16th Int. Conf. on Principles and Practice of Constraint Programming

St Andrews, Scotland,6-10th September 2010

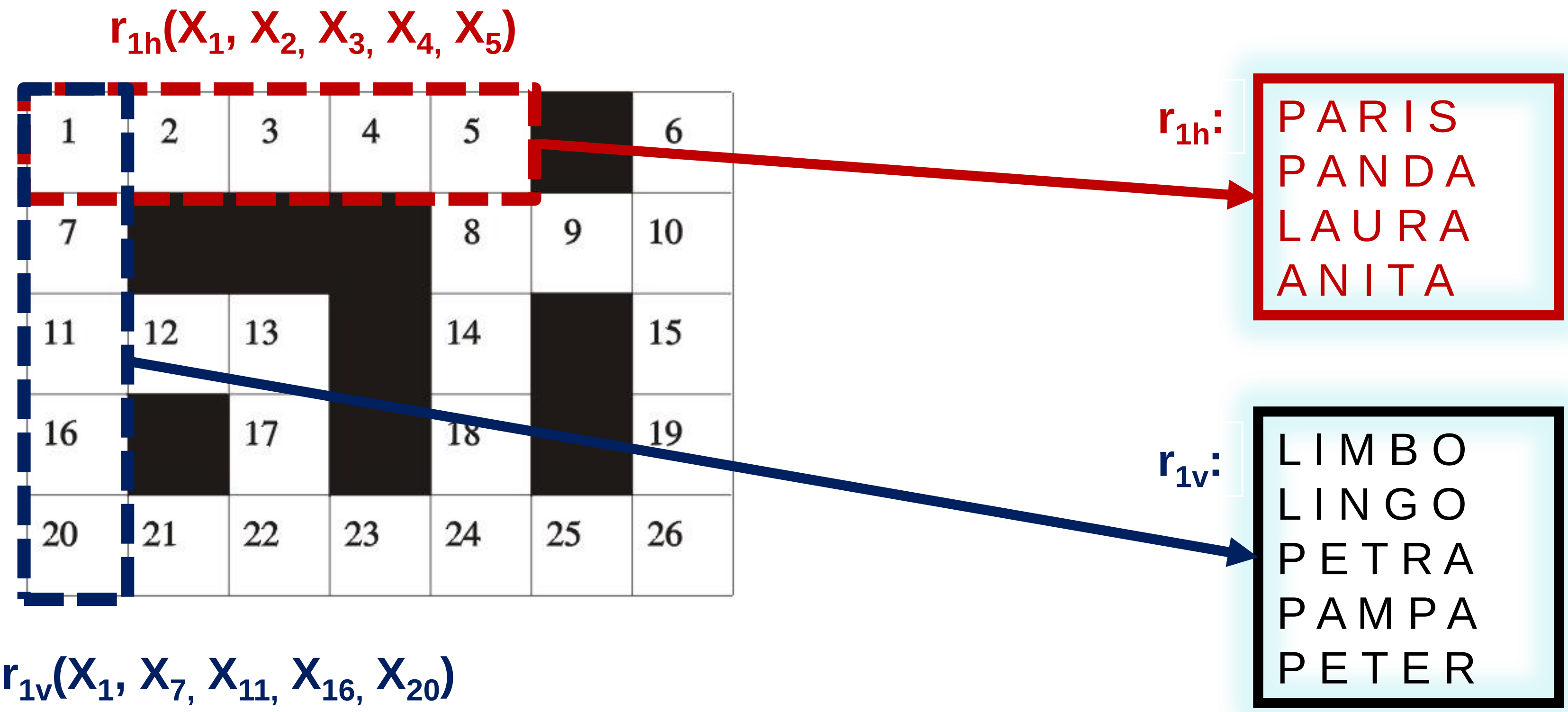
Structural Tractability of Enumerating CSP Solutions

UNIVERSITÀ DELLA CALABRIA
CAMPUS DI ARCAVACATA

Gianluigi Greco and **Francesco Scarcello**

University of Calabria, Italy

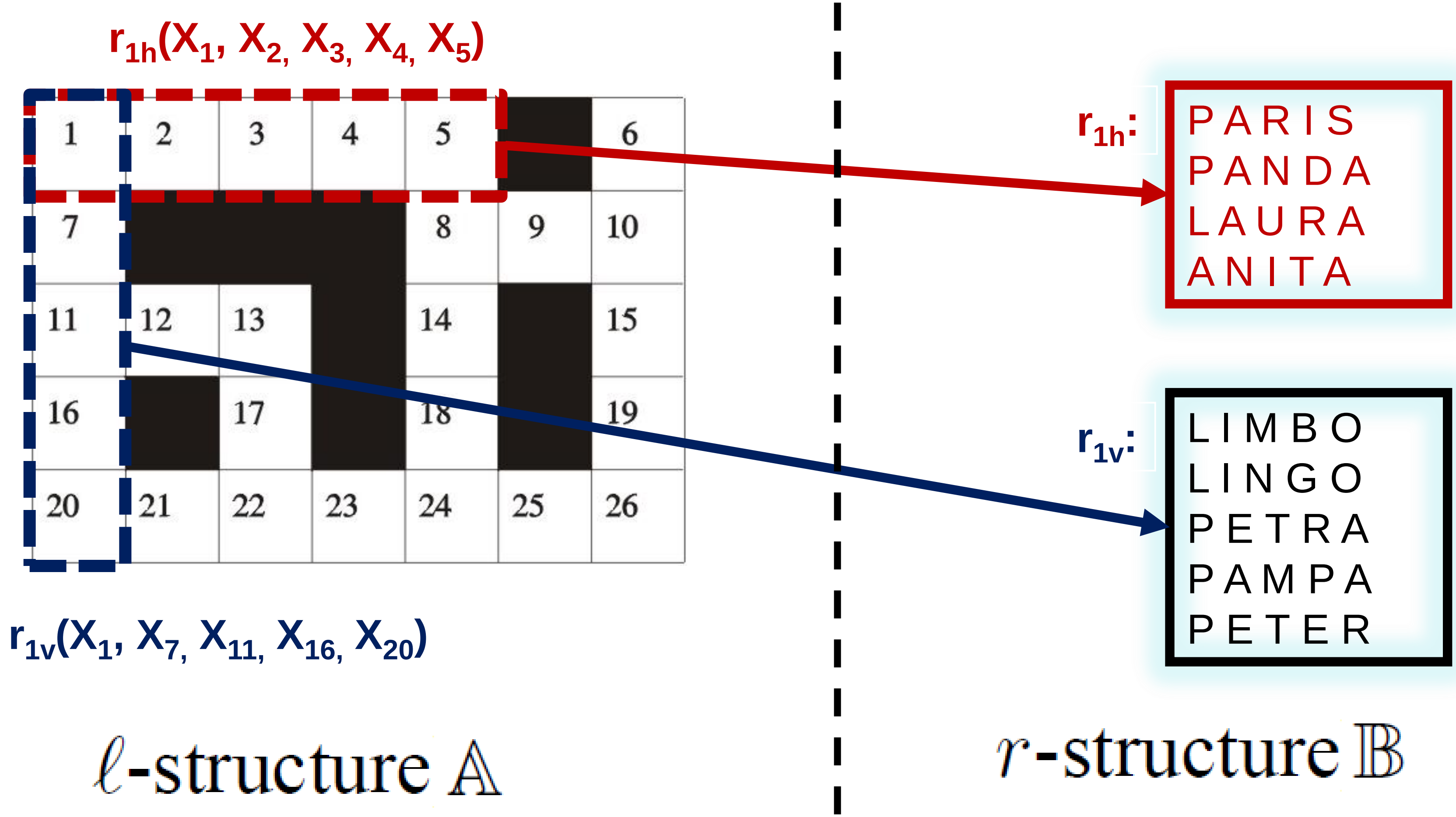
CSPs as Homomorphism Problems



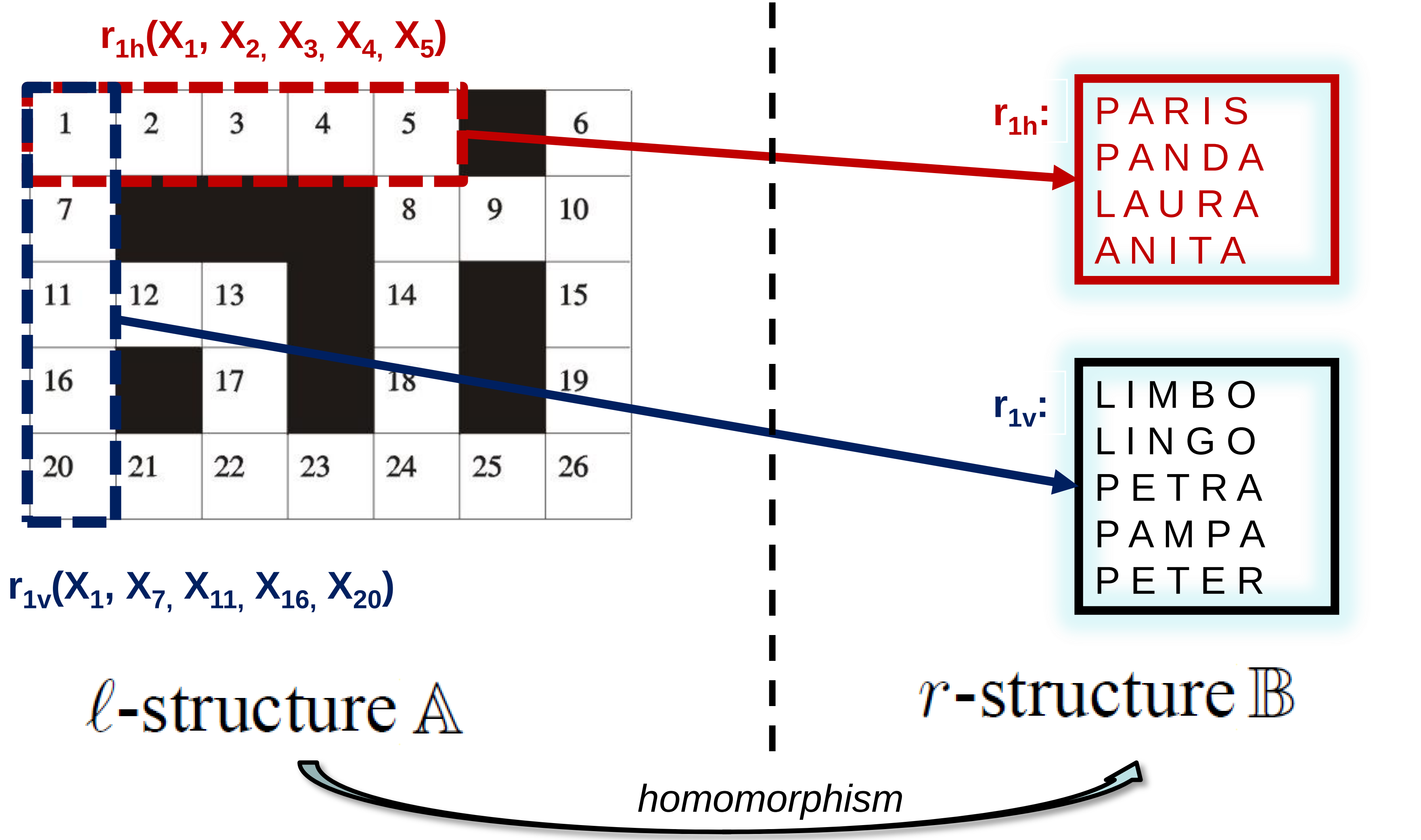
- Set of variables $\{X_1,\ldots,X_{26}\}$
- Set of constraint scopes

