

# **The Optimal Jacobian Accumulation Problem**

Andrew Lyons

3/14/07

# Jacobians by Automatic Differentiation (AD)

code for  $F(\mathbf{x}_0) : \mathbb{R}^n \rightarrow \mathbb{R}^m, \quad \mathbf{x} \mapsto \mathbf{y}$

$\downarrow$  AD

*augmented* code for  $F(\mathbf{x}_0), F'(\mathbf{x}_0)$

Jacobian  $J \equiv F'(\mathbf{x}) = \left( \frac{\partial y_j}{\partial x_i} \right)_{i=1,\dots,n}^{j=1,\dots,m} \in \mathbb{R}^{m \times n}$

# Jacobians by Automatic Differentiation (AD)

$$[ y_1 = x_1 * x_2 * \sin(x_1 * x_2), \quad y_2 = \cos(\sin(x_1 * x_2)) ]^T$$

$$(v_{-1} = x_1, \ v_0 = x_2)$$

$$v_1 = v_{-1} * v_0$$

$$v_2 = \sin(v_1)$$

$$v_3 = v_1 * v_2$$

$$v_4 = \cos(v_2)$$

$$(y_1 = v_3, \ y_2 = v_4)$$

Generally

$$v_j = \varphi_j(v_i)_{i \prec j}$$

$$c_{ji} = \frac{\partial \varphi_j}{\partial v_i} \in \mathbb{R}$$

# Jacobians by Automatic Differentiation (AD)

$$[ y_1 = x_1 * x_2 * \sin(x_1 * x_2), \quad y_2 = \cos(\sin(x_1 * x_2)) ]^T$$

$$(v_{-1} = x_1, \ v_0 = x_2)$$

$$v_1 = v_{-1} * v_0 \quad c_{1-1} = v_0 \quad c_{10} = v_{-1}$$

$$v_2 = \sin(v_1) \quad c_{21} = \cos(v_2)$$

$$v_3 = v_1 * v_2 \quad c_{31} = v_2 \quad c_{32} = v_1$$

$$v_4 = \cos(v_2) \quad c_{42} = -\sin(v_2)$$

$$(y_1 = v_3, \ y_2 = v_4)$$

Generally

$$v_j = \varphi_j(v_i)_{i \prec j}$$

$$c_{ji} = \frac{\partial \varphi_j}{\partial v_i} \in \mathbb{R}$$

# Linearized Computational Graph

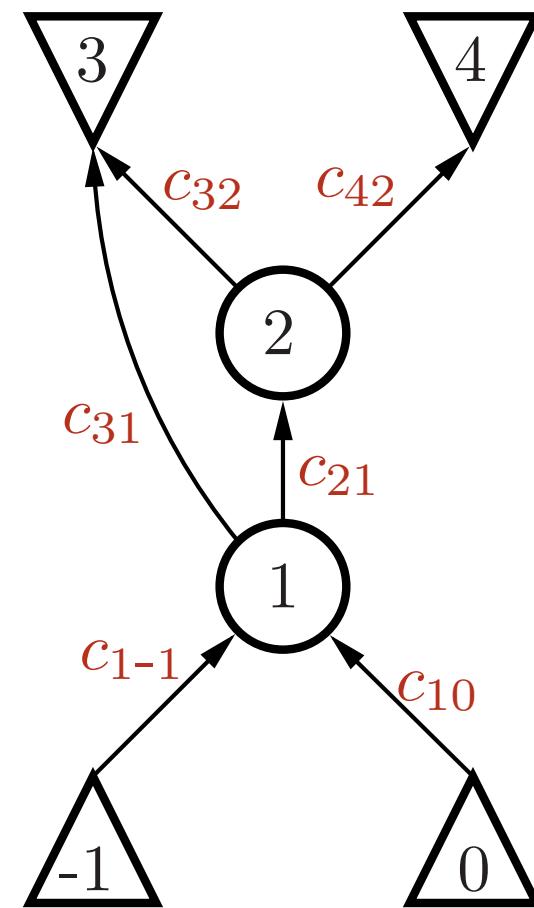
$v_i \prec v_j \Leftrightarrow v_i$  is an argument of  $\varphi_j$

