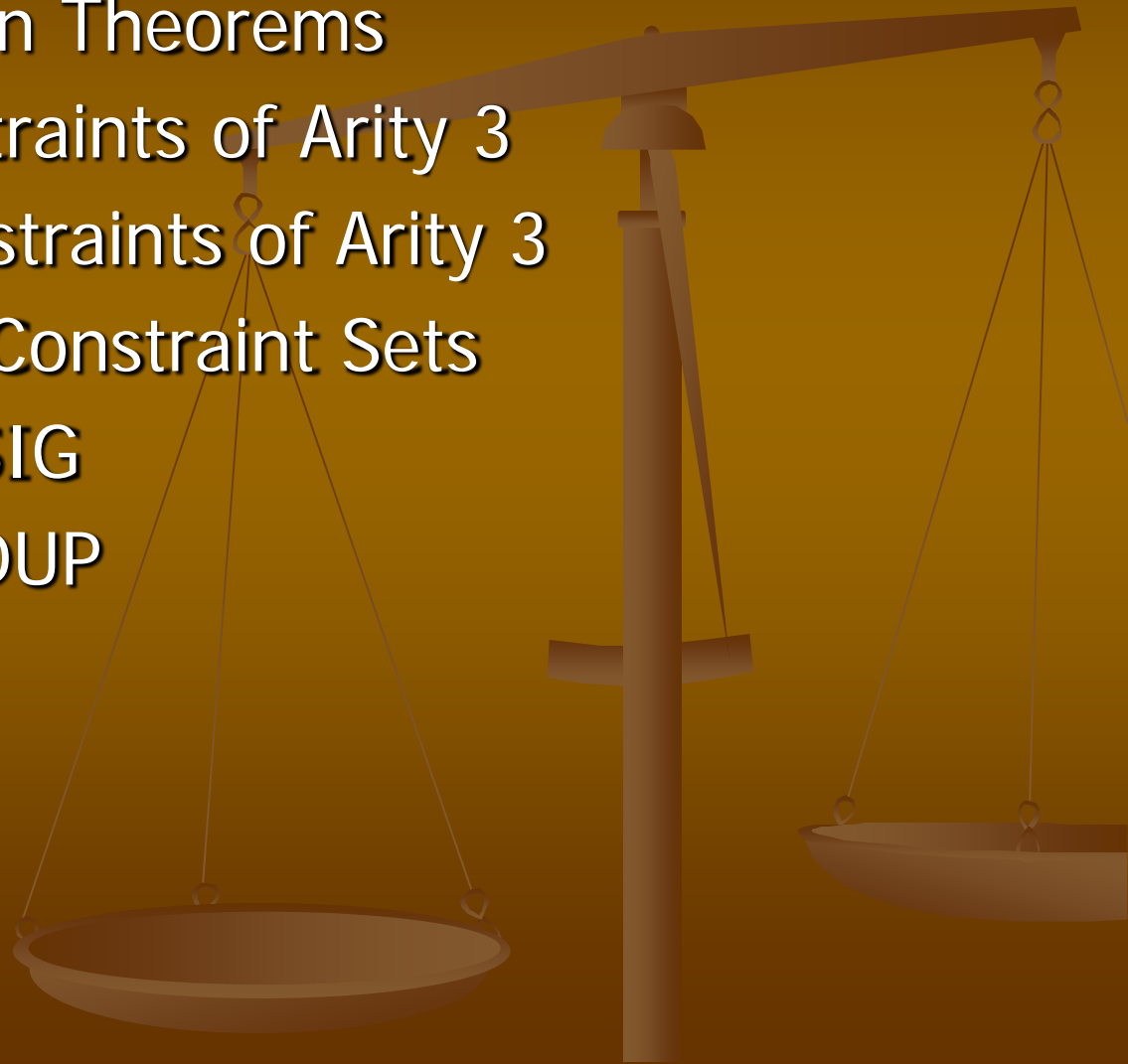
1. $\#\text{CSP}_1^*(F)$ is in FP_C .

2. $\#\text{CSP}_2^*(F) \equiv_{\text{AP}} \text{Holant}^*(F)$.