- Set of constraint relations

CSPs as Homomorphism Problems



CSPs as Homomorphism Problems



Questions

Questions

INPUT: CSP instance $(\mathbb{A}, \mathbb{B})$

- Decide the *existence* of a homomorphism
- *Enumerate* all the homomorphisms $\mathbb{A}^{\mathbb{B}}$
- For a set of variales X, enumerate the *projection* $\mathbb{A}^{\mathbb{B}}[X]$

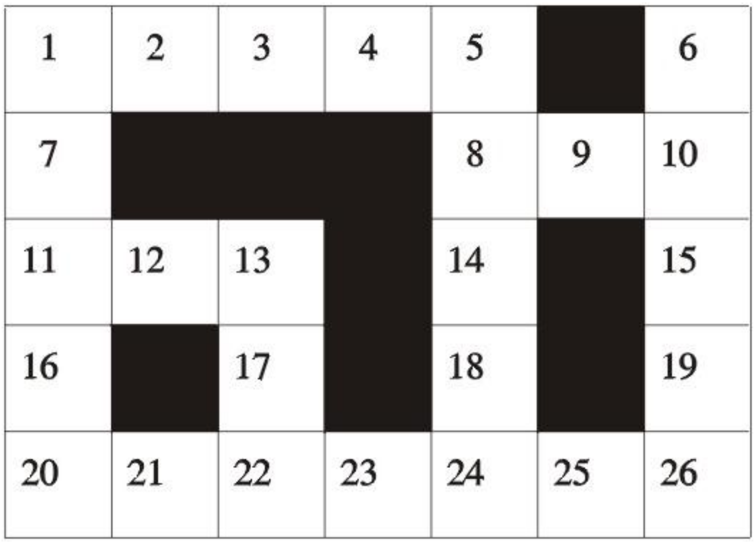
Questions

INPUT: CSP instance $(\mathbb{A}, \mathbb{B})$

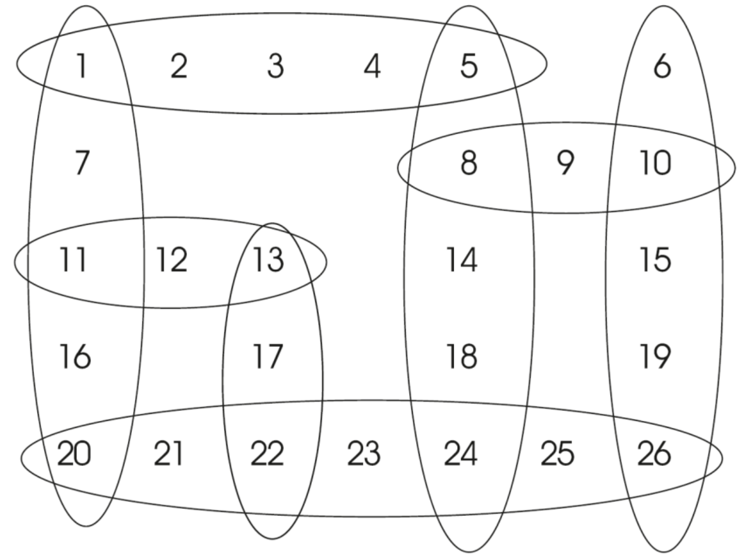
- Decide the *existence* of a homomorphism
- *Enumerate* all the homomorphisms $\mathbb{A}^{\mathbb{B}}$
- For a set of variales X, enumerate the *projection* $\mathbb{A}^{\mathbb{B}}[X]$

- Tractable decision and closure properties imply tractable search
  [R. Dechter and A. Itai, 1992]
- Non-uniform case
  [D. Cohen, 2004]

CSPs and Hypergraphs



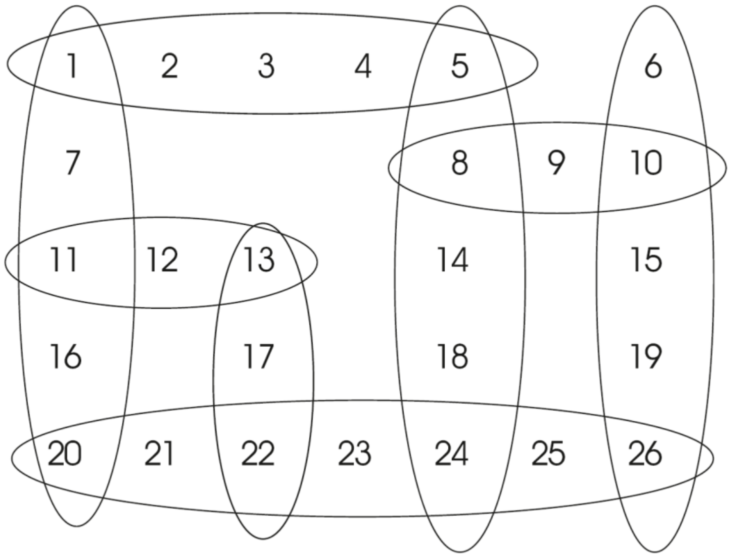
ℓ-structure $\mathbb{A}$



$\mathcal{H}_{\mathbb{A}}$

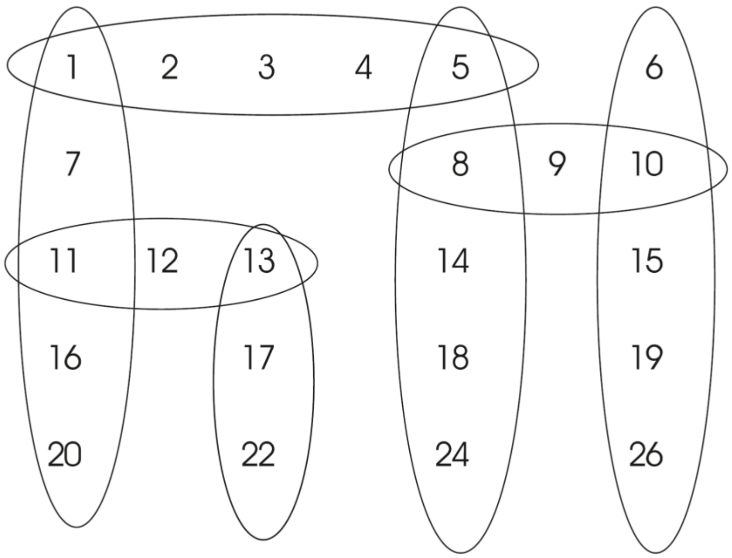
- Variables map to nodes
- Scopes map to hyperedges

Structurally Restricted CSPs



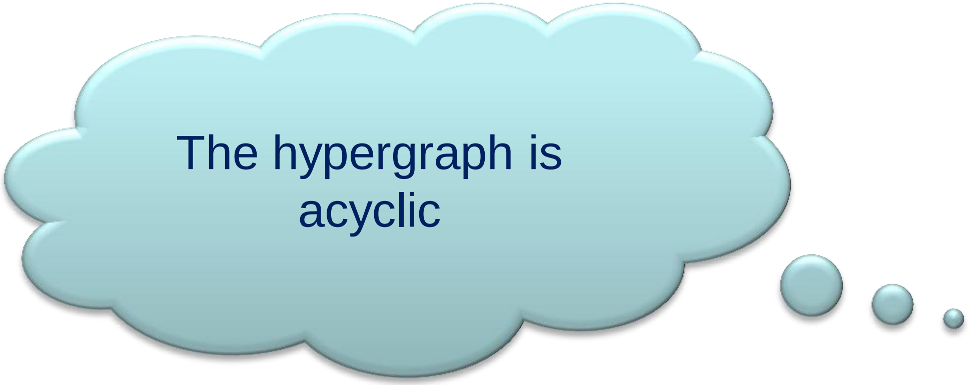
$\mathcal{H}_{\mathbb{A}}$

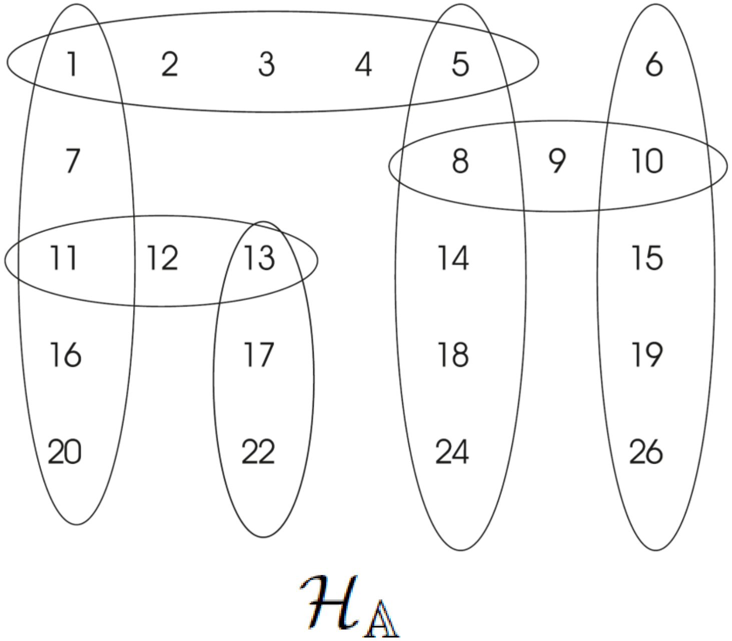
Structurally Restricted CSPs



$\mathcal{H}_{\mathbb{A}}$

Structurally Restricted CSPs

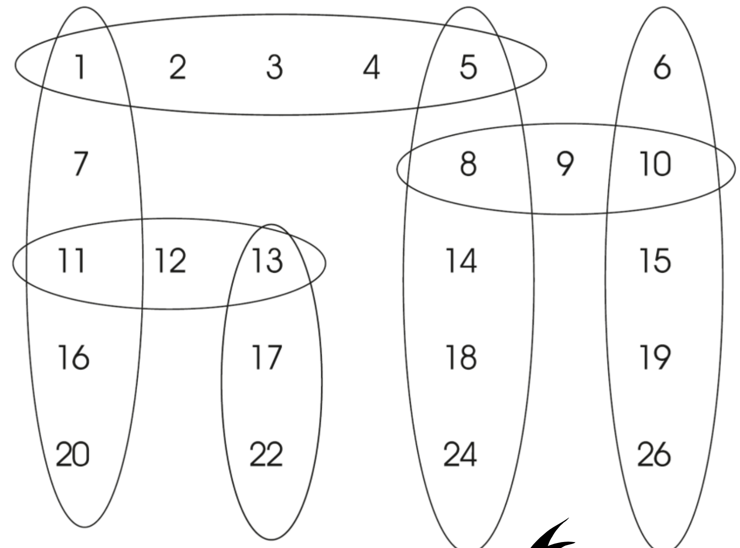




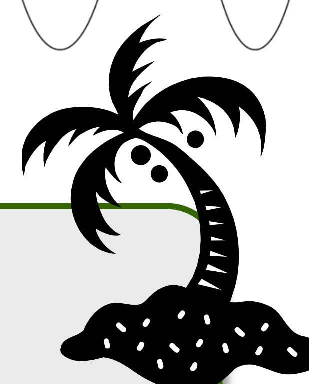
- Acyclicity is efficiently recognizable
- Acyclic CSPs can be efficiently solved
- Generalized arc consistency $\rightarrow$ Global consistency