$$v_1 = \varphi_1(v_{-1}, v_0)$$

$$v_2 = \varphi_2(v_1)$$

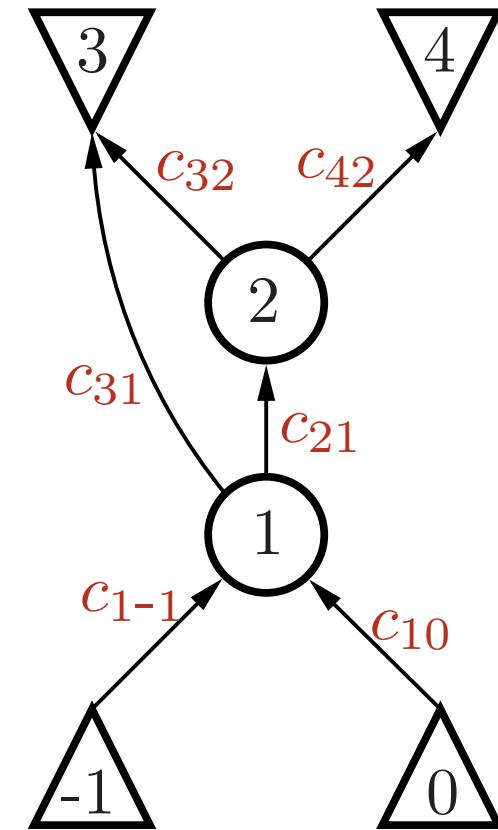
$$v_3 = \varphi_3(v_1, v_2)$$

$$v_4 = \varphi_4(v_2)$$



## Baur's Formula (Baur, Strassen 1983)

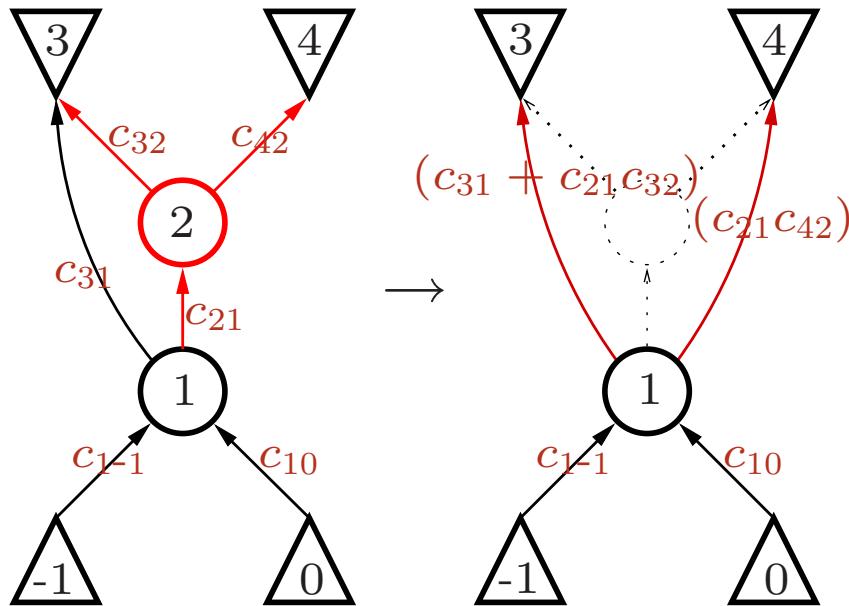
$$J_{y_\ell, x_k} = \sum_{[x_k \rightarrow y_\ell] \in \mathbf{G}} \prod_{c_{ji} \in [x_k \rightarrow y_\ell]} c_{ji}$$



$$J = \begin{pmatrix} c_{1-1}c_{31} + c_{1-1}c_{21}c_{32} & c_{10}c_{31} + c_{10}c_{21}c_{32} \\ c_{1-1}c_{21}c_{42} & c_{10}c_{21}c_{42} \end{pmatrix}$$



