Efficient Model Transformations For Novices

An overview of known techniques

Sten Vercammen University of Antwerp December 11, 2015

Graph Pattern Matching

- Search Plans

defines traversal order

based on heuristic

- Constraint Satisfaction Problems (CSP)

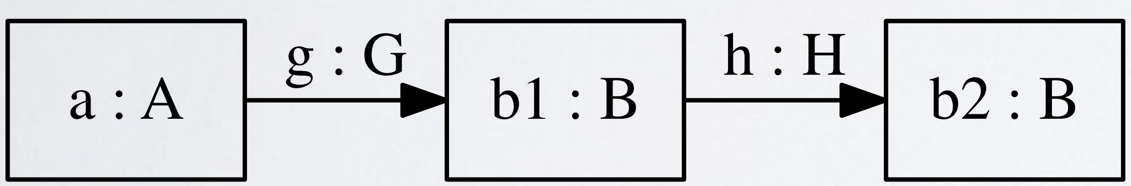
$\langle X, D, C \rangle$

set of objects

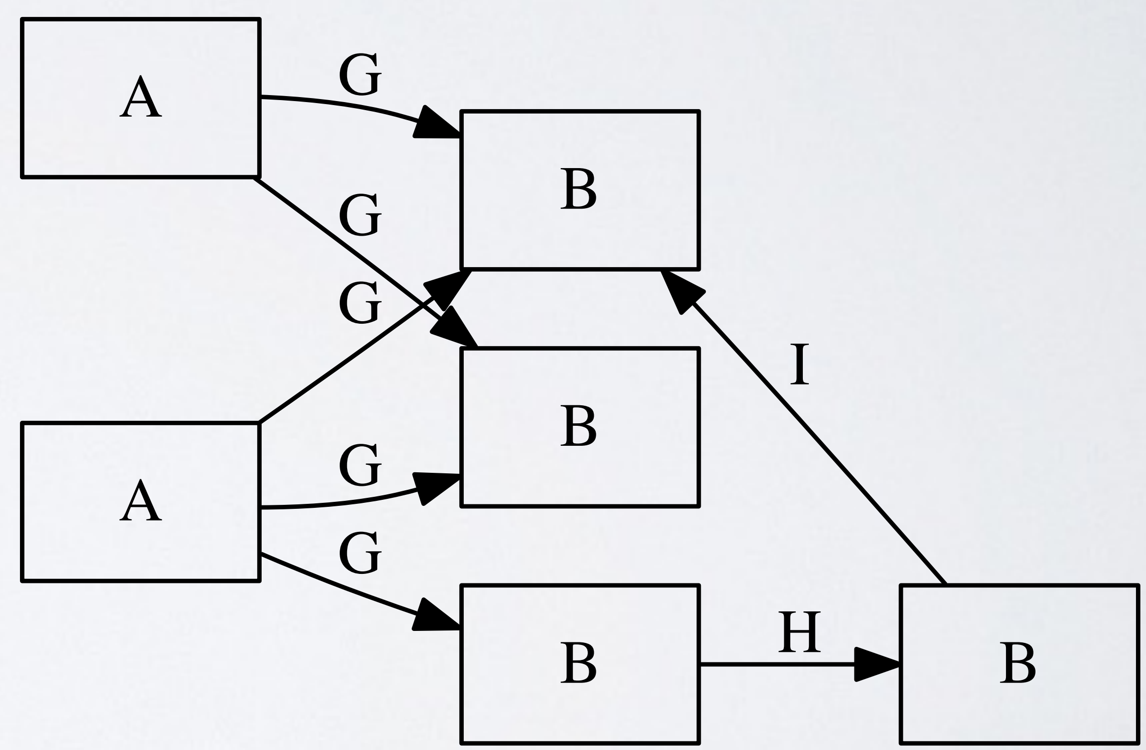
based on constraints

Search Plan*

- Pattern Graph P



- Host Graph H



*: G.V. Batz, M. Kroll, R. Geiß, A first experimental evaluation of search plan driven graph pattern matching, in: Applications of Graph Transformations with Industrial Relevance, Springer, 2008, pp. 471–486.

Primitive Matching Operations

• **lkp**(*x*)	binds pattern element *x* to host graph element
• **in**(*v*, e)	binds incoming pattern edge e to incoming edge on already bound *v*
• **out**(*v*, e)	binds outgoing pattern edge e to outgoing edge on already bound *v*
• **src**(e)	binds source of already bound edge e
• **tgt**(e)	binds target of already bound edge e

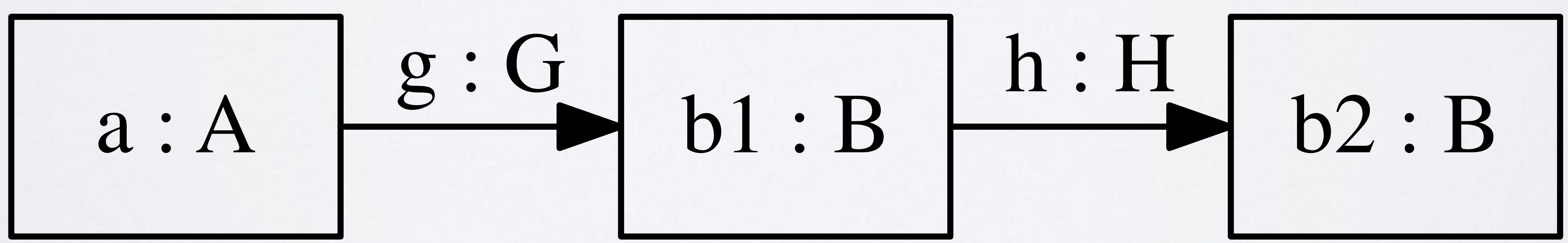
Valid Search Plan

$P = \langle o_1, \ldots, o_2 \rangle$ is valid:

- If every element in pattern graph is bound **exactly** once
- If o_i requires a bound element x, x must be bound in one of $o_1, \ldots, o_{i-1}$

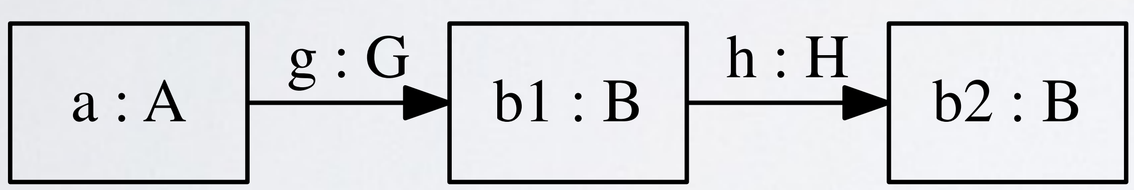
Example Search Plan

- $P_1 = \langle\ \mathbf{out}(a, g), \mathbf{lkp}(b1), \mathbf{tgt}(g), \mathbf{lkp}(h)\ \rangle$
- $P_2 = \langle\ \mathbf{lkp}(b_1), \mathbf{out}(b_1, h), \mathbf{in}(b_1, h), \mathbf{tgt}(h), \mathbf{src}(g)\ \rangle$
- $P_3 = \langle\ \mathbf{lkp}(h), \mathbf{tgt}(h), \mathbf{src}(h), \mathbf{in}(b_1, g), \mathbf{src}(g)\ \rangle$