Structurally Restricted CSPs

The hypergraph is acyclic

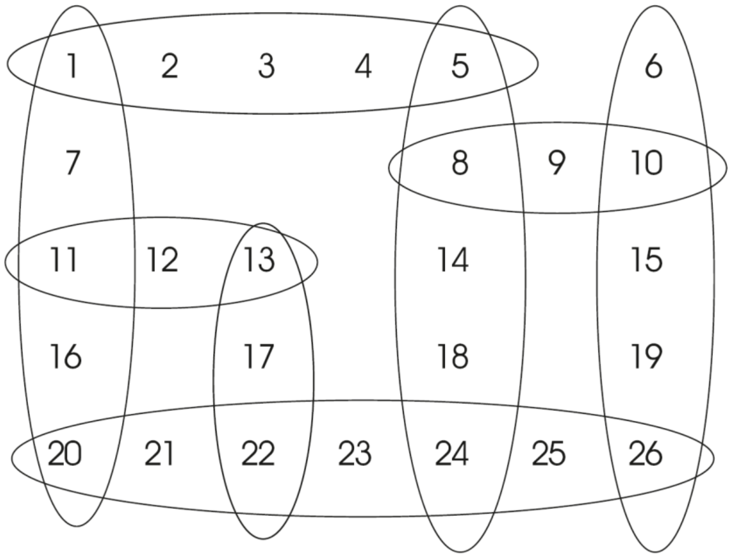


$\mathcal{H}_{\mathbb{A}}$



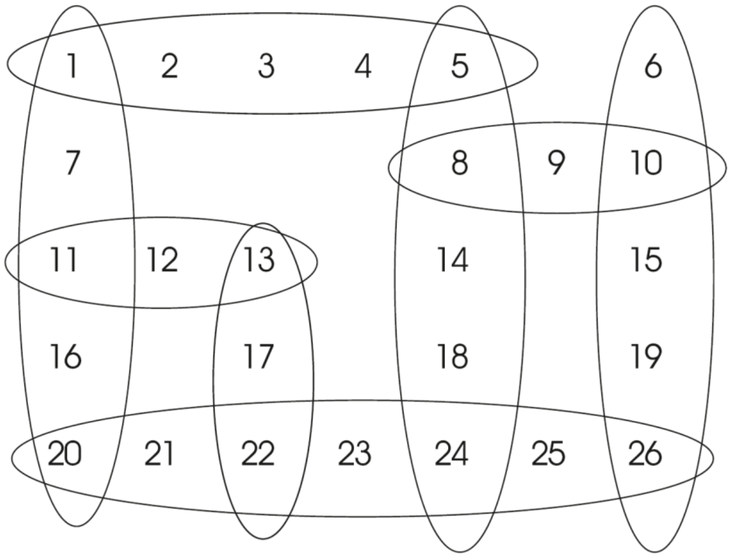
- Acyclicity is efficiently recognizable
- Acyclic CSPs can be efficiently solved
- Generalized arc consistency $\rightarrow$ Global consistency

Decomposition Methods

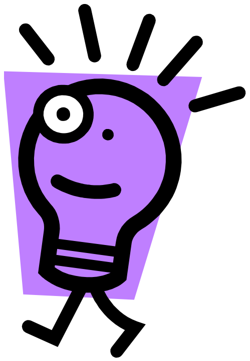


$\mathcal{H}_{\mathbb{A}}$

Decomposition Methods



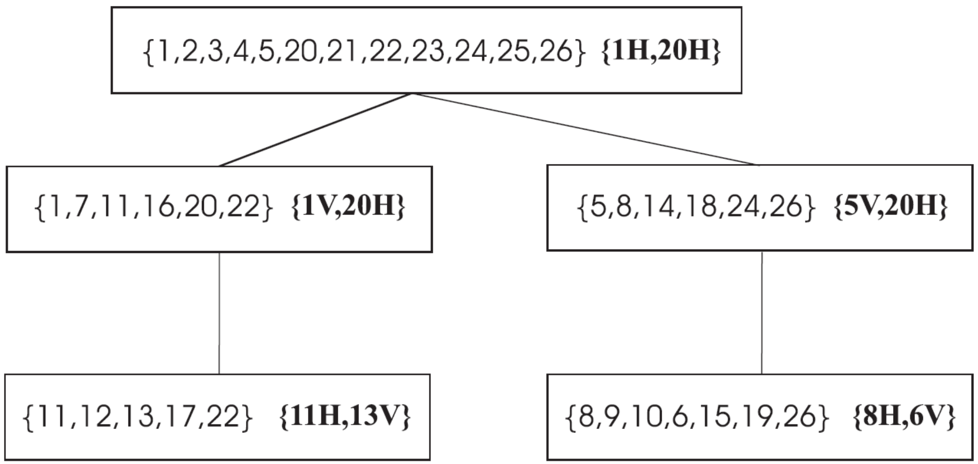
$\mathcal{H}_{\mathbb{A}}$

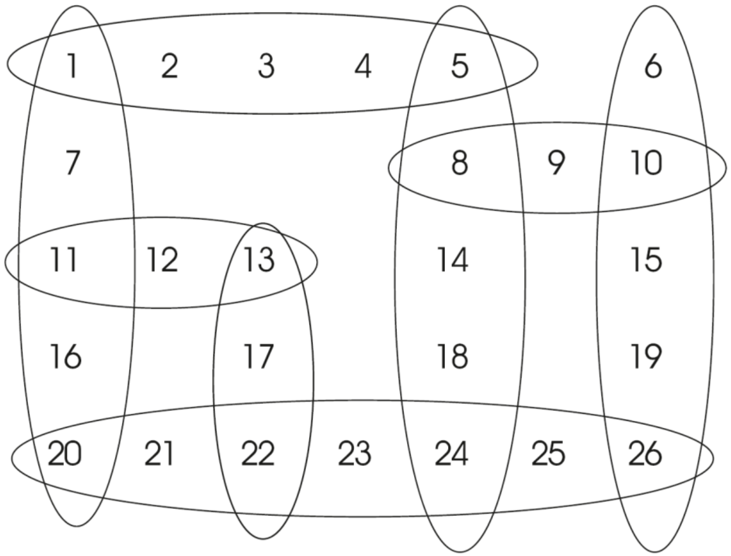


Transform the hypergraph into an acyclic one:

- Organize its edges (or nodes) in clusters
- Arrange the clusters as a tree,
  by satisfying the connectedness condition

Generalized Hypertree Decompositions

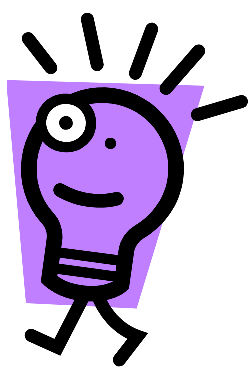




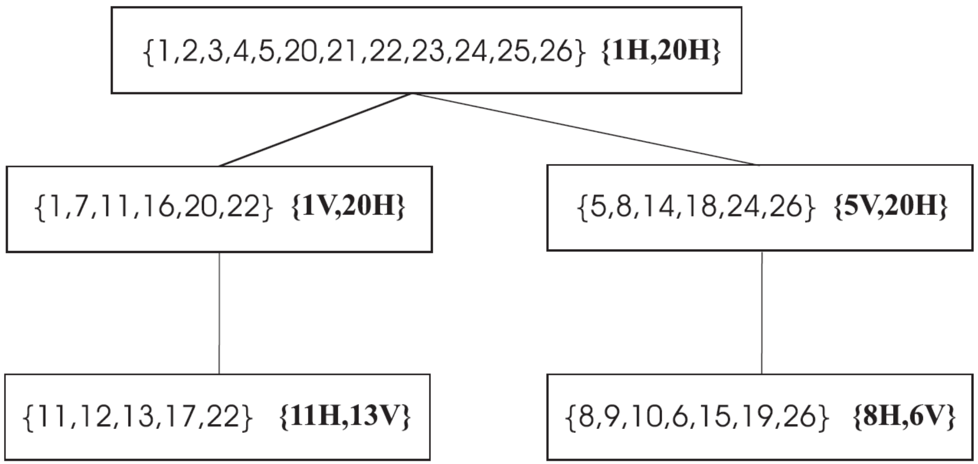
$\mathcal{H}_{\mathbb{A}}$

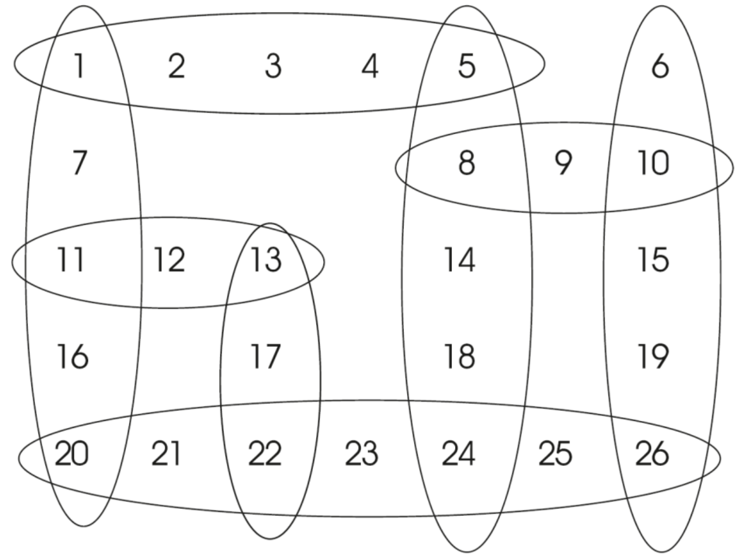
Transform the hypergraph into an acyclic one:

- Organize its edges (or nodes) in clusters
- Arrange the clusters as a tree,
  by satisfying the connectedness condition



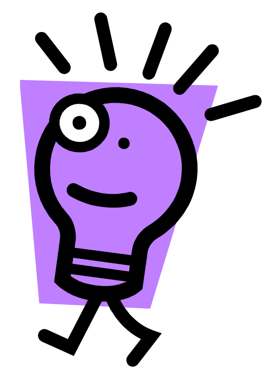
Generalized Hypertree Decompositions





$\mathcal{H}_{\mathbb{A}}$

Each cluster can be seen as a **subproblem**



Transform the hypergraph into an acyclic one:

- Organize its edges (or nodes) in clusters
- Arrange the clusters as a tree,
  by satisfying the connectedness condition

Outline

Decomposition Methods and Tree Projections

Enumeration without Certificates

Enumeration with Certificates

Outline

- Decomposition Methods and Tree Projections
- Enumeration without Certificates
- Enumeration with Certificates

Revisiting Decomposition Methods

CSP instance $(\mathbb{A}, \mathbb{B})$

$$\mathbb{A}_{\mathcal{V}} = \ell\text{-DM}(\mathbb{A}) \quad \mathbb{B}_{\mathcal{V}} = r\text{-DM}(\mathbb{A}, \mathbb{B})$$

Scopes

Solutions

Work on subproblems

Revisiting Decomposition Methods

CSP instance $(\mathbb{A}, \mathbb{B})$

$$\mathbb{A}_{\mathcal{V}} = \ell\text{-DM}(\mathbb{A}) \quad \mathbb{B}_{\mathcal{V}} = r\text{-DM}(\mathbb{A}, \mathbb{B})$$

(Noticeable) Examples

CSP instance $(\mathbb{A}, \mathbb{B})$

$$\mathbb{A}_{\mathcal{V}} = \ell\text{-DM}(\mathbb{A}) \quad \mathbb{B}_{\mathcal{V}} = r\text{-DM}(\mathbb{A}, \mathbb{B})$$

- *Treewidth*: take all views that can be computed with at most k variables
- *Generalized hypertree width*: take all views that can be computed by joining at most k atoms (k query views)
- *Fractional hypertree width*: take all views that can be computed through subproblems having fractional cover at most k (or use Marx's $O(k^3)$ approximation to have polynomially many views)

Acyclicity in Decomposition Methods

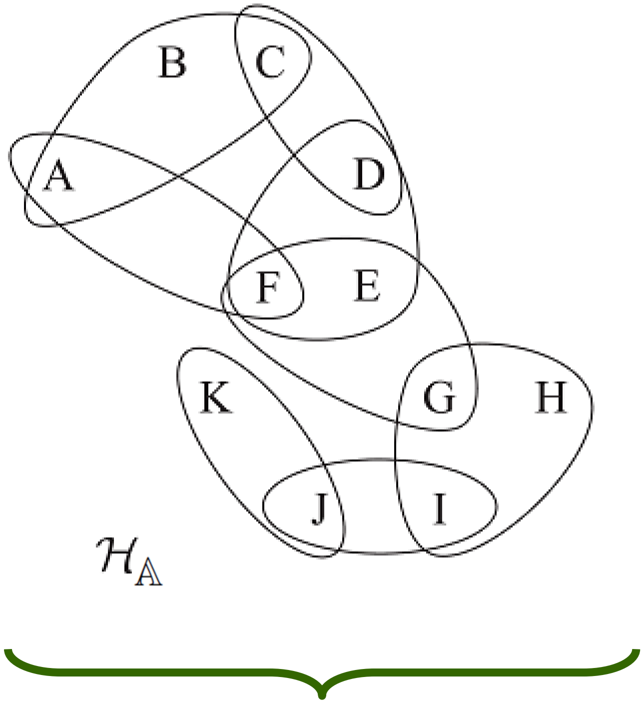
CSP instance $(\mathbb{A}, \mathbb{B})$

$$\mathbb{A}_{\mathcal{V}} = \ell\text{-DM}(\mathbb{A}) \quad \mathbb{B}_{\mathcal{V}} = r\text{-DM}(\mathbb{A}, \mathbb{B})$$

Working on subproblems is not necessarily beneficial…

Tree Projections (by Example)

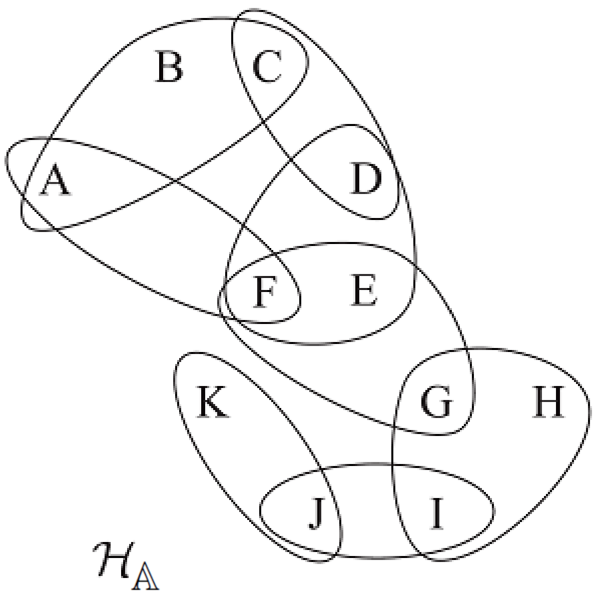
$\mathbb{A}$: $r_1(A,B,C)$ $r_2(A,F)$ $r_3(C,D)$ $r_4(D,E,F)$
$r_5(E,F,G)$ $r_6(G,H,I)$ $r_7(I,J)$ $r_8(J,K)$



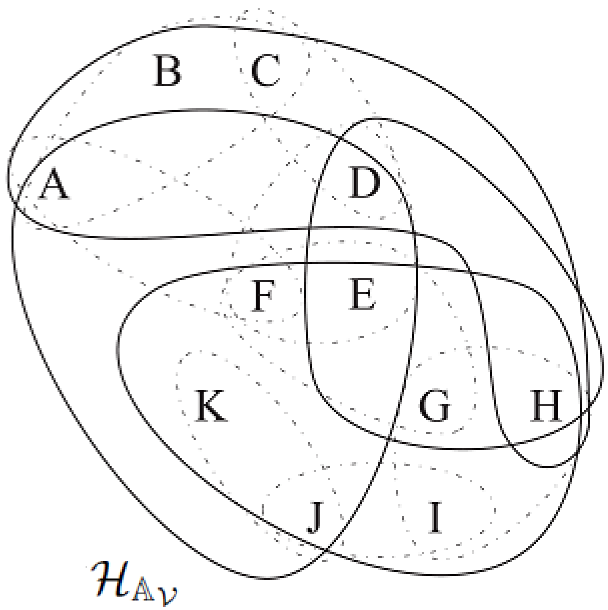
Structure of the CSP

Tree Projections (by Example)

$\mathbb{A}: \quad r_1(A,B,C) \quad r_2(A,F) \quad r_3(C,D) \quad r_4(D,E,F)$
$\quad r_5(E,F,G) \quad r_6(G,H,I) \quad r_7(I,J) \quad r_8(J,K)$



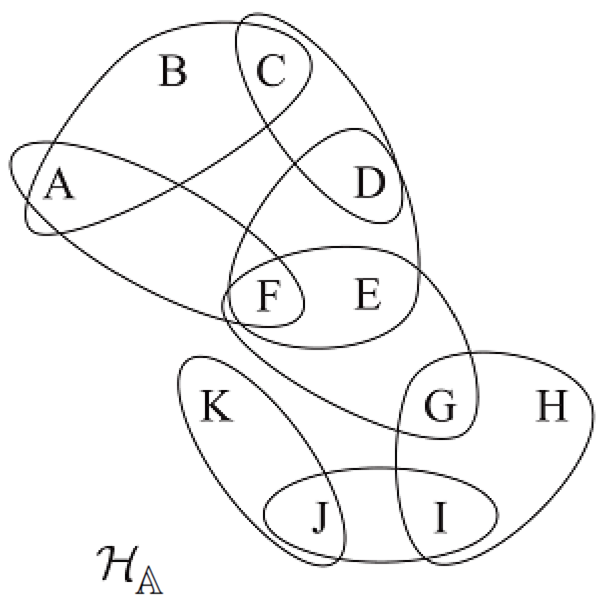
Structure of the CSP



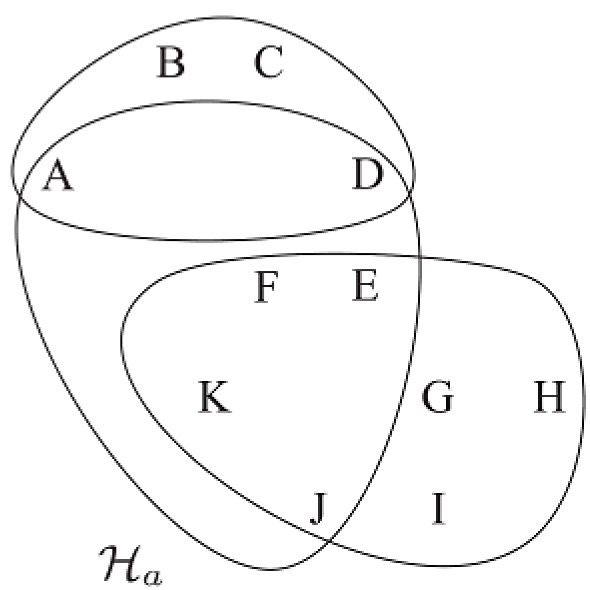
Available Views

Tree Projections (by Example)

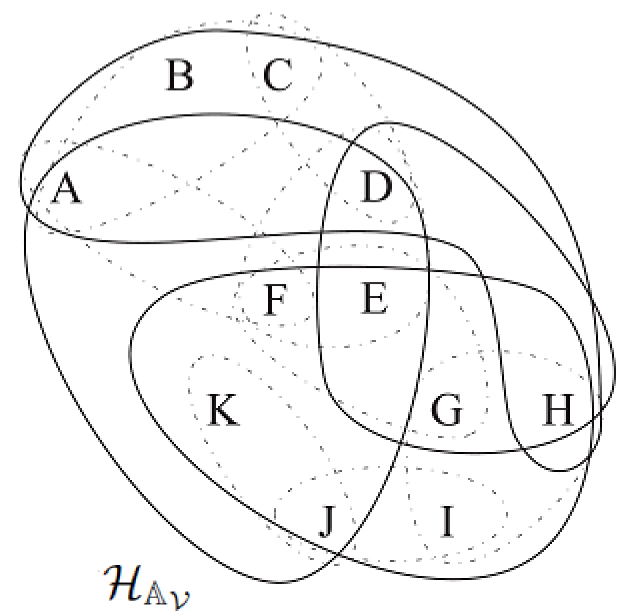
$$\mathbb{A}: \quad r_1(A,B,C) \quad r_2(A,F) \quad r_3(C,D) \quad r_4(D,E,F)$$
$$r_5(E,F,G) \quad r_6(G,H,I) \quad r_7(I,J) \quad r_8(J,K)$$