## Vertex Elimination (Griewank, Reese 1991)



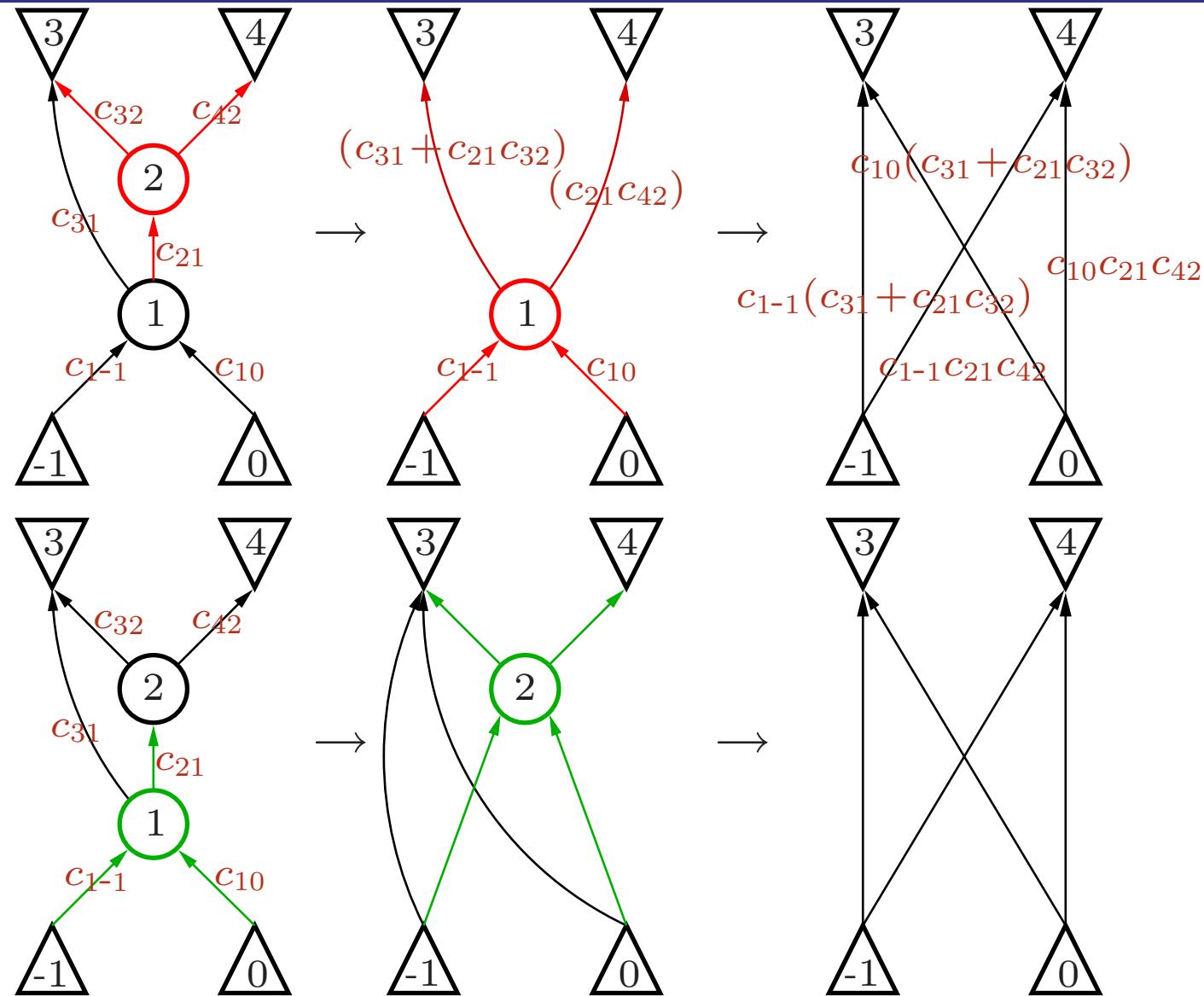
$$\begin{aligned} dv_1 &= c_{31} + c_{32} * c_{21} \\ dv_2 &= c_{21} * c_{42} \end{aligned}$$

$$\begin{pmatrix} c_{1-1}c_{31} + c_{1-1}c_{21}c_{32} & c_{10}c_{31} + c_{10}c_{21}c_{32} \\ c_{1-1}c_{21}c_{42} & c_{10}c_{21}c_{42} \end{pmatrix}$$



$$\begin{pmatrix} c_{1-1}(c_{31} + c_{21}c_{32}) & c_{10}(c_{31} + c_{21}c_{32}) \\ c_{1-1}(c_{21}c_{42}) & c_{10}(c_{21}c_{42}) \end{pmatrix}$$