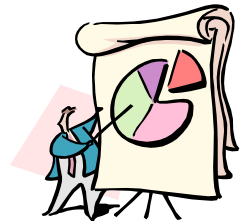
VI. Counting CSPs of Degree 2

1. Two Classification Theorems
2. Symmetric Constraints of Arity 3
3. Asymmetric Constraints of Arity 3
4. Two Symmetric Constraint Sets
5. Constraint Set SIG
6. Constraint Set DUP



Two Classification Theorems

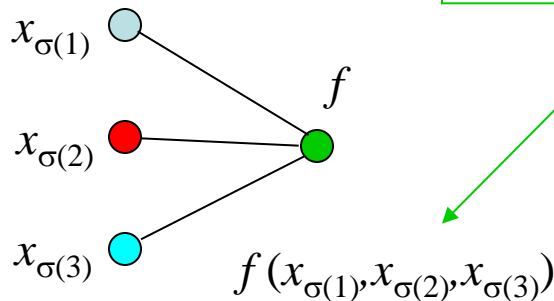
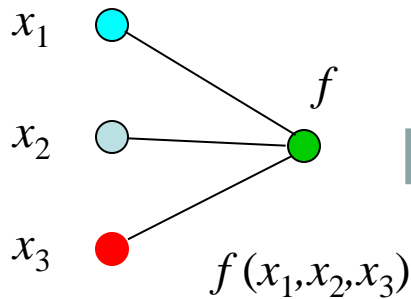
- Here are two classification theorems of Yamakami (2012) regarding ternary constraints.
- **Theorem:** [Yamakami (2012)]
Let f be any ternary constraint.
 - If $f \notin \text{SIG}$, then $\#\text{SAT} \leq_{\text{AP}} \#\text{CSP}^*_2(f)$.
- **Theorem:** [Yamakami (2012)]
Let f be any ternary constraint in SIG_1 .
 1. If $f \in \text{DUP}$, then $\#\text{CSP}^*_2(f) \in \text{FP}$.
 2. Otherwise, $\#\text{SAT} \leq_{\text{AP}} \#\text{CSP}^*_2(f)$.
- In the next slide, we will explain SIG and DUP.



Symmetric Constraints of Arity 3

- Let S_3 be the set of all permutations over $\{1,2,3\}$.
- A constraint is called **symmetric** if its output values depend only on the **Hamming weights of input bits**.
- In other words, the following equations hold.

$$\begin{cases} f(001) = f(010) = f(100) \\ f(011) = f(101) = f(110) \end{cases}$$



for any permutation $\sigma \in S_3$



Asymmetric Constraints of Arity 3

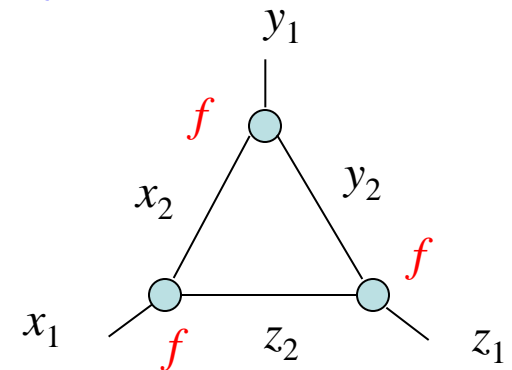
- Any constraint that is not symmetric is called **asymmetric**.

$$F = OR(x_1, x_2) \cdot OR(x_2, x_3) \cdot OR(x_3, x_1) \quad \leftarrow \text{symmetric}$$

$$F = OR(x_1, x_2) \cdot OR(x_2, x_3) \quad \leftarrow \text{asymmetric}$$

- There is a natural question of how we should deal with asymmetric constraints of arity 3.
- A useful method that we take is “**symmetrization**” of asymmetric constraints.
- Let f be any ternary constraint. We define **Sym(f)** as follows:

$$\begin{aligned} \text{Sym}(f)(x_1, y_1, z_1) \\ = \sum_{x_2, y_2, z_2 \in \{0,1\}} f(x_1, x_2, z_3) f(y_1, y_2, x_3) f(z_1, z_2, y_3) \end{aligned}$$



- (Claim)**

For any ternary constraint f , $\text{Sym}(f)$ is symmetric.

Two Symmetric Constraint Sets

- Cai, Lu, and Xia (2009) recognized two important sets of ternary **symmetric** constraints, which we tentatively call **Sig⁽¹⁾** and **Sig⁽²⁾**.
- For any **symmetric** constraint f of arity 3, we define:
 $f \in \text{Sig}^{(1)} \Leftrightarrow f(000)+f(011)=0$ and $f(001)+f(111)=0$
 $f \in \text{Sig}^{(2)} \Leftrightarrow \exists \alpha, \beta \in \mathbb{C}$ (not both zero) s.t.
 $\alpha f(000)+\beta f(001)-\alpha f(011)=0$ and $\alpha f(001)+\beta f(011)-\alpha f(111)=0$
- $\text{Sym}(f)$ behaves quite differently on **Sig⁽¹⁾** and **Sig⁽²⁾**.
- **Lemma:** [Yamakami (2012)]
 - If $f \in \text{Sig}^{(1)}$, then $\text{Sym}(f) \in \text{DG}$.
 - If $f \in \text{Sig}^{(2)}$, then $\text{Sym}(f) \in \text{Sig}^{(2)}$.