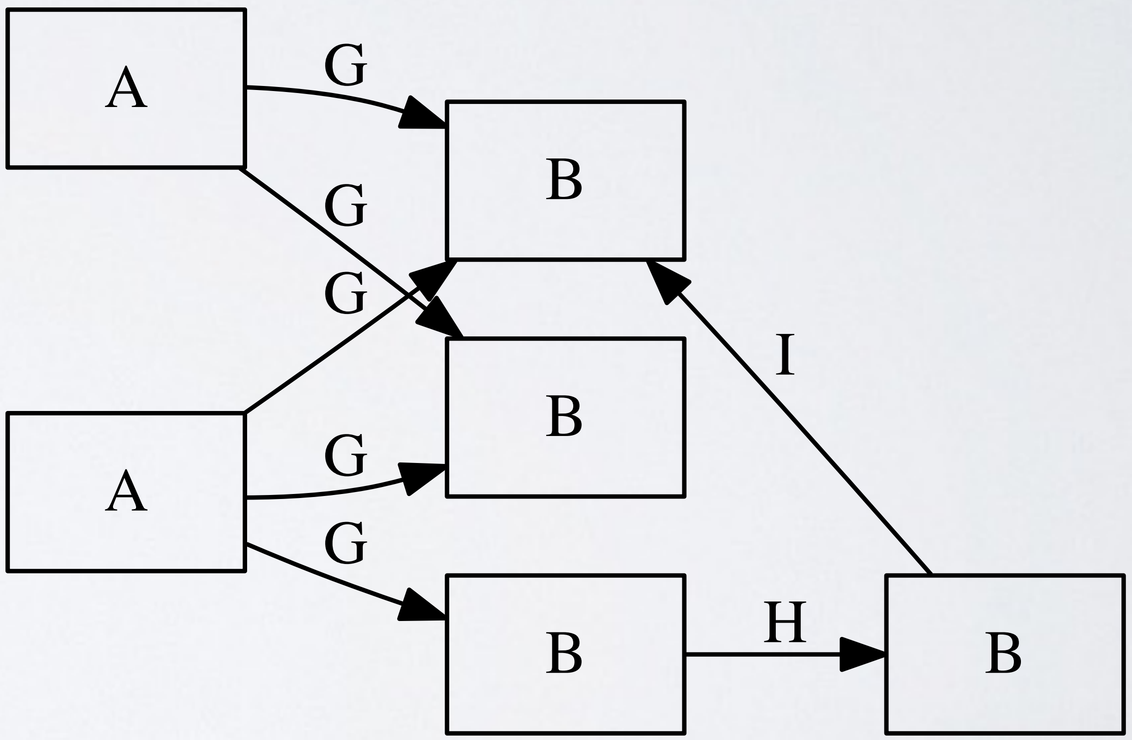


$P_2 = \langle\ \mathbf{lkp}(b_1), \mathbf{out}(b_1, h), \mathbf{in}(b_1, h), \mathbf{tgt}(h), \mathbf{src}(g)\ \rangle$

- ✦ Pattern Graph P

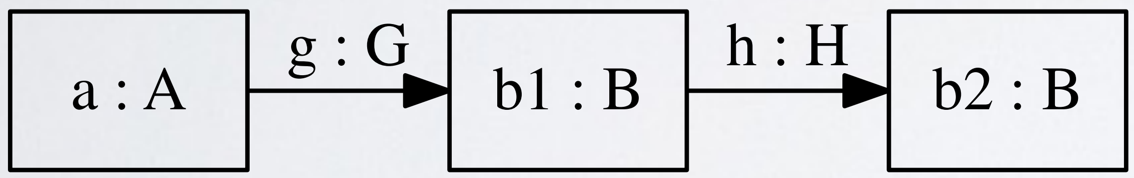


- ✦ Host Graph H

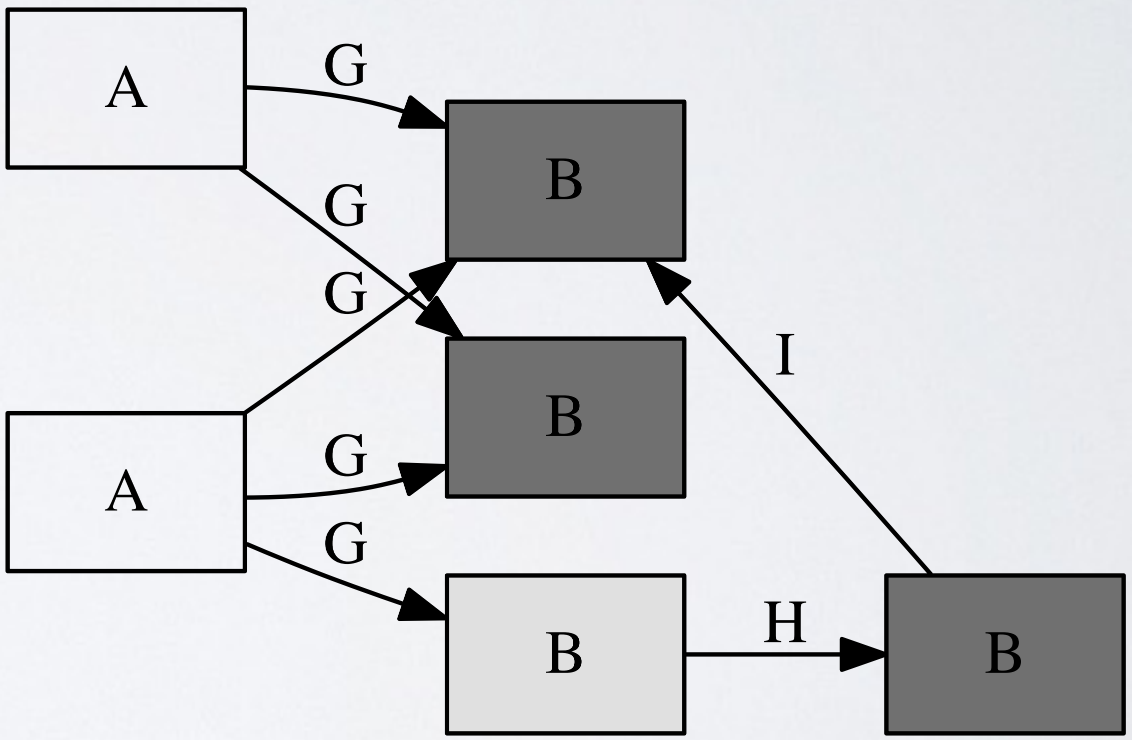


$$P_2 = \langle\ \mathbf{lkp}(b_1), \mathbf{out}(b_1, h), \mathbf{in}(b_1, h), \mathbf{tgt}(h), \mathbf{src}(g)\ \rangle$$