Structure of the CSP



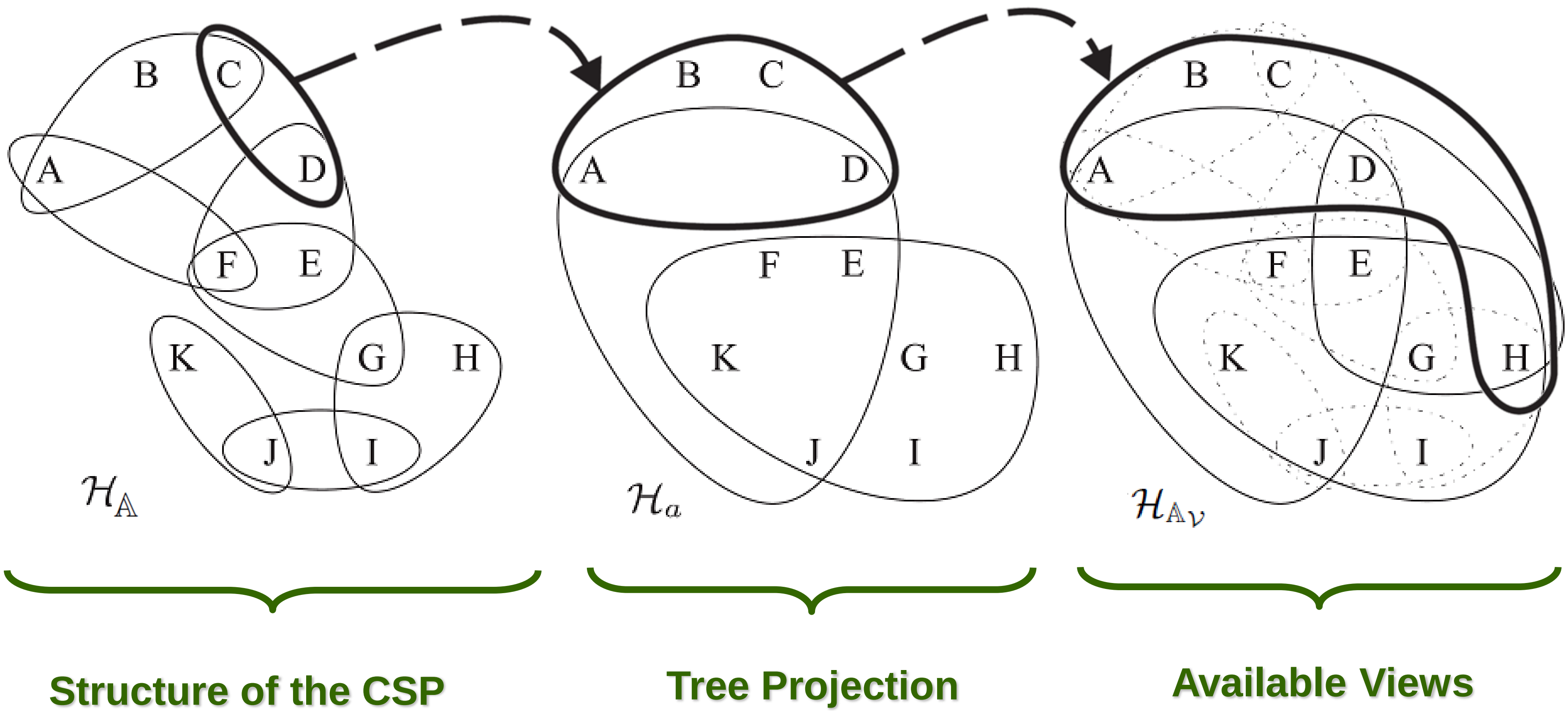
Tree Projection



Available Views

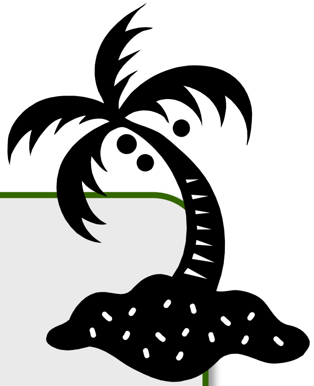
Tree Projections (by Example)

$\mathbb{A}$: $r_1(A,B,C)$ $r_2(A,F)$ $r_3(C,D)$ $r_4(D,E,F)$
$r_5(E,F,G)$ $r_6(G,H,I)$ $r_7(I,J)$ $r_8(J,K)$



From Acyclicity to Tree Projections

- Deciding whether a tree projection exists is NP-complete
  [G. Gottlob, Z. Miklós, and T. Schwentick, 2009]
- The existence of a tree projection ensures polynomial-time solvability
  [N. Goodman and O. Shmueli, 1984], [Y. Sagiv and O. Shmueli, 1993]



- Acyclicity is efficiently recognizable
- Acyclic CSPs can be efficiently solved
- Generalized arc consistency $\rightarrow$ Global consistency

Outline

- Decomposition Methods and Tree Projections
- **Enumeration without Certificates**
- Enumeration with Certificates

The Algorithm: Backtracking and Propagation

Input: An ECSP instance $(\mathbb{A}, \mathbb{B}, O)$, where $O = \{X_1, \ldots, X_m\}$;
Output: $\mathbb{A}^{\mathbb{B}}[O]$;
Method: update $(\mathbb{A}, \mathbb{B}, O)$ with any of its domain-restricted versions;
let $\mathbb{A}_{\mathcal{V}} := \ell\text{-}\mathtt{DM}(\mathbb{A})$, $\mathbb{B}_{\mathcal{V}} := r\text{-}\mathtt{DM}(\mathbb{A}, \mathbb{B})$;
invoke $\mathtt{Propagate}(1, (\mathbb{A}_{\mathcal{V}}, \mathbb{B}_{\mathcal{V}}), m, \langle\rangle)$;

Procedure $\mathtt{Propagate}(i$: integer, $(\mathbb{A}_{\mathcal{V}}, \mathbb{B}_{\mathcal{V}})$: pair of structures, m: integer, $\langle a_1, \ldots, a_{i-1}\rangle$: tuple of values in A^i);
begin
let $\mathbb{B}'_{\mathcal{V}} := \mathtt{GAC}(\mathbb{A}_{\mathcal{V}}, \mathbb{B}_{\mathcal{V}})$;
let $activeValues := dom(X_i)^{\mathbb{B}'_{\mathcal{V}}}$;
for each element $\langle a_i \rangle \in activeValues$ **do**
  if $i = m$ **then**
    output $\langle a_1, \ldots, a_{m-1}, a_m\rangle$;
  else
    update $dom(X_i)^{\mathbb{B}'_{\mathcal{V}}}$ with $\{\langle a_i\rangle\}$; /* X_i *is fixed to value* a_i */
    $\mathtt{Propagate}(i+1, (\mathbb{A}_{\mathcal{V}}, \mathbb{B}'_{\mathcal{V}}), m, \langle a_1, \ldots, a_{i-1}, a_i\rangle)$;
end.

The Algorithm: Backtracking and Propagation

Input: An ECSP instance $(\mathbb{A}, \mathbb{B}, O)$, where $O = \{X_1, \ldots, X_m\}$;
Output: $\mathbb{A}^{\mathbb{B}}[O]$;
Method: update $(\mathbb{A}, \mathbb{B}, O)$ with any of its domain-restricted versions;
let $\mathbb{A}_{\mathcal{V}} := \ell\text{-}\mathtt{DM}(\mathbb{A})$, $\mathbb{B}_{\mathcal{V}} := r\text{-}\mathtt{DM}(\mathbb{A}, \mathbb{B})$;
invoke $\mathtt{Propagate}(1, (\mathbb{A}_{\mathcal{V}}, \mathbb{B}_{\mathcal{V}}), m, \langle\rangle)$;

Procedure $\mathtt{Propagate}(i$: integer, $(\mathbb{A}_{\mathcal{V}}, \mathbb{B}_{\mathcal{V}})$: pair of structures, m: integer,
$\langle a_1, \ldots, a_{i-1}\rangle$: tuple of values in A^i);
begin
let $\mathbb{B}'_{\mathcal{V}} := \mathtt{GAC}(\mathbb{A}_{\mathcal{V}}, \mathbb{B}_{\mathcal{V}})$;
let $\mathit{activeValues} := \mathit{dom}(X_i)^{\mathbb{B}'_{\mathcal{V}}}$;
for each element $\langle a_i\rangle \in \mathit{activeValues}$ **do**
  if $i = m$ **then**
    output $\langle a_1, \ldots, a_{m-1}, a_m\rangle$;
  else
    update $\mathit{dom}(X_i)^{\mathbb{B}'_{\mathcal{V}}}$ with $\{\langle a_i\rangle\}$; /* X_i *is fixed to value* a_i */
    $\mathtt{Propagate}(i+1, (\mathbb{A}_{\mathcal{V}}, \mathbb{B}'_{\mathcal{V}}), m, \langle a_1, \ldots, a_{i-1}, a_i\rangle)$;
end.

The Algorithm: Backtracking and Propagation

Input: An ECSP instance $(\mathbb{A}, \mathbb{B}, O)$, where $O = \{X_1, \ldots, X_m\}$;
Output: $\mathbb{A}^{\mathbb{B}}[O]$;
Method: update $(\mathbb{A}, \mathbb{B}, O)$ with any of its domain-restricted versions;
let $\mathbb{A}_{\mathcal{V}} := \ell\text{-}\mathtt{DM}(\mathbb{A})$, $\mathbb{B}_{\mathcal{V}} := r\text{-}\mathtt{DM}(\mathbb{A}, \mathbb{B})$;
invoke $\mathtt{Propagate}(1, (\mathbb{A}_{\mathcal{V}}, \mathbb{B}_{\mathcal{V}}), m, \langle\rangle)$;

Procedure $\mathtt{Propagate}(i$: integer, $(\mathbb{A}_{\mathcal{V}}, \mathbb{B}_{\mathcal{V}})$: pair of structures, m: integer,
$\langle a_1, \ldots, a_{i-1}\rangle$: tuple of values in A^i);

begin
let $\mathbb{B}'_{\mathcal{V}} := \mathtt{GAC}(\mathbb{A}_{\mathcal{V}}, \mathbb{B}_{\mathcal{V}})$;
let $activeValues := dom(X_i)^{\mathbb{B}'_{\mathcal{V}}}$;
for each element $\langle a_i\rangle \in activeValues$ **do**
  if $i = m$ **then**
    output $\langle a_1, \ldots, a_{m-1}, a_m\rangle$;
  else
    update $dom(X_i)^{\mathbb{B}'_{\mathcal{V}}}$ with $\{\langle a_i\rangle\}$; /* X_i *is fixed to value* a_i */
    $\mathtt{Propagate}(i+1, (\mathbb{A}_{\mathcal{V}}, \mathbb{B}'_{\mathcal{V}}), m, \langle a_1, \ldots, a_{i-1}, a_i\rangle)$;
end.