DG = set of degenerate constraints

Constraint Set SIG

- Recall that S_3 is the set of all permutations over $\{1,2,3\}$.
- Notation:** For a ternary constraint f and a permutation $\sigma \in S_3$, we define a permuted constraint f_σ as:

$$f_\sigma(x_1, x_2, x_3) := f(x_{\sigma(1)}, x_{\sigma(2)}, x_{\sigma(3)})$$

- Using this notation, we define SIG as follows.

$$SIG = \{f \mid \forall \sigma \in S_3 [Sym(f_\sigma) \notin DG \rightarrow Sym(f_\sigma) \in Sig^{(1)} \cup Sig^{(2)}]\}$$

$$SIG_0 = \{f \mid \forall \sigma \in S_3 [Sym(f_\sigma) \in DG]\}$$

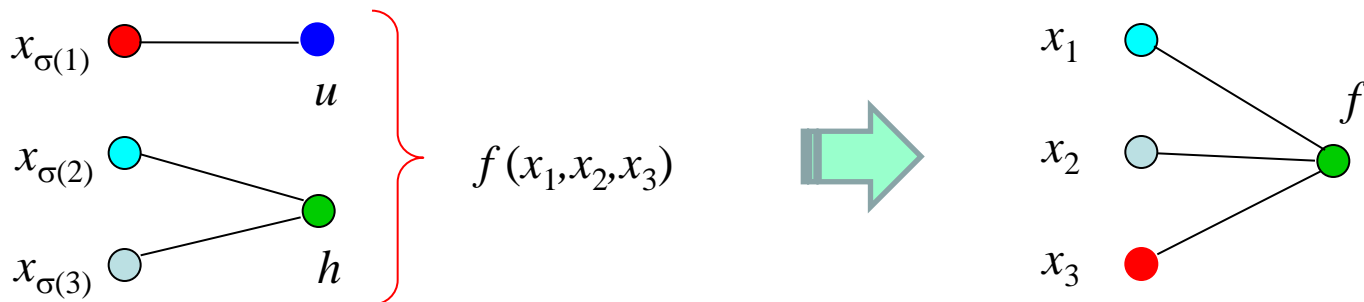
$$SIG_1 = \{f \mid \exists \sigma \in S_3 [Sym(f_\sigma) \notin DG \wedge Sym(f_\sigma) \in Sig^{(1)}]\}$$

$$SIG_2 = \{f \mid \exists \sigma \in S_3 [Sym(f_\sigma) \notin DG \wedge Sym(f_\sigma) \in Sig^{(2)}]\}$$



Constraint Set DUP

- We define DUP as a set of certain simple-structured constraints.
- Notation:** $f = (f_0, f_1) \Leftrightarrow \begin{cases} f(0, x_2, x_3) = f_0(x_2, x_3) \\ f(1, x_2, x_3) = f_1(x_2, x_3) \end{cases}$
- Let f be any ternary constraint.
- $f \in \text{DUP} \Leftrightarrow \exists \sigma$: permutation of variable indices, $\exists u$: unary constraint, and $\exists h$: binary constraint s.t. $f = u(x_{\sigma(1)}) \cdot (h, h)$.
- Graphically,** f can be expressed as follows:



Two Classification Theorems (again)

- Recall the two classification theorems of Yamakami (2012).
- **Theorem:** [Yamakami (2012)]
Let f be any ternary constraint.
 - If $f \notin \text{SIG}$, then $\#\text{SAT} \leq_{\text{AP}} \#\text{CSP}^*_2(f)$.
- **Theorem:** [Yamakami (2012)]
Let f be any ternary constraint in SIG_1 .
 1. If $f \in \text{DUP}$, then $\#\text{CSP}^*_2(f) \in \text{FP}$.
 2. Otherwise, $\#\text{SAT} \leq_{\text{AP}} \#\text{CSP}^*_2(f)$.
- **(Open Problem)** How about the case of $f \in \text{SIG}_2$?