- Pattern Graph P

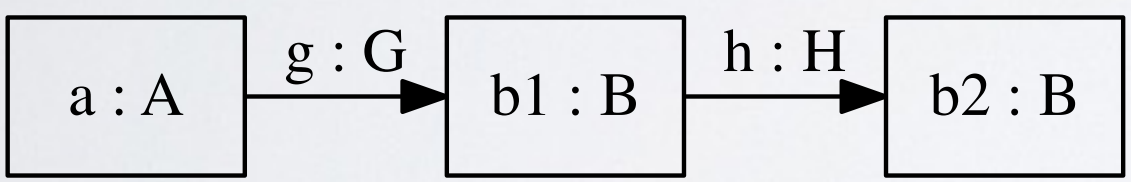


- Host Graph H

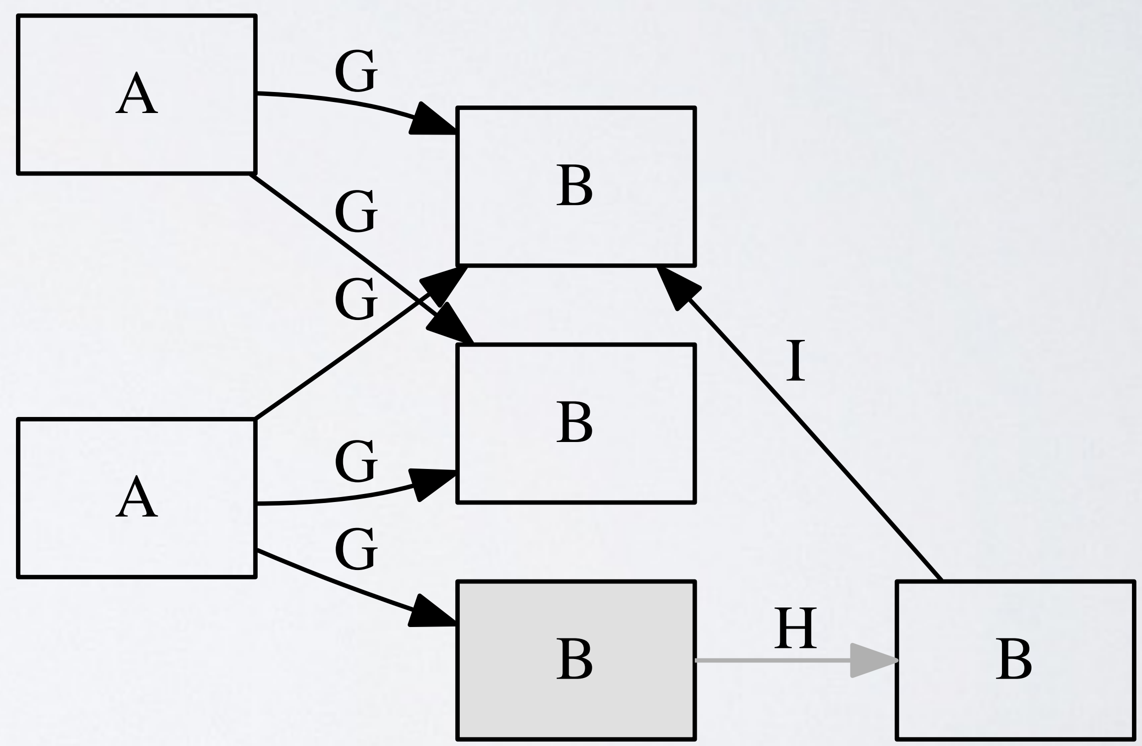


$P_2 = \langle\ \mathbf{lkp}(b_1), \mathbf{out}(b_1, h), \mathbf{in}(b_1, h), \mathbf{tgt}(h), \mathbf{src}(g)\ \rangle$

- Pattern Graph P

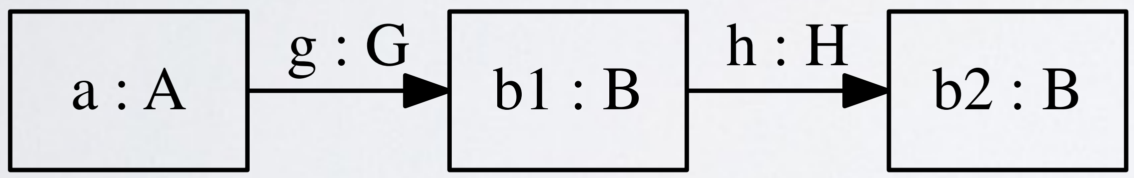


- Host Graph H

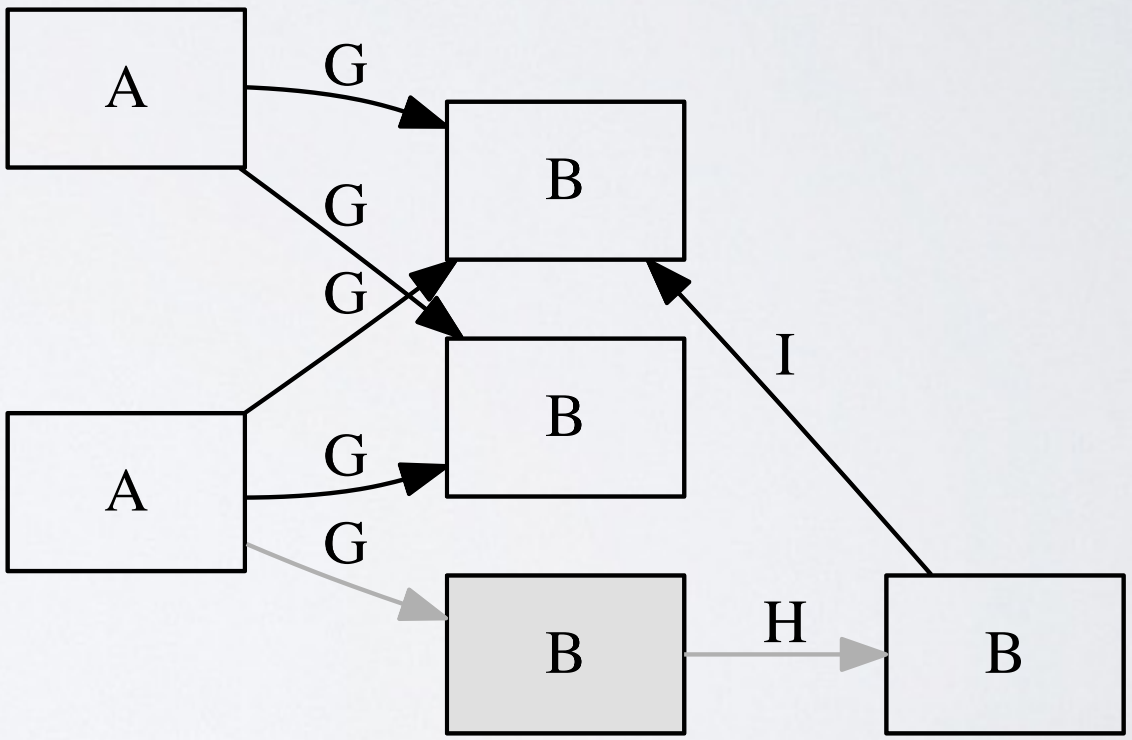


$P_2 = \langle\ \mathbf{lkp}(b_1), \mathbf{out}(b_1, h), \mathbf{in}(b_1, h), \mathbf{tgt}(h), \mathbf{src}(g)\ \rangle$