## Vertex Elimination (Griewank, Reese 1991)



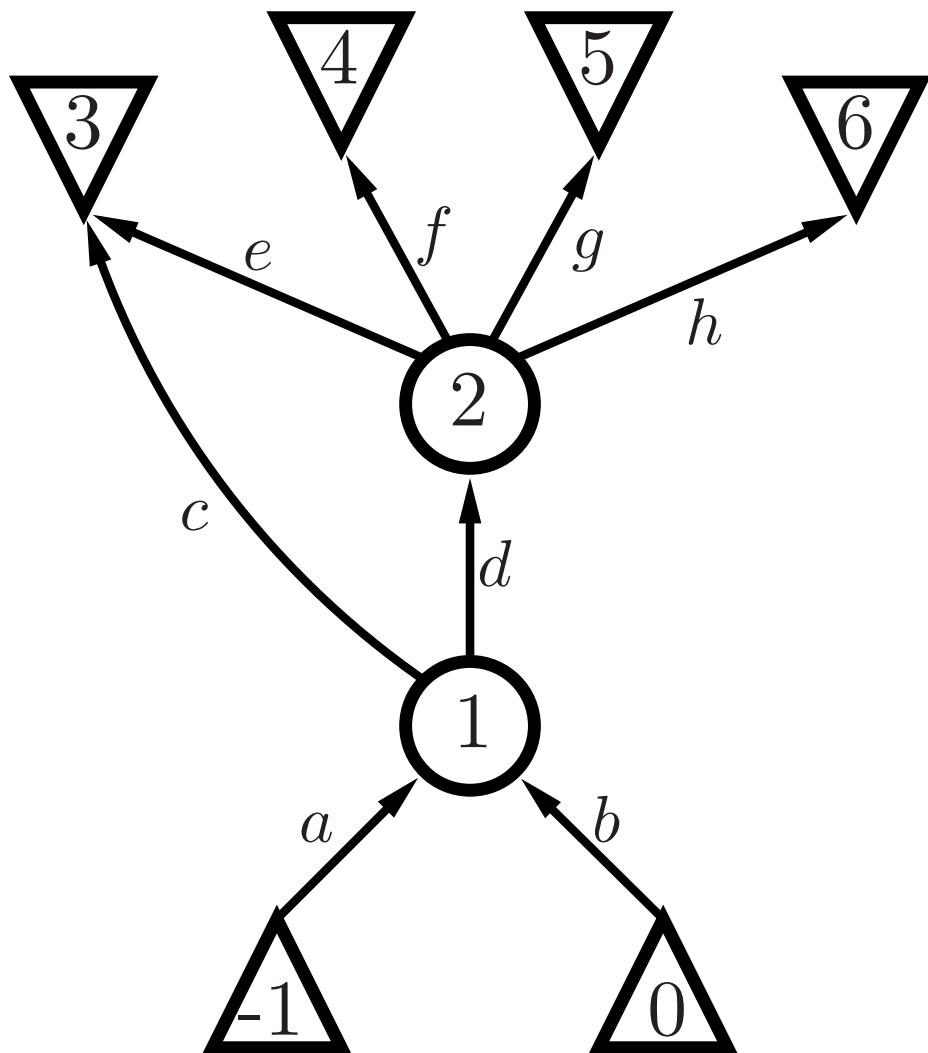
## Vertex Elimination (Griewank, Reese 1991)

Similar to symbolic Gaussian elimination  
(for unsymmetric LU factorization)