VII. Elimination of Constant Unary Constraints

1. Elimination of Constant Unary Constraints
2. Key Proposition
3. Special Constraint Sets



Elimination of Constant Unary Constraints

- Hereafter, we consider only **real-weighted** constraints.
- Recall the two constant unary constraints:

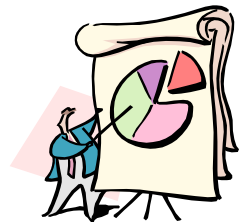
$$\Delta_0 = [1,0] \text{ and } \Delta_1 = [0,1].$$

- **Yamakami** (2014) proved the following theorem.
- **Theorem:** [Yamakami (2014)]

Let F be any nonempty set of real-valued constraints.

There exists a constant unary constraint $h \in \{ \Delta_0, \Delta_1 \}$ for which $\#CSP(h, F) \equiv_{AP} \#CSP(F)$.

- This means that either $h = \Delta_0$ or $h = \Delta_1$ can be completely eliminated from $F \cup \{ h \}$.

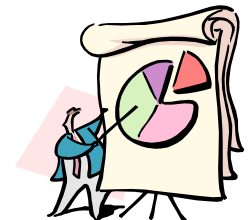


Key Proposition

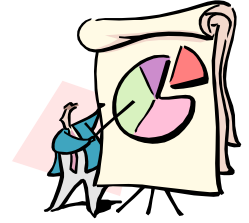
- A key is the following proposition on signatures f .
- **Proposition:** [Yamakami (2014)]

Let F be any nonempty set of real-valued constraints of arity ≥ 2 .

1. If either $F \subseteq DG \cup ED_1^{(+)}$ or $F \subseteq DG^{(-)} \cup ED_1 \cup AZ \cup AZ_1 \cup B_0$, then $\#CSP(F) \in FP_R$.
2. Otherwise, for any constraint set G , $\#CSP^*(g,G) \leq_{AP} \#CSP(F,G)$, where g is an appropriate constraint of one of the following 3 special forms.
 - a) $[0,y,z]$ with $y,z > 0$.
 - b) $[x,y,0]$ with $x,y > 0$.
 - c) $[x,y,z]$ with $x,y,z > 0$ and $xz \neq y^2$.



Special Constraint Sets I



- We introduce 6 special constraint sets.
- **DG** = set of constraints f that are expressed by products of unary functions, as shown before
- **ED₁** = set of constraints of the form: $[x, \pm x]$, $[x, 0, \dots, 0, \pm x]$ of arity ≥ 2 , and $[0, x, 0]$ with $x \neq 0$
- **ED₁⁽⁺⁾** = set of constraints of the form: $[x, y]$, $[x, 0, \dots, 0, y]$ of arity ≥ 2 , $[0, x, 0]$ with $x, y \neq 0$
- **AZ** = set of constraints of arity ≥ 3 of the form: $[0, x, 0, x, \dots, 0 \text{ or } x]$, $[x, 0, x, 0, \dots, x \text{ or } 0]$ with $x \neq 0$
- **AZ₁** = set of constraints of arity ≥ 3 of the form: $[0, x, 0, -x, 0, x, \dots, 0 \text{ or } x \text{ or } -x]$ and $[x, 0, -x, 0, x, 0, \dots, -x \text{ or } x \text{ or } 0]$ with $x \neq 0$

Special Constraint Sets II

- B_0 = set of constraints of the form $[z_0, z_1, \dots, z_k]$ with $k \geq 2$ and $z_0 \neq 0$ such that
 - 1) $z_{2i+1} = z_{2i+2} = (-1)^{i+1}z_0$ for all i satisfying $2i+1 \in [k]$ or $2i+2 \in [k]$, or
 - 2) $z_{2i} = z_{2i+1} = (-1)^i z_0$ for all i satisfying $2i \in [k]$ or $2i+1 \in [k]$
- For example, B_0 contains $[1, 1, -1]$ and $[1, -1, 1]$.



Key Proposition (again)

- Recall the key proposition regarding signatures f .
- **Theorem:** [Yamakami (2014)]

Let F be any nonempty set of real-valued constraints of arity ≥ 2 .

1. If either $F \subseteq DG \cup ED_1^{(+)}$ or $F \subseteq DG^{(-)} \cup ED_1 \cup AZ \cup AZ_1 \cup B_0$, then $\#CSP(F) \in FP_R$.
2. Otherwise, for any constraint set G , $\#CSP^*(g, G) \leq_{AP} \#CSP(F, G)$, where g is an appropriate constraint of one of the following 3 special forms.
 - a) $[0, y, z]$ with $y, z > 0$.
 - b) $[x, y, 0]$ with $x, y > 0$.
 - c) $[x, y, z]$ with $x, y, z > 0$ and $xz \neq y^2$.



Thank you for listening

Thank you for listening

Q & A

I'm happy to take your question!



END