The Algorithm: Backtracking and Propagation

Input: An ECSP instance $(\mathbb{A}, \mathbb{B}, O)$, where $O = \{X_1, \ldots, X_m\}$;
Output: $\mathbb{A}^{\mathbb{B}}[O]$;
Method: update $(\mathbb{A}, \mathbb{B}, O)$ with any of its domain-restricted versions;
let $\mathbb{A}_{\mathcal{V}} := \ell\text{-DM}(\mathbb{A})$, $\mathbb{B}_{\mathcal{V}} := r\text{-DM}(\mathbb{A}, \mathbb{B})$;
invoke $\texttt{Propagate}(1, (\mathbb{A}_{\mathcal{V}}, \mathbb{B}_{\mathcal{V}}), m, \langle\rangle)$;

Procedure $\texttt{Propagate}$(i: integer, $(\mathbb{A}_{\mathcal{V}}, \mathbb{B}_{\mathcal{V}})$: pair of structures, m: integer, $\langle a_1, \ldots, a_{i-1}\rangle$: tuple of values in A^i);

begin

let $\mathbb{B}'_{\mathcal{V}} := \texttt{GAC}(\mathbb{A}_{\mathcal{V}}, \mathbb{B}_{\mathcal{V}})$;
let $activeValues := dom(X_i)^{\mathbb{B}'_{\mathcal{V}}}$;
for each element $\langle a_i \rangle \in activeValues$ **do**
  if $i = m$ **then**
    output $\langle a_1, \ldots, a_{m-1}, a_m\rangle$;
  else
    update $dom(X_i)^{\mathbb{B}'_{\mathcal{V}}}$ with $\{\langle a_i \rangle\}$; /* *X_i is fixed to value a_i* */
    $\texttt{Propagate}(i+1, (\mathbb{A}_{\mathcal{V}}, \mathbb{B}'_{\mathcal{V}}), m, \langle a_1, \ldots, a_{i-1}, a_i\rangle)$;

end.

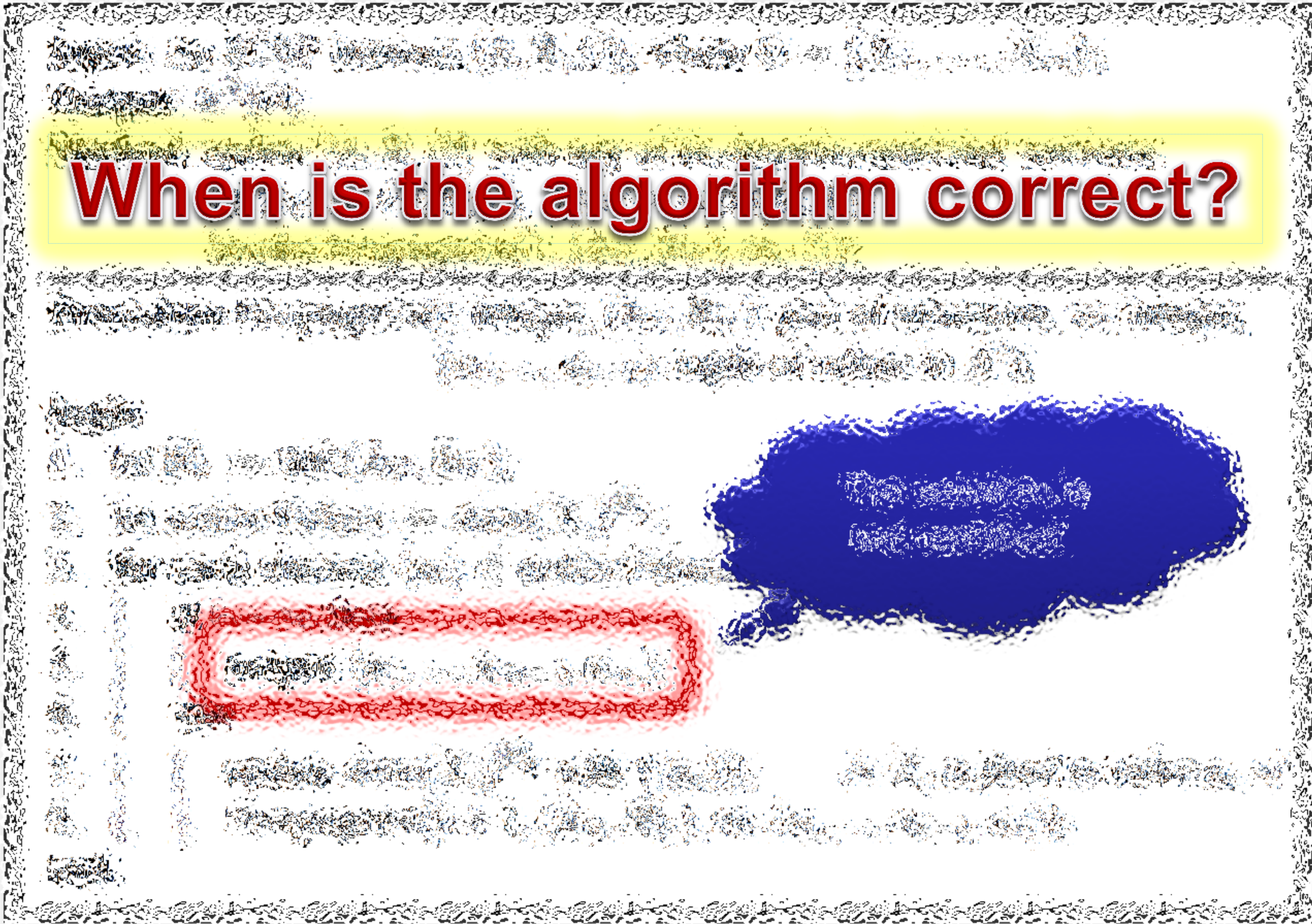
The Algorithm: Backtracking and Propagation

Input: An ECSP instance $(\mathbb{A}, \mathbb{B}, O)$, where $O = \{X_1, \ldots, X_m\}$;
Output: $\mathbb{A}^{\mathbb{B}}[O]$;
Method: update $(\mathbb{A}, \mathbb{B}, O)$ with any of its domain-restricted versions;
let $\mathbb{A}_{\mathcal{V}} := \ell\text{-}\mathtt{DM}(\mathbb{A})$, $\mathbb{B}_{\mathcal{V}} := r\text{-}\mathtt{DM}(\mathbb{A}, \mathbb{B})$;
invoke $\mathtt{Propagate}(1, (\mathbb{A}_{\mathcal{V}}, \mathbb{B}_{\mathcal{V}}), m, \langle\rangle)$;

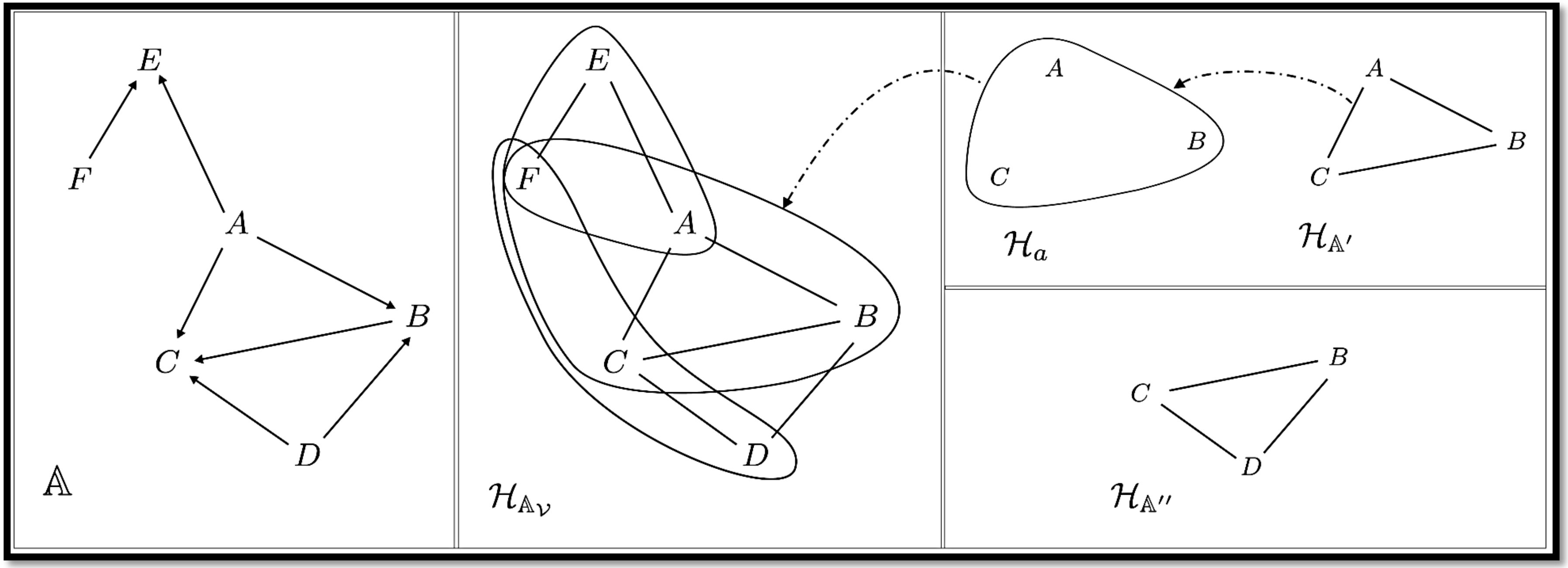
Procedure $\mathtt{Propagate}(i$: integer, $(\mathbb{A}_{\mathcal{V}}, \mathbb{B}_{\mathcal{V}})$: pair of structures, m: integer,
$\langle a_1, \ldots, a_{i-1}\rangle$: tuple of values in $A^i)$;
begin
let $\mathbb{B}'_{\mathcal{V}} := \mathtt{GAC}(\mathbb{A}_{\mathcal{V}}, \mathbb{B}_{\mathcal{V}})$;
let $activeValues := dom(X_i)^{\mathbb{B}'_{\mathcal{V}}}$;
for each element $\langle a_i\rangle \in activeValues$
  if $i = m$ **then**
    output $\langle a_1, \ldots, a_{m-1}, a_m\rangle$;
  else
    update $dom(X_i)^{\mathbb{B}'_{\mathcal{V}}}$ with $\{\langle a_i\rangle\}$;   /* X_i *is fixed to value* a_i */
    $\mathtt{Propagate}(i+1, (\mathbb{A}_{\mathcal{V}}, \mathbb{B}'_{\mathcal{V}}), m, \langle a_1, \ldots, a_{i-1}, a_i\rangle)$;
end.

The solution is
not certified!