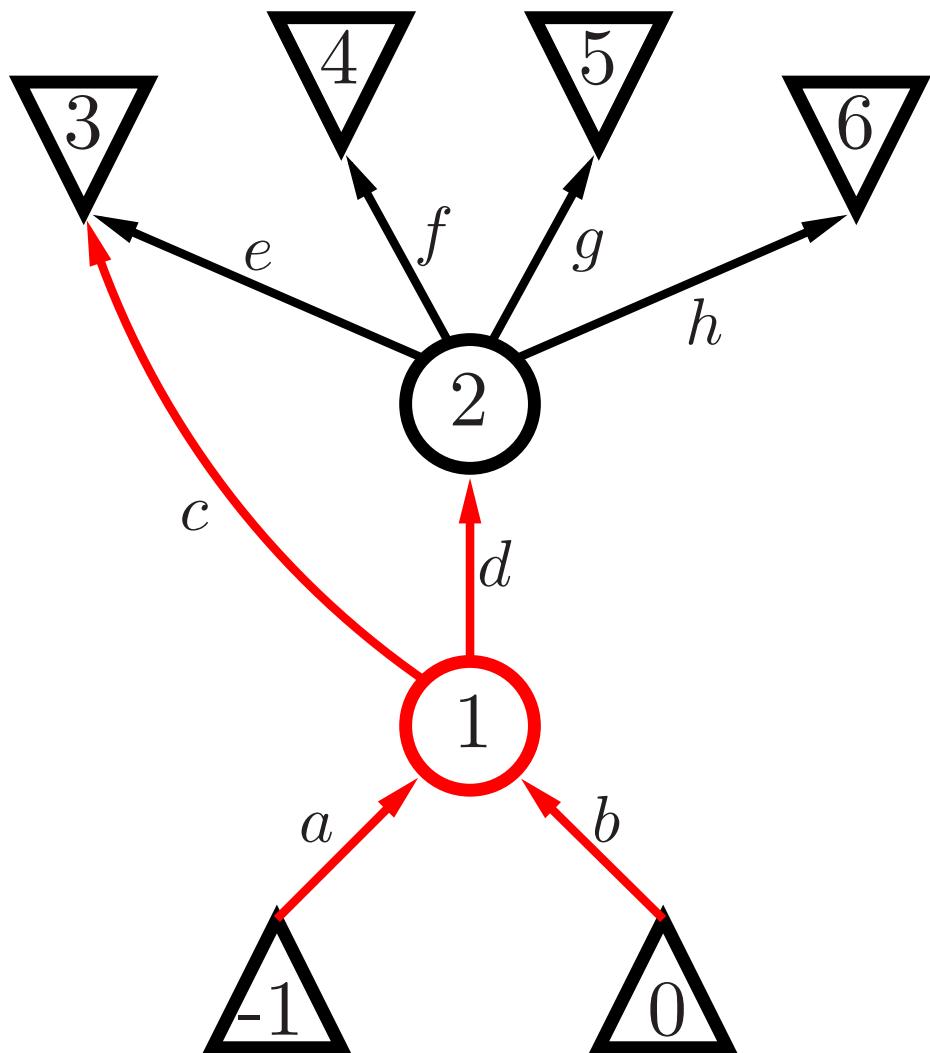
. . . But not completely:

- We have acyclic graphs
- We are concerned with *multiplications* instead of *fill-in*  
(which is NP-complete on DAGs)