✦ Pattern Graph P

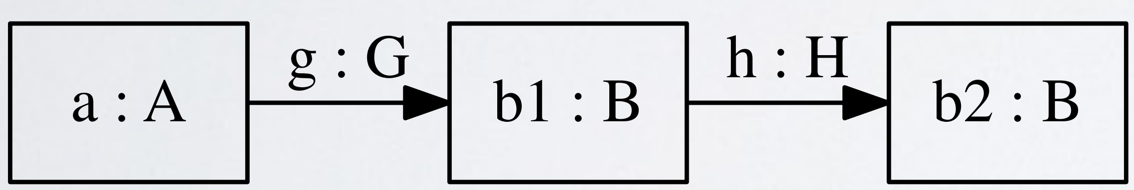


✦ Host Graph H

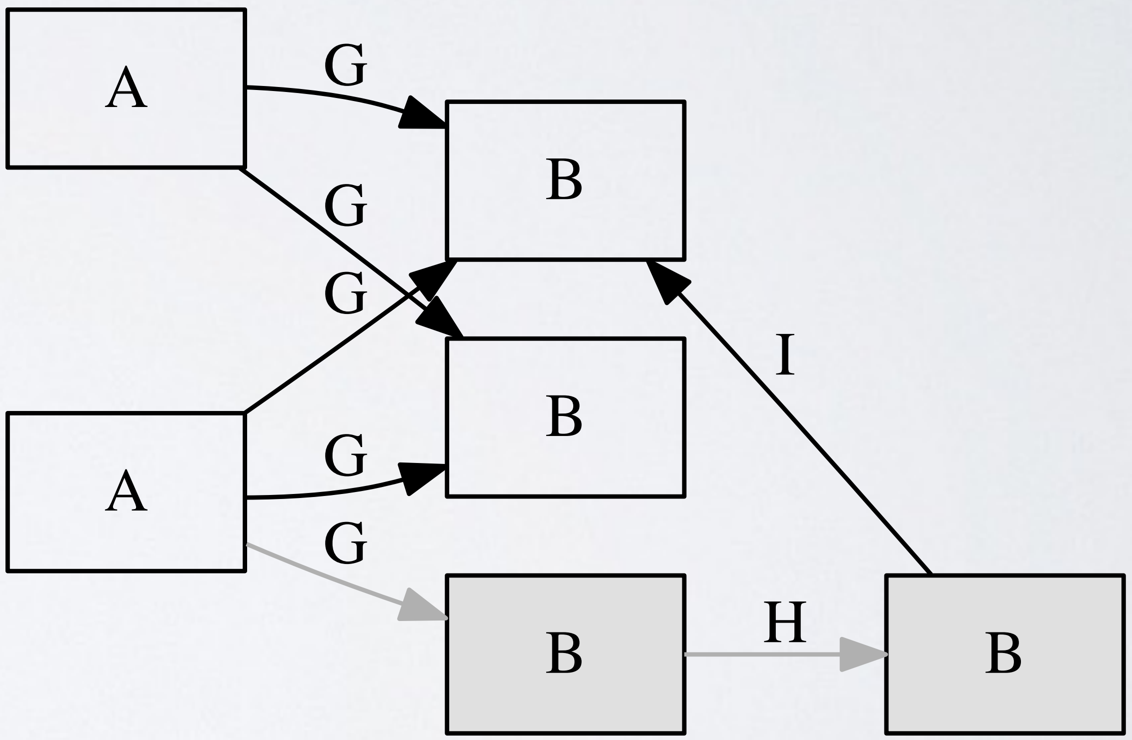


$P_2 = \langle\ \mathbf{lkp}(b_1), \mathbf{out}(b_1, h), \mathbf{in}(b_1, h), \mathbf{tgt}(h), \mathbf{src}(g)\ \rangle$

✦ Pattern Graph P

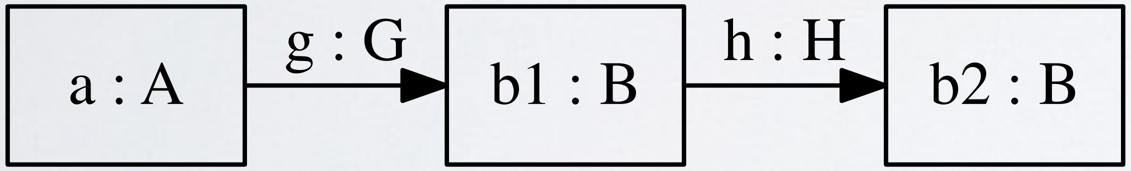


✦ Host Graph H

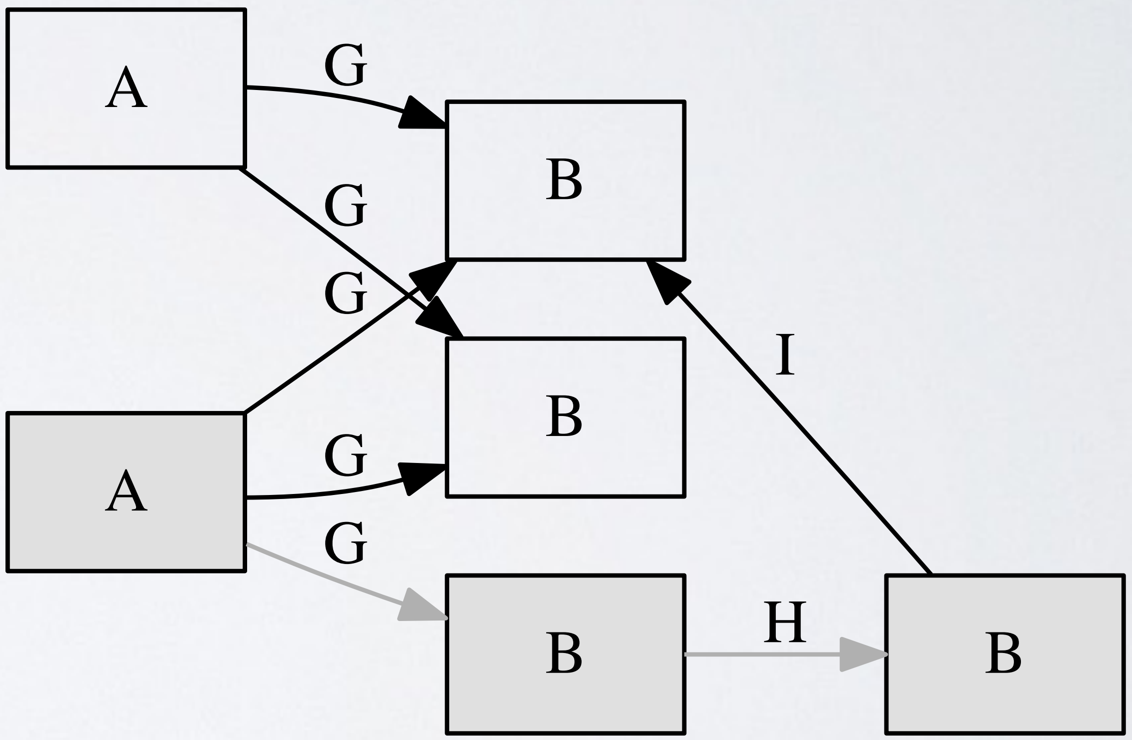


$P_2 = \langle\ \mathbf{lkp}(b_1), \mathbf{out}(b_1, h), \mathbf{in}(b_1, h), \mathbf{tgt}(h), \mathbf{src}(g)\ \rangle$