The Algorithm: Backtracking and Propagation

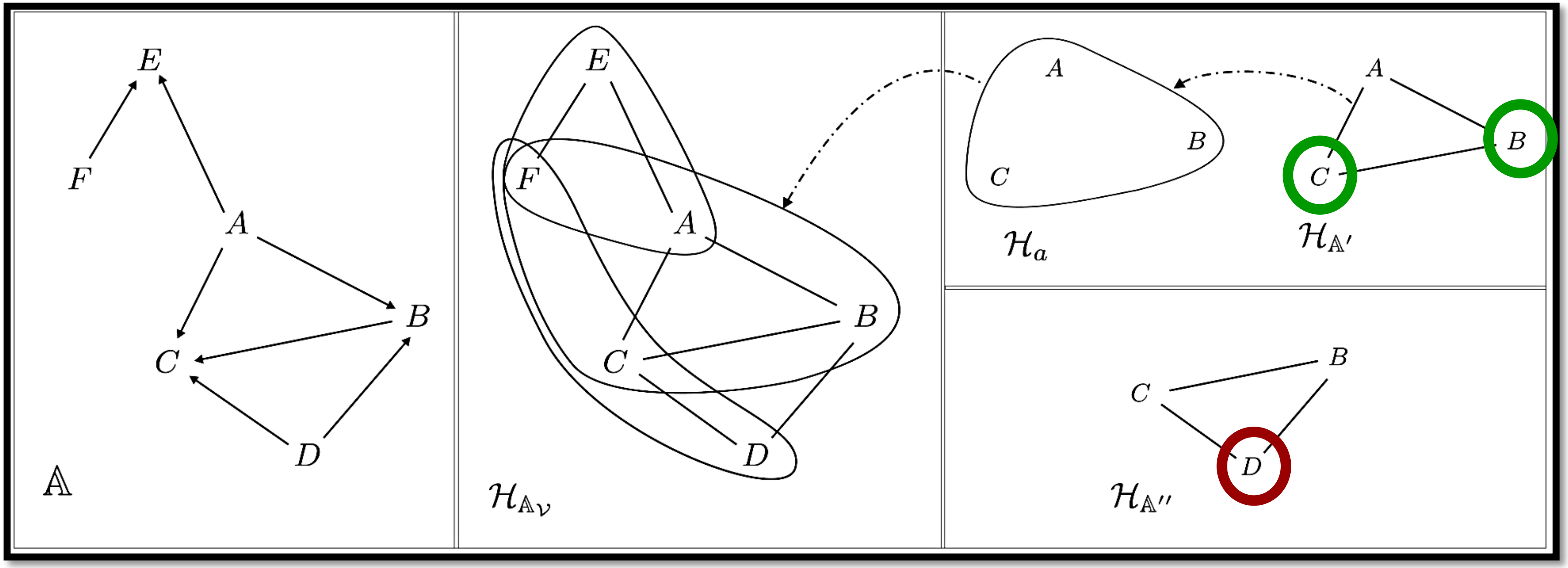


Tp-covering



E
F
A
B
C
D
𝔸
E
F
A
B
C
D
$\mathcal{H}_{\mathbb{A}_\mathcal{V}}$
A
B
C
$\mathcal{H}_a$
A
B
C
$\mathcal{H}_{\mathbb{A}'}$
B
C
D
$\mathcal{H}_{\mathbb{A}''}$

Tp-covering



{B,C} is individually tp-covered

{D} is not individually tp-covered

Tight Characterizations for the Correctness

When is the algorithm correct?

Thm.

Let $\mathbb{A}$ be an ℓ-structure, and let $O \subseteq A$ be a set of variables. The following are equivalent:

(1) O is individually tp-covered

(2) For every r-structure $\mathbb{B}$, $\texttt{ComputeAllSolutions}_{\mathrm{DM}}$ computes $\mathbb{A}^{\mathbb{B}}[O]$.

Proof: Discussion on the Base (Decision) Case

- The existence of a tree projection implies ”GAC $\rightarrow$ Global”
  [Y. Sagiv and O. Shmueli, 1983]

Proof: Discussion on the Base (Decision) Case

- The existence of a tree projection implies ”GAC $\rightarrow$ Global”
  [Y. Sagiv and O. Shmueli, 1983]

Over the views.

Proof: Discussion on the Base (Decision) Case

- The existence of a tree projection implies "GAC $\rightarrow$ Global"
  [Y. Sagiv and O. Shmueli, 1983]
- For *generalized hypertree decompositions*, the existence of a tree projection for the core of the left-structure implies "GAC $\rightarrow$ Global"
  [H. Chen and V. Dalmau, 2005]

Proof: Discussion on the Base (Decision) Case

- The existence of a tree projection implies ”GAC → Global”
  [Y. Sagiv and O. Shmueli, 1983]
- For *generalized hypertree decompositions*, the existence of a tree projection for the core of the left-structure implies ”GAC → Global”
  [H. Chen and V. Dalmau, 2005]

Over the views, i.e., k-consistency

Proof: Discussion on the Base (Decision) Case

- The existence of a tree projection implies ”GAC $\rightarrow$ Global”
  [Y. Sagiv and O. Shmueli, 1983]
- For *generalized hypertree decompositions*, the existence of a tree projection for the core of the left-structure implies ”GAC $\rightarrow$ Global”
  [H. Chen and V. Dalmau, 2005]
- For *tree decompositions*, and on classes of CSPs having *bounded arity*, the existence of a tree projection for the core of the left-structure is a sufficient and necessary condition to imply ”GAC $\rightarrow$ Global”
  [A. Atserias, A. Bulatov, and V. Dalmau, 2007]

Proof: Discussion on the Base (Decision) Case

- The existence of a tree projection implies ”GAC $\rightarrow$ Global”
  [Y. Sagiv and O. Shmueli, 1983]
- For *generalized hypertree decompositions*, the existence of a tree projection for the core of the left-structure implies ”GAC $\rightarrow$ Global”
  [H. Chen and V. Dalmau, 2005]
- For *tree decompositions*, and on classes of CSPs having *bounded arity*, the existence of a tree projection for the core of the left-structure is a sufficient and necessary condition to imply ”GAC $\rightarrow$ Global”
  [A. Atserias, A. Bulatov, and V. Dalmau, 2007]

- For *generalized hypertree decompositions*, the existence of a tree projection for the core is also necessary to imply ”GAC $\rightarrow$ Global”
  [G. Greco and F. Scarcello, 2010]

Proof: Discussion on the Base (Decision) Case

Thm [G. Greco and F. Scarcello, 2010].

Let $\mathbb{A}$ be an ℓ-structure, and let $\mathbb{A}_{\mathcal{V}}$ be a v-structure.
The following are equivalent:

(1) There is a core $\mathbb{A}'$ of $\mathbb{A}$ such that $(\mathcal{H}_{\mathbb{A}'}, \mathcal{H}_{\mathbb{A}_{\mathcal{V}}})$ has a tree projection

(2) For every r-structure $\mathbb{B}$, for every r-structure $\mathbb{B}_{\mathcal{V}}$ that is legal, enforcing generalized arc consistency on $\mathbb{B}_{\mathcal{V}}$ is a correct decision procedure

- For *generalized hypertree decompositions*, the existence of a tree projection for the core is also necessary to imply "GAC $\rightarrow$ Global" [G. Greco and F. Scarcello, 2010]

Complexity Issues of $\texttt{ComputeAllSolutions}_{\mathrm{DM}}$

Complexity Issues of ComputeAllSolutions$_{DM}$

- If *O* is tp-covered, the algorithm runs **W**ith **P**olynomial **D**elay…

Complexity Issues of $\texttt{ComputeAllSolutions}_{\mathrm{DM}}$

- If *O* is tp-covered, the algorithm runs **W**ith **P**olynomial **D**elay…
- …and the result is essentially tight

Thm.

Assume $FPT \neq W[1]$. Let $\mathbf{A}$ be any class of ℓ-structures of bounded arity. Then, the following are equivalent:

(1) $\mathbf{A}$ *has bounded treewdith modulo homomorphic equivalence;*
(2) For every $\mathbb{A} \in \mathbf{A}$, for every r-structure $\mathbb{B}$, and for every set of variables $O \subseteq drv(\mathbb{A})$, the ECSP *instance $(\mathbb{A}, \mathbb{B}, O)$ is solvable WPD.*

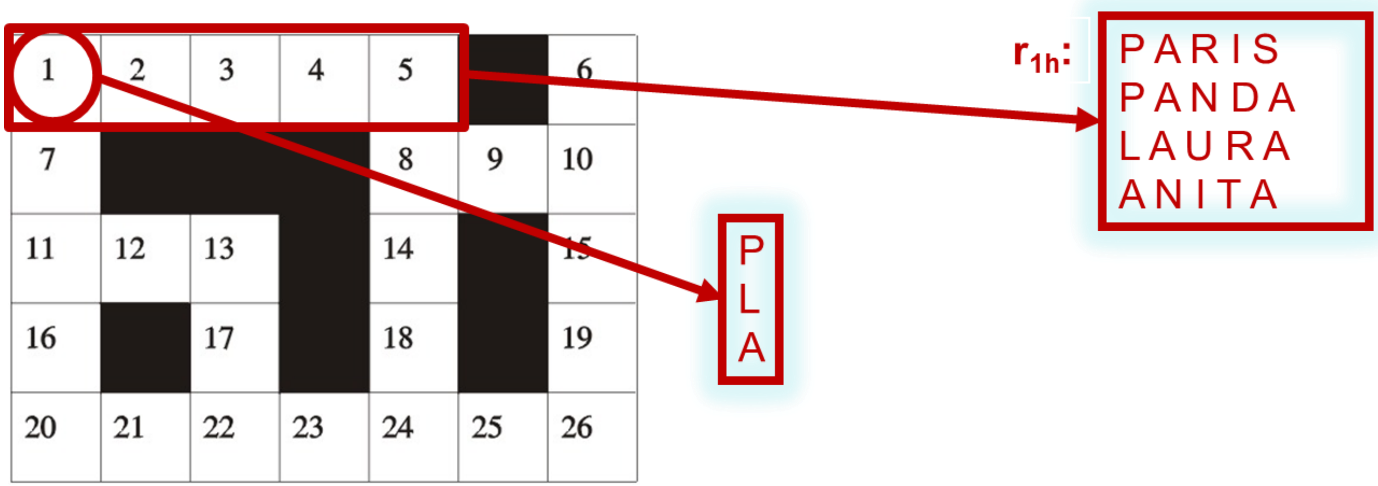
Complexity Issues of `ComputeAllSolutions`$_{DM}$

- If *O* is tp-covered, the algorithm runs **W**ith **P**olynomial **D**elay…
- …and the result is essentially tight

Thm.

Assume $FPT \neq W[1]$. *Let* **A** *be any class of* ℓ*-structures of bounded arity. Then, the following are equivalent:*

(1) **A** *has bounded treewdith modulo homomorphic equivalence;*
(2) For every $\mathbb{A} \in$ **A**, *for every* r*-structure* $\mathbb{B}$, *and for every set of variables* $O \subseteq drv(\mathbb{A})$, *the* ECSP *instance* $(\mathbb{A}, \mathbb{B}, O)$ *is solvable WPD.*



Complexity Issues of `ComputeAllSolutions`$_{DM}$

- If *O* is tp-covered, the algorithm runs **W**ith **P**olynomial **D**elay…
- …and the result is essentially tight

Thm.