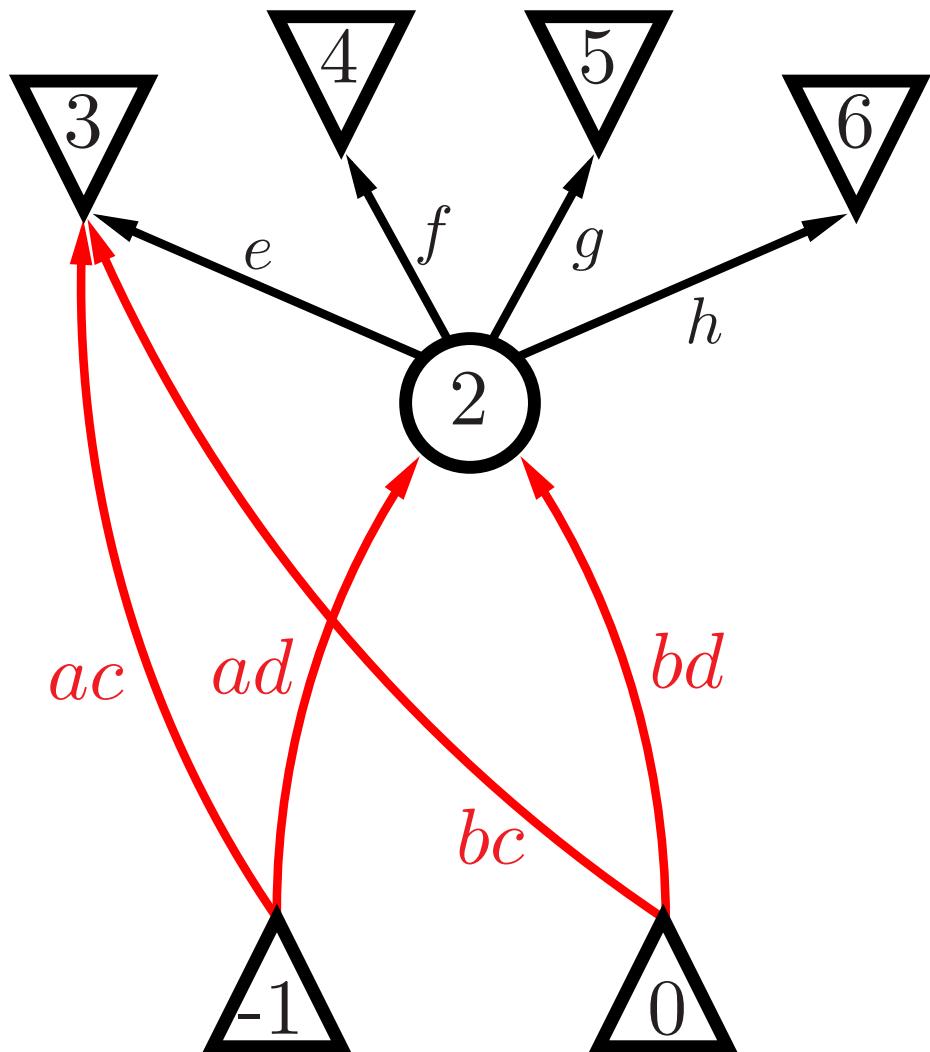
## Vertex Elimination



## Vertex Elimination

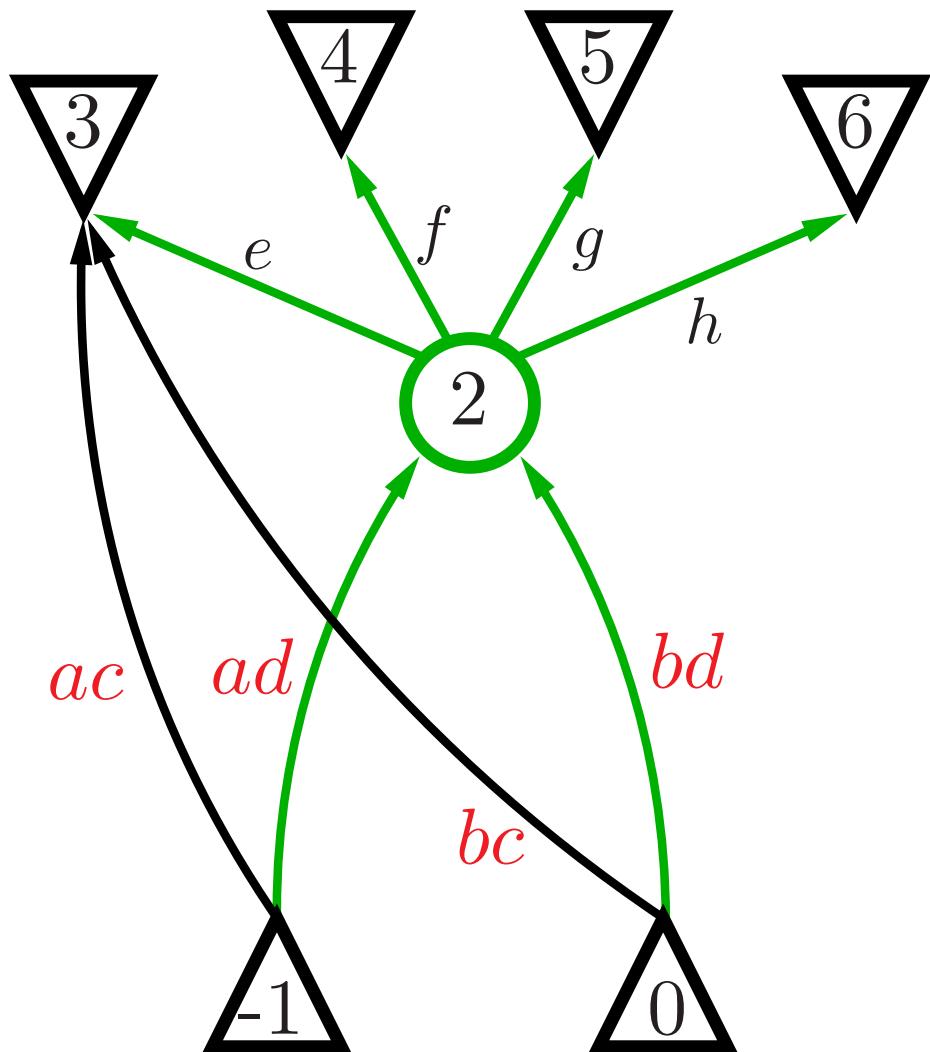


## Vertex Elimination



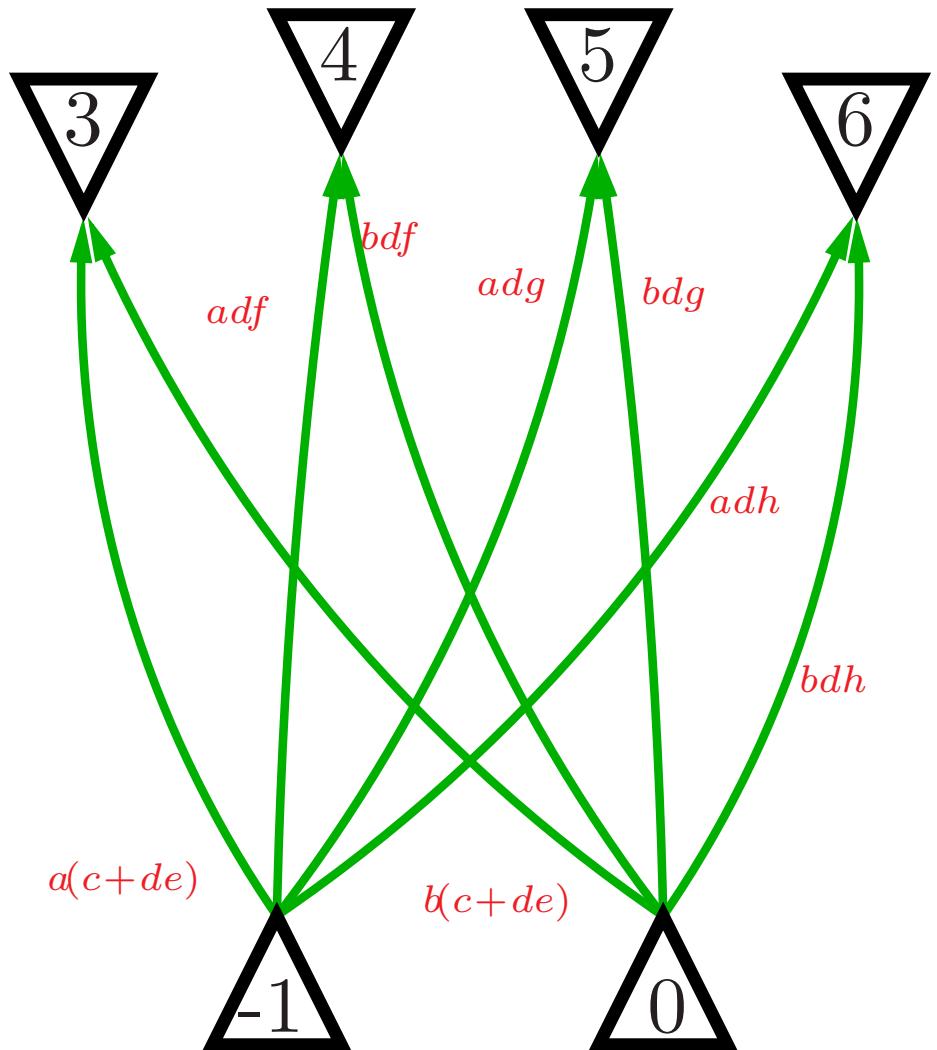
$$\begin{aligned}dv_{ac} &= dv_a * dv_c \\dv_{ad} &= dv_a * dv_d \\dv_{bc} &= dv_b * dv_c \\dv_{bd} &= dv_b * dv_d\end{aligned}$$

## Vertex Elimination



$$\begin{aligned}dv_{ac} &= dv_a * dv_c \\dv_{ad} &= dv_a * dv_d \\dv_{bc} &= dv_b * dv_c \\dv_{bd} &= dv_b * dv_d\end{aligned}$$