✦ Pattern Graph P



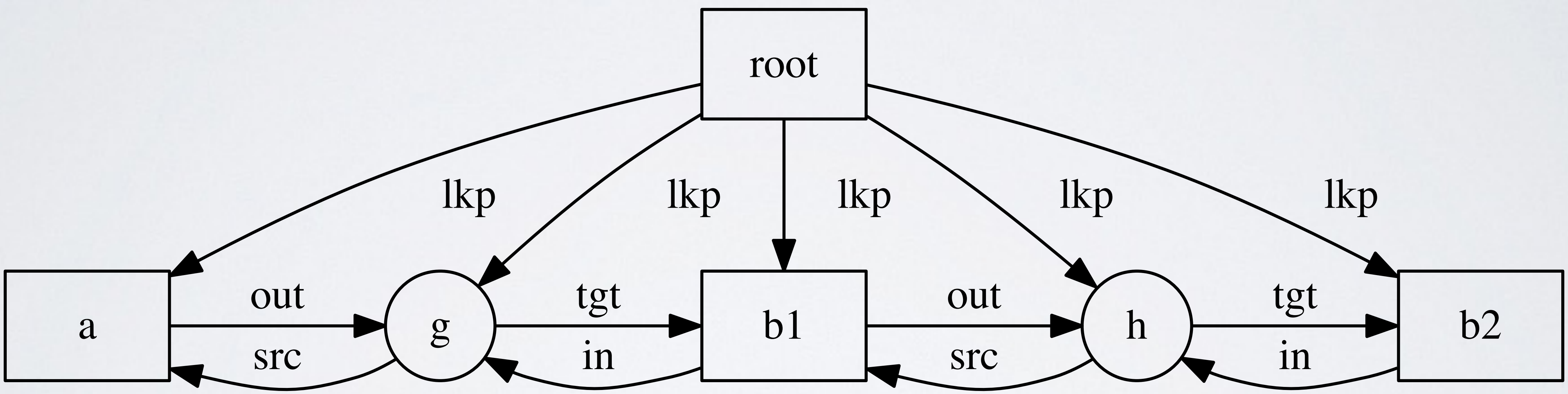
✦ Host Graph H



Cost Model

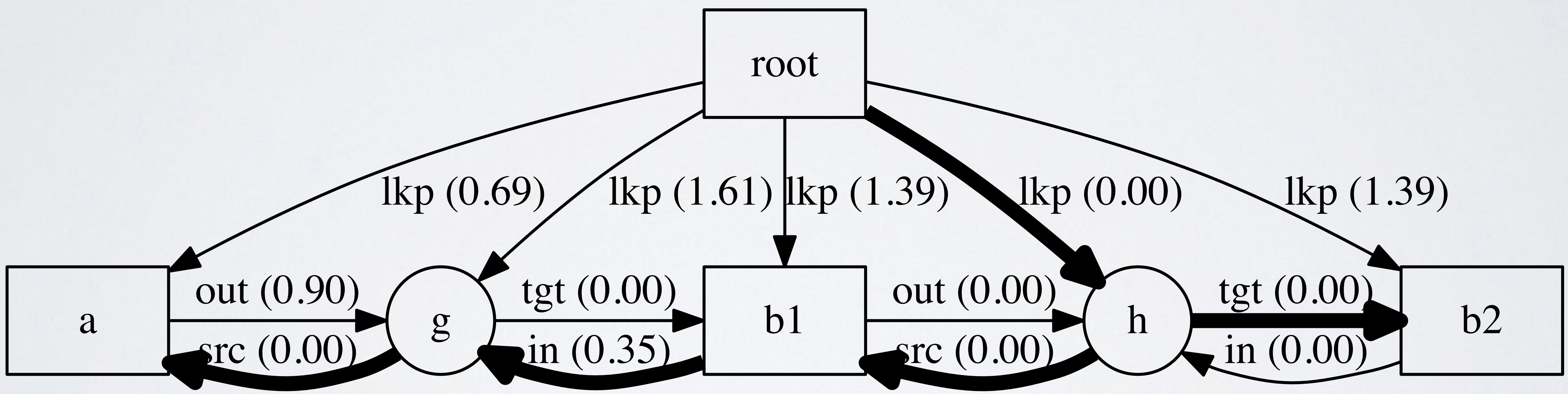
• **lkp**(*x*)	|typeof(*x*) in *H*|
• **in**(*v*, e), **out**(*v*, e)	AVG(possibilities)
• **src**(e), **tgt**(e)	1
• $P = \langle o_1, \ldots, o_2 \rangle$	$\sum_{i=1}^{k} \prod_{j=1}^{i} c_j$

Plan Graph



Plan Graph With Cost

$$c(P) = c_1 + c_1 c_2 + \ldots + c_1 \ldots c_k$$



MDST with Edmonds* algorithm

*: J. Edmonds, Optimum branchings, Journal of Research of the National Bureau of Standards B 71 (4) (1967) 233–240.

Alternative Techniques

- Optimisation problem, different heuristic
- Heuristic with type info, cardinality constraints,…
- Adaptive search plans[1]
- Indexing on type and/or by storing reverse associations, caching, pivoting and overlapped pattern matching[2]

1: G. Varr´o, K. Friedl, D. Varr´o, Adaptive graph pattern matching for model transformations using model-sensitive search plans, Electronic Notes in Theoretical Computer Science 152 (2006) 191–205.
2: C. Aˆ. G. Gomes, A framework for efficient model transformations.

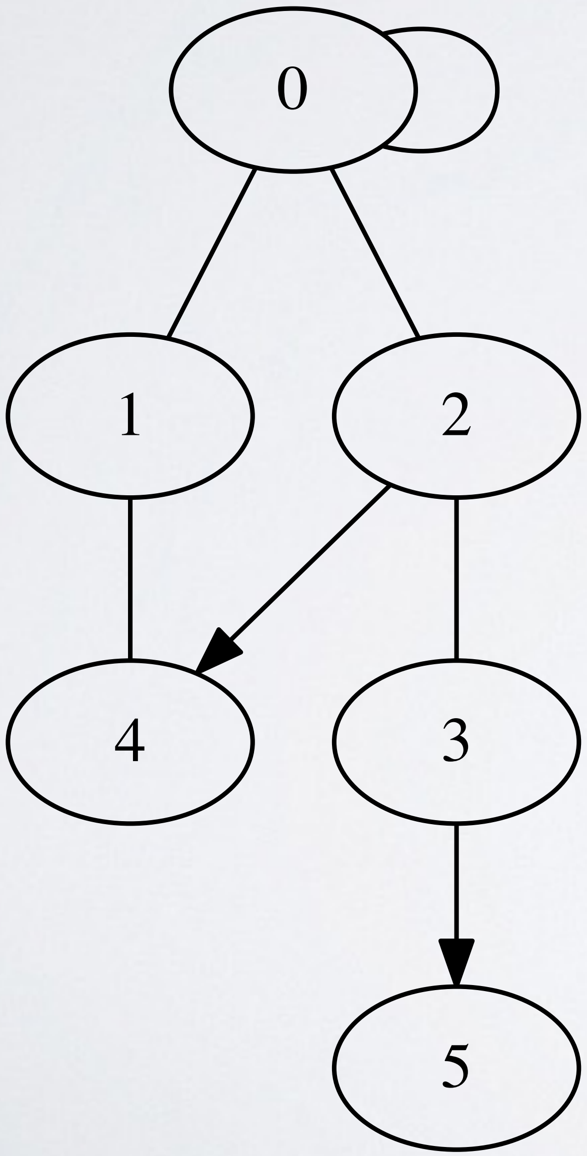
Constraint Satisfaction Problems

Definition

- $\langle X, D, C \rangle$
- X = elements of pattern graph: variables
- D = set of domains, for each variable
- C = links and attributes

Adjacency Matrices

✦ Example Graph



✦ Adjacency Matrix I

x	0	1	2	3	4	5
0	2	1	1	0	0	0
1	1	0	0	0	1	0
2	1	0	0	1	1	0
3	0	0	1	0	0	1
4	0	1	0	0	0	0
5	0	0	0	0	0	0

Ullmann*

$$\mathbf{M}^{*}[v,w] = \begin{cases} 1, & \text{if } \deg(v) \leq \deg(w) \text{ for } v \in V_P \text{ and } w \in V_H \\ 0, & \text{otherwize.} \end{cases}$$