Assume $FPT \neq W[1]$. *Let* $\mathbf{A}$ *be any class of* ℓ*-structures of bounded arity. Then, the following are equivalent:*

(1) $\mathbf{A}$ *has bounded treewdith modulo homomorphic equivalence;*
(2) For every $\mathbb{A} \in \mathbf{A}$*, for every* τ*-structure* $\mathbb{B}$*, and for every set of variables* $O \subseteq drv(\mathbb{A})$*, the* ECSP *instance* $(\mathbb{A}, \mathbb{B}, O)$ *is solvable WPD.*

There is no efficient algorithm for the no-promise problem

Outline

Decomposition Methods and Tree Projections

Enumeration without Certificates

Enumeration with Certificates

A Simple Modification

Input: An ECSP instance $(\mathbb{A}, \mathbb{B}, O)$, where $O = \{X_1, \ldots, X_m\}$;
Output: for each solution $h \in \mathbb{A}^{\mathbb{B}}[O]$, a certified solution (h, h');
Method: let $A = \{X_1, \ldots, X_m, X_{m+1}, \ldots, X_n\}$ be the variables of $\mathbb{A}$;
update $(\mathbb{A}, \mathbb{B}, A)$ with any of its domain restricted versions;
let $\mathbb{A}_{\mathcal{V}} := \ell\text{-DM}(\mathbb{A})$, $\mathbb{B}_{\mathcal{V}} := r\text{-DM}(\mathbb{A}, \mathbb{B})$;
invoke $\texttt{CPropagate}(1, (\mathbb{A}_{\mathcal{V}}, \mathbb{B}_{\mathcal{V}}), m, \langle\rangle)$;

Procedure $\texttt{CPropagate}(i$: integer, $(\mathbb{A}_{\mathcal{V}}, \mathbb{B}_{\mathcal{V}})$: pair of structures, m: integer,
$\langle a_1, \ldots, a_{i-1}\rangle$: tuple of values in A^i);

begin
let $\mathbb{B}'_{\mathcal{V}} := \texttt{GAC}(\mathbb{A}_{\mathcal{V}}, \mathbb{B}_{\mathcal{V}})$;
if $i > 1$ and $\mathbb{B}'_{\mathcal{V}}$ is empty **then** output "DM failure" and HALT;
let $activeValues := dom(X_i)^{\mathbb{B}'_{\mathcal{V}}}$;
for each element $\langle a_i \rangle \in activeValues$ **do**
  if $i = n$ **then**
    output the certified solution $(\langle a_1, \ldots, a_m\rangle, \langle a_{m+1}, \ldots, a_n\rangle)$;
  else
    update $dom(X_i)^{\mathbb{B}'_{\mathcal{V}}}$ with $\{\langle a_i \rangle\}$;   /* X_i *is fixed to value* a_i */
    $\texttt{CPropagate}(i+1, (\mathbb{A}_{\mathcal{V}}, \mathbb{B}'_{\mathcal{V}}), m, \langle a_1, \ldots, a_{i-1}, a_i\rangle)$;
    if $i > m$ **then** BREAK;
end.

A Simple Modification

Input: An ECSP instance $(\mathbb{A}, \mathbb{B}, O)$, where $O = \{X_1, \ldots, X_m\}$;
Output: for each solution $h \in \mathbb{A}^{\mathbb{B}}[O]$, a certified solution (h, h');
Method: let $A = \{X_1, \ldots, X_m, X_{m+1}, \ldots, X_n\}$ be the variables of $\mathbb{A}$;
update $(\mathbb{A}, \mathbb{B}, A)$ with any of its domain restricted versions;
let $\mathbb{A}_{\mathcal{V}} := \ell\text{-DM}(\mathbb{A})$, $\mathbb{B}_{\mathcal{V}} := r\text{-DM}(\mathbb{A}, \mathbb{B})$;
invoke $\texttt{CPropagate}(1, (\mathbb{A}_{\mathcal{V}}, \mathbb{B}_{\mathcal{V}}), m, \langle\rangle)$;

Procedure $\texttt{CPropagate}(i$: integer, $(\mathbb{A}_{\mathcal{V}}, \mathbb{B}_{\mathcal{V}})$: pair of structures, m: integer,
$\langle a_1, \ldots, a_{i-1}\rangle$: tuple of values in A^i);

begin
let $\mathbb{B}'_{\mathcal{V}} := \texttt{GAC}(\mathbb{A}_{\mathcal{V}}, \mathbb{B}_{\mathcal{V}})$;
if $i > 1$ and $\mathbb{B}'_{\mathcal{V}}$ is empty **then** output "DM failure" and HALT;
let $activeValues := dom(X_i)^{\mathbb{B}'_{\mathcal{V}}}$;
for each element $\langle a_i\rangle \in activeValues$ **do**
  if $i = n$ **then**
    output the certified solution $(\langle a_1, \ldots, a_m\rangle, \langle a_{m+1}, \ldots, a_n\rangle)$;
  else
    update $dom(X_i)^{\mathbb{B}'_{\mathcal{V}}}$ with $\{\langle a_i\rangle\}$;   /* X_i *is fixed to value* a_i */
    $\texttt{CPropagate}(i+1, (\mathbb{A}_{\mathcal{V}}, \mathbb{B}'_{\mathcal{V}}), m, \langle a_1, \ldots, a_{i-1}, a_i\rangle)$;
    if $i > m$ **then** BREAK;
end.

A Simple Modification

Input: An ECSP instance $(\mathbb{A}, \mathbb{B}, O)$, where $O = \{X_1, \ldots, X_m\}$;
Output: for each solution $h \in \mathbb{A}^{\mathbb{B}}[O]$, a certified solution (h, h');
Method: let $A = \{X_1, \ldots, X_m, X_{m+1}, \ldots, X_n\}$ be the variables of $\mathbb{A}$;
update $(\mathbb{A}, \mathbb{B}, A)$ with any of its domain restricted versions;
let $\mathbb{A}_{\mathcal{V}} := \ell\text{-}\mathtt{DM}(\mathbb{A})$, $\mathbb{B}_{\mathcal{V}} := r\text{-}\mathtt{DM}(\mathbb{A}, \mathbb{B})$;
invoke $\mathtt{CPropagate}(1, (\mathbb{A}_{\mathcal{V}}, \mathbb{B}_{\mathcal{V}}), m, \langle\rangle)$;

Procedure $\mathtt{CPropagate}$(i: integer, $(\mathbb{A}_{\mathcal{V}}, \mathbb{B}_{\mathcal{V}})$: pair of structures, m: integer,
$\langle a_1, \ldots, a_{i-1}\rangle$: tuple of values in A^i);

begin

let $\mathbb{B}'_{\mathcal{V}} := \mathtt{GAC}(\mathbb{A}_{\mathcal{V}}, \mathbb{B}_{\mathcal{V}})$;
if $i > 1$ and $\mathbb{B}'_{\mathcal{V}}$ is empty **then** output "DM failure" and HALT;
let $activeValues := dom(X_i)^{\mathbb{B}'_{\mathcal{V}}}$;
for each element $\langle a_i \rangle \in activeValues$ **do**
  if $i = n$ **then**
    output the certified solution $(\langle a_1, \ldots, a_m\rangle, \langle a_{m+1}, \ldots, a_n\rangle)$;
  else
    update $dom(X_i)^{\mathbb{B}'_{\mathcal{V}}}$ with $\{\langle a_i\rangle\}$; /* X_i *is fixed to value* a_i */
    $\mathtt{CPropagate}(i + 1, (\mathbb{A}_{\mathcal{V}}, \mathbb{B}'_{\mathcal{V}}), m, \langle a_1, \ldots, a_{i-1}, a_i\rangle)$;
    if $i > m$ **then** BREAK;

end.

A Simple Modification

When is the algorithm correct?

A Simple Modification

When is the algorithm correct?

Thm.

Let $\mathbb{A}$ *be an* ℓ*-structure, and* $O \subseteq A$ *be a set of variables. Then, for every* r*-structure* $\mathbb{B}$, $\texttt{ComputeCertifiedSolutions}_{\mathrm{DM}}$ *computes WPD a subset of the solutions in* $\mathbb{A}^{\mathbb{B}}[O]$*, with a certificate for each of them. Moreover,*

- *If* $\texttt{ComputeCertifiedSolutions}_{\mathrm{DM}}$ *outputs "*DM *failure", then* $(\mathcal{H}_{\mathbb{A}}, \mathcal{H}_{\ell\text{-}\mathrm{DM}(\mathbb{A})})$ *does not have a tree projection;*
- *otherwise,* $\texttt{ComputeCertifiedSolutions}_{\mathrm{DM}}$ *computes WPD* $\mathbb{A}^{\mathbb{B}}[O]$.

Complexity Issues

Thm.

Assume FPT $\neq$ W[1]. *Let* **A** *be any bounded-arity recursively-enumerable class of* ℓ*-structures closed under taking minors. Then, the following are equivalent:*

(1) **A** *has bounded treewdith;*

(2) For every $\mathbb{A} \in \mathbf{A}$, *for every* r*-structure* $\mathbb{B}$, *and for every set of variables* $O \subseteq A$, *the* ECSP *instance* $(\mathbb{A}, \mathbb{B}, O)$ *is solvable WPD.*

Comparing The Results

Thm.

Assume $FPT \neq W[1]$. Let $\mathbf{A}$ be any class of ℓ-structures of bounded arity. Then, the following are equivalent:

(1) $\mathbf{A}$ *has bounded treewdith modulo homomorphic equivalence;*
(2) For every $\mathbb{A} \in \mathbf{A}$, for every r-structure $\mathbb{B}$, and for every set of variables $O \subseteq drv(\mathbb{A})$, the ECSP *instance $(\mathbb{A}, \mathbb{B}, O)$ is solvable WPD.*

Thm.

Assume FPT $\neq$ W[1]. *Let $\mathbf{A}$ be any bounded-arity recursively-enumerable class of ℓ-structures closed under taking minors. Then, the following are equivalent:*

(1) $\mathbf{A}$ *has bounded treewdith;*
(2) For every $\mathbb{A} \in \mathbf{A}$, for every r-structure $\mathbb{B}$, and for every set of variables $O \subseteq A$, the ECSP *instance $(\mathbb{A}, \mathbb{B}, O)$ is solvable WPD.*

Comparing The Results

Thm.

Assume $FPT \neq W[1]$. Let $\mathbf{A}$ be any class of ℓ-structures of bounded arity. Then, the following are equivalent:

(1) $\mathbf{A}$ has bounded treewidth modulo homomorphic equivalence;
(2) For every $\mathbb{A} \in \mathbf{A}$, for every r-structure $\mathbb{B}$, and for every set of variables $O \subseteq drv(\mathbb{A})$, the ECSP *instance $(\mathbb{A}, \mathbb{B}, O)$ is solvable WPD.*

Thm.

Assume FPT $\neq$ W[1]. *Let $\mathbf{A}$ be any bounded-arity recursively-enumerable class of ℓ-structures closed under taking minors. Then, the following are equivalent:*

(1) $\mathbf{A}$ has bounded treewidth;
(2) For every $\mathbb{A} \in \mathbf{A}$, for every r-structure $\mathbb{B}$, and for every set of variables $O \subseteq A$, the ECSP *instance $(\mathbb{A}, \mathbb{B}, O)$ is solvable WPD.*

Thank you!