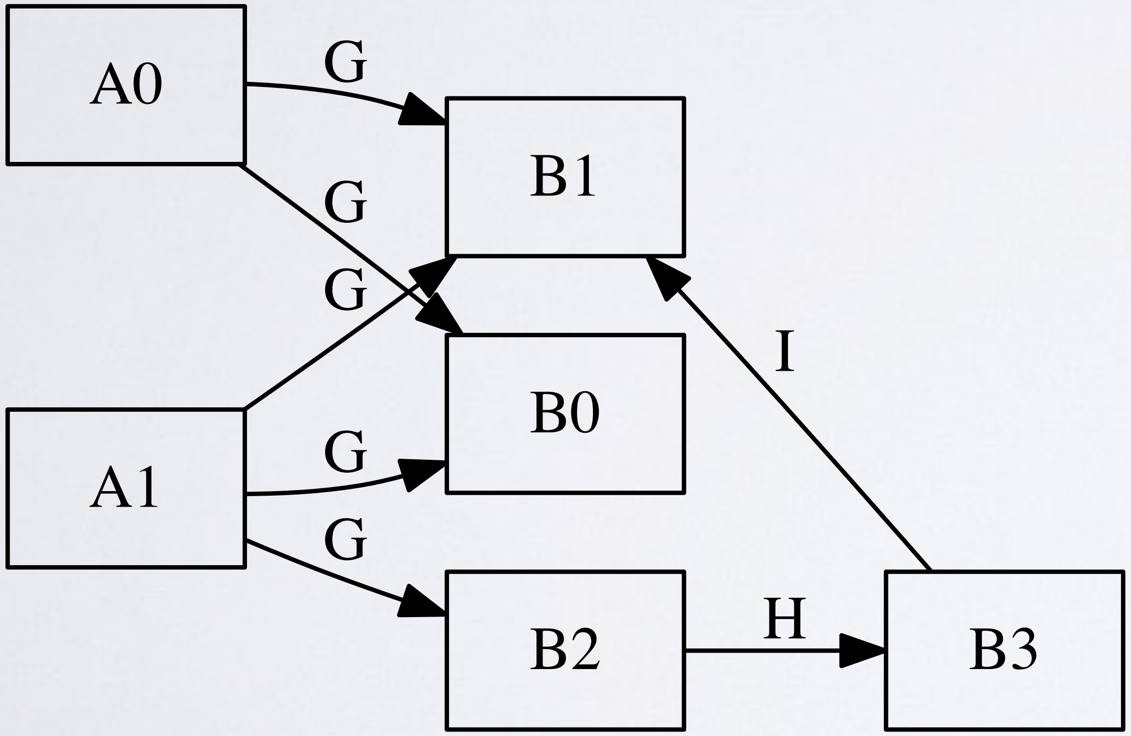
$\deg : V \rightarrow \mathbb{N}^{+}$ mapping vertex to its degree

isomorph iff $\boldsymbol{M(MH)^T = P}$

*: J. R. Ullmann, An algorithm for subgraph isomorphism, Journal of the ACM (JACM) 23 (1) (1976) 31–42.

Ullmann Example

- Relabelled Host Graph

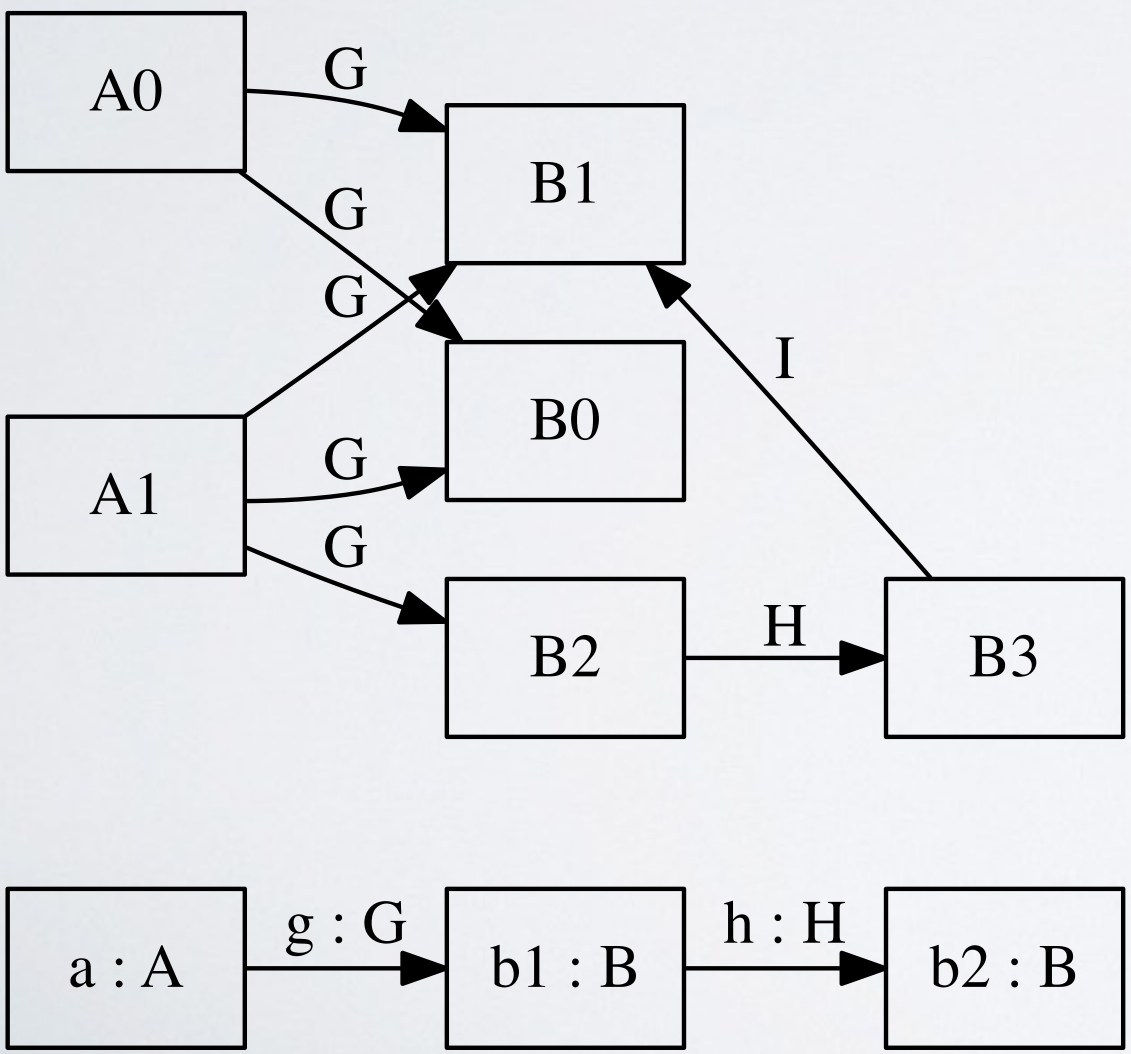


- Adjacency Matrix *H*

x	A0	A1	B0	B1	B2	B3
A0	0	0	1	1	0	0
A1	0	0	1	1	1	0
B0	0	0	0	0	0	0
B1	0	0	0	0	0	0
B2	0	0	0	0	0	1
B3	0	0	0	1	0	0

Ullmann Example

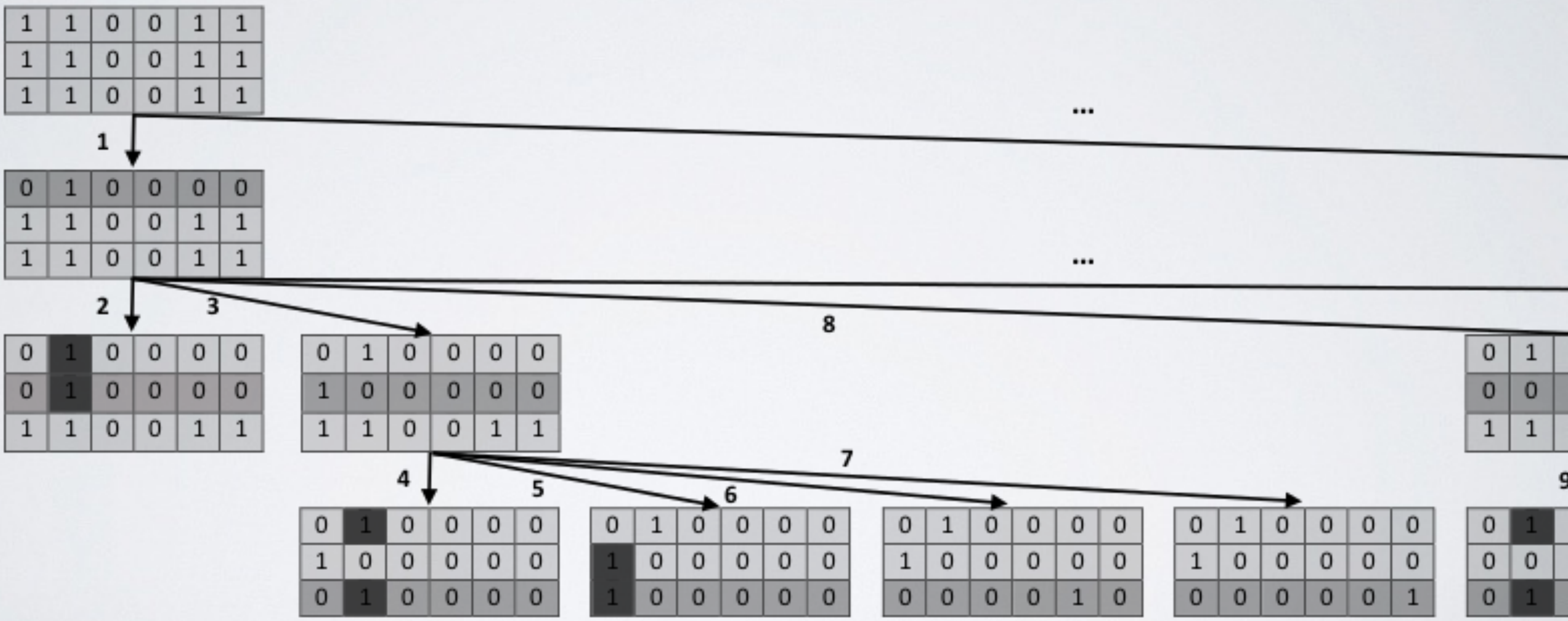
✦ Host and Pattern Graph



✦ $|V_P| \times |V_H|$ matrix $\mathbf{M}^*$

x	A0	A1	B0	B1	B2	B3
a	1	1	0	0	1	1
b1	1	1	0	0	1	1
b2	1	1	0	0	1	1

Ullmann Example

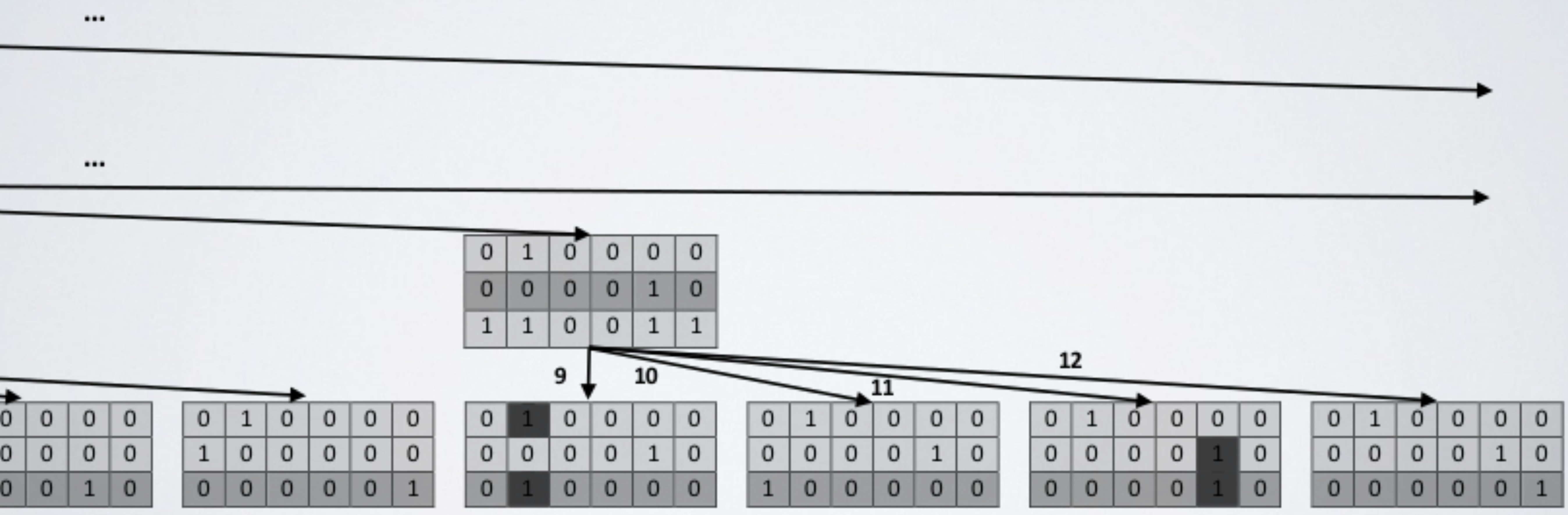


Ullmann Example

$\mathbf{P}$ is isomorphic if and only if $\forall i,j, \mathbf{P}[i,j]=1 : \mathbf{M}(\mathbf{MH})^T[j,i]=1$.

$$
\begin{aligned}
\mathbf{M}(\mathbf{MH})^T &= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \times \left(\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \right)^T \\
&= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \times \left(\begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \right)^T \\
&= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}
\end{aligned}
$$

Ullmann Example



Ullmann Example

$\mathbf{P}$ is isomorphic if and only if $\forall i,j, \mathbf{P}[i,j]=1 : \mathbf{M}(\mathbf{MH})^T[j,i]=1$.

$$
\begin{aligned}
\mathbf{M}(\mathbf{MH})^T &= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \times \left(\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \right)^T \\
&= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \times \left(\begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \right)^T \\
&= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}
\end{aligned}
$$

VF2*

- Simular to Ullmann
- $G = (V_G, E_G)$, $H = (V_H, E_H)$
- $M : V_H \rightarrow V_G$: isomorphic vertex mapping
- $M(c)$ holds set of current matches
- M_H, M_G represent vertices from H, G in $M(s)$

Note: G represents host graph, H the pattern

*: L. P. Cordella, P. Foggia, C. Sansone, M. Vento, A (sub) graph isomorphism algorithm for matching large graphs, Pattern Analysis and Machine Intelligence, IEEE Transactions on 26 (10) (2004) 1367– 1372.

VF2 Notations

- $N^{in}{}_{H}(s)$ vertices adjacent to $M_H(s)$ angling incoming edges
- $N^{out}{}_{H}(s)$ idem for outgoing edges
- $N_H(s) = N^{in}{}_{H}(s) + N^{out}{}_{H}(s)$
- $\tilde{V}_H = V_H - M_H(s) - N_H(s)$
- candidate $p = (v_p, v_p) \in P(s)$
- priority $N^{out}{}_{H}(s)$, $N^{in}{}_{H}(s)$, $\tilde{V}_H$

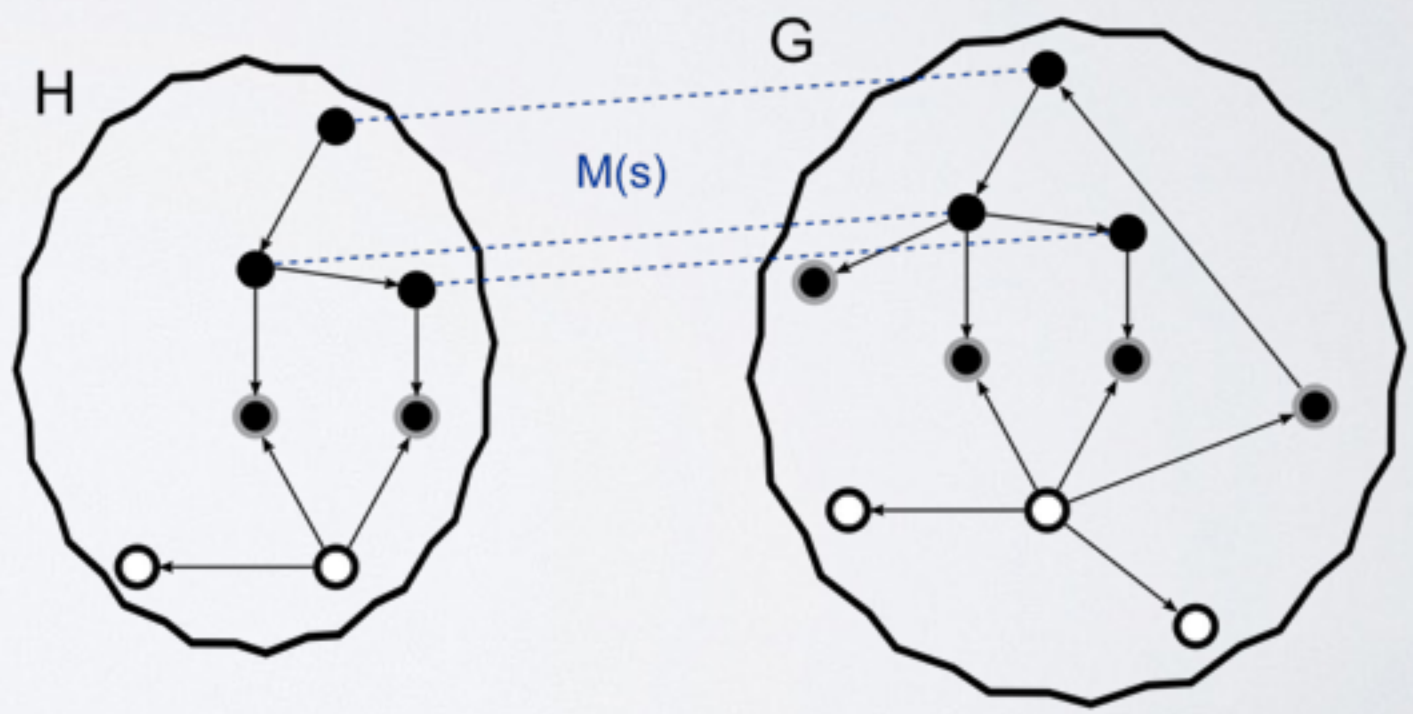


Image from: E. Syriani, A multi-paradigm foundation for model transformation language engineering, Ph.D. thesis, McGill University (2011)

VF2 Algorithm

- match 1 vertex
- extend match with candidate p if (else backtrack)
- run test on $s' = s \cup p$ (in order)
  - M(s') valid
  - ext. edges between $M_H(s')$ and $N_H(s') \leq M_G(s')$ and $N_G(s')$
  - ext. edges between $N_H(s')$ and $\tilde{V}_H(s') \leq N_G(s')$ and $\tilde{V}_{H\,G}(s')$

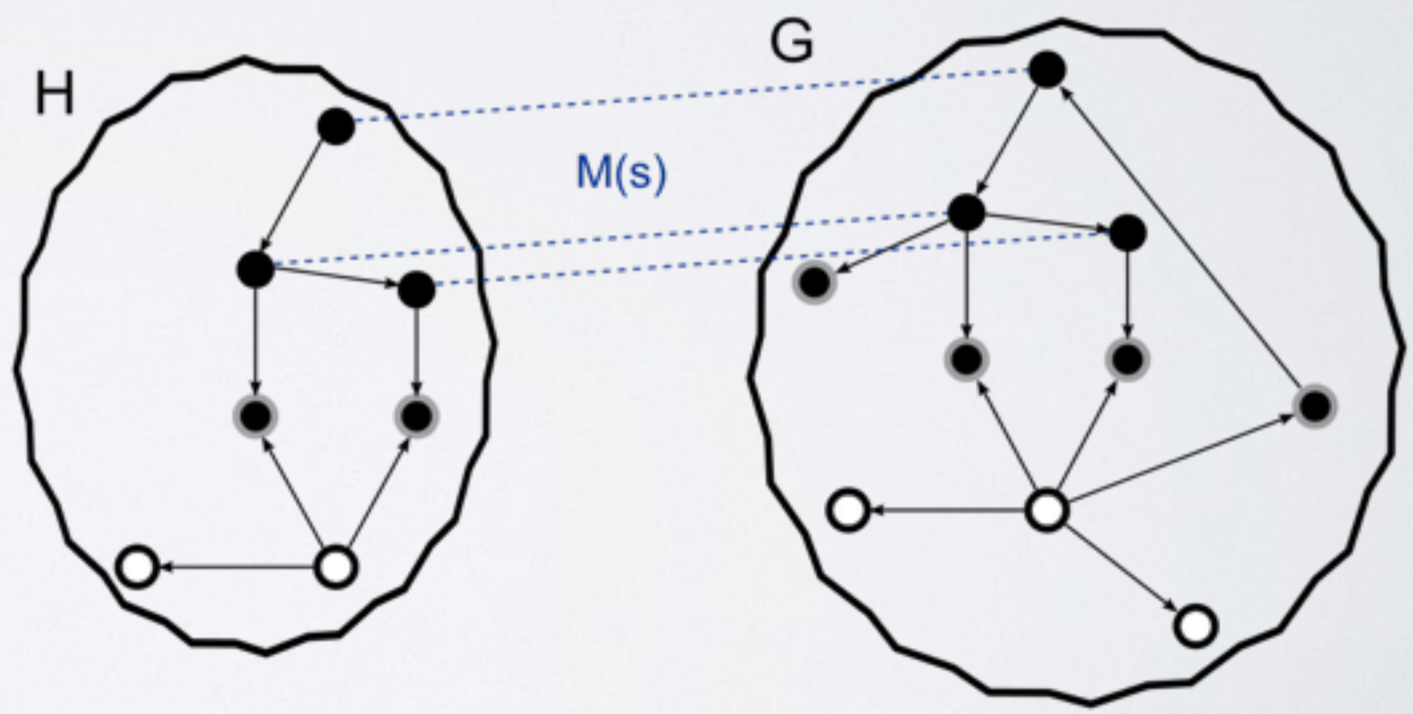


Image from: E. Syriani, A multi-paradigm foundation for model transformation language engineering, Ph.D. thesis, McGill University (2011)

Look-back on Ullmann and VF2

- Main difference in backtracking step:
  - Ullmann only compares pairs of adjacent vertices
  - VF2 compares verity with neighbourhood
  - Ullmann's $\mathbf{M}^*$ matrix verifies semantic compatibility between vertices in match
  - VF2 feasibility test ensures correct match

Ullmann vs VF2*

- VF2 best case: $O(N^2)$, for $N = |V_H|+|V_G|$
- VF2 worst case: $O(N!N)$
- VF2 in both cases order of lineair magnitude faster
- VF2 lineair spatial complexity vs cubic
- Combination possible for lower time complexity

*: P. Foggia, C. Sansone, M. Vento, A database of graphs for isomorphism and sub-graph isomorphism benchmarking, in: Proc. of the 3rd IAPR TC-15 International Workshop on Graph-based Representations, 2001, pp. 176–187.