## Vertex Elimination



$$dv_{ac} = dv_a * dv_c$$

$$dv_{ad} = dv_a * dv_d$$

$$dv_{bc} = dv_b * dv_c$$

$$dv_{bd} = dv_b * dv_d$$

$$dv_{ade} = dv_{ad} * dv_e$$

$$dv_{adf} = dv_{ad} * dv_f$$

$$dv_{adg} = dv_{ad} * dv_g$$

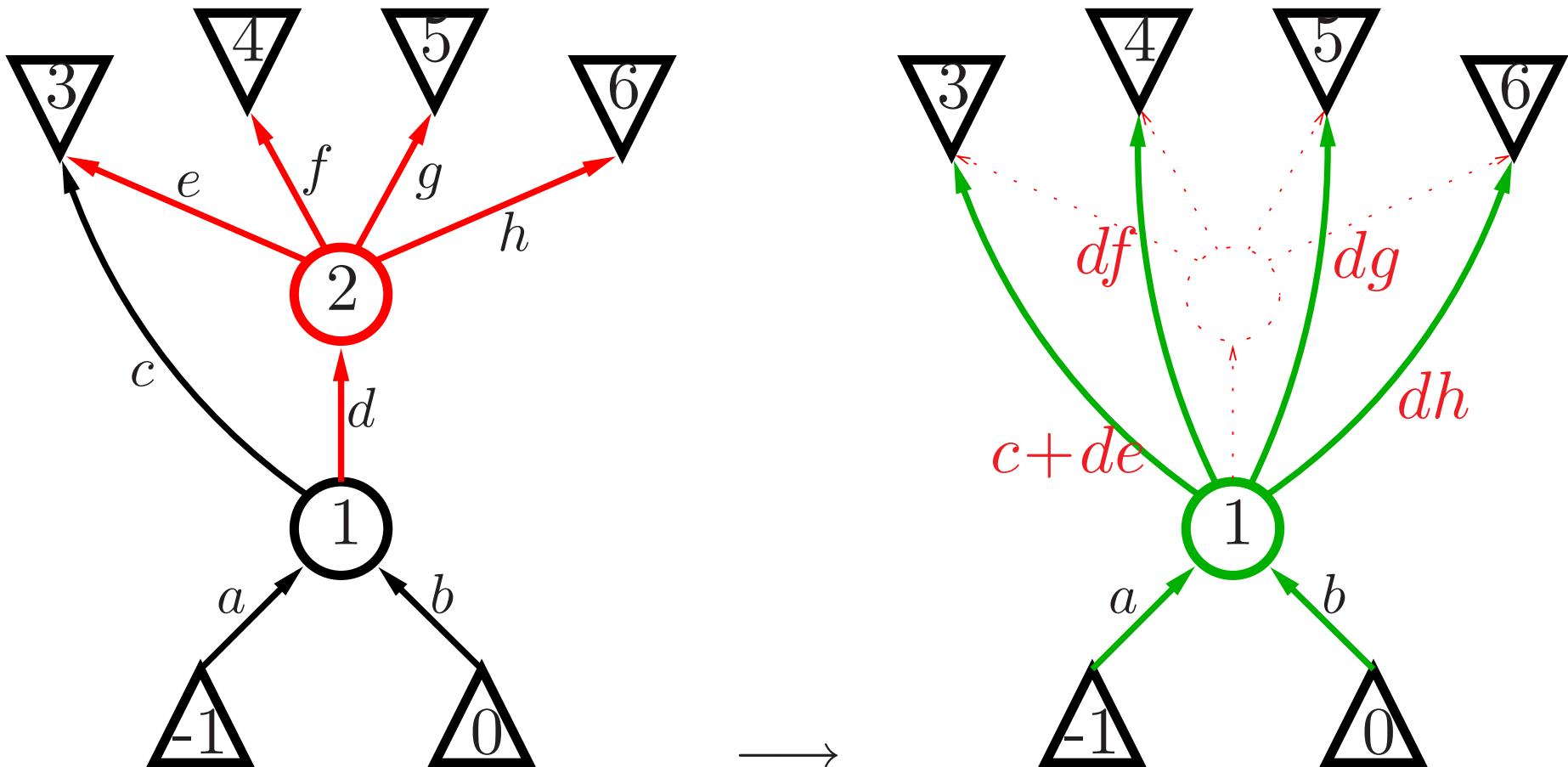
$$dv_{adh} = dv_{ad} * dv_h$$

$$dv_{bde} = dv_{bd} * dv_e$$

$$dv_{bdg} = dv_{bd} * dv_g$$

$$dv_{bdh} = dv_{bd} * dv_h$$

## Vertex Elimination



## Vertex Elimination

$$\begin{array}{c}
 \begin{matrix} & v_2, v_1 \\ \left( \begin{array}{cc} a(c + de) & b(c + de) \\ a(df) & b(df) \\ a(dg) & b(dg) \\ a(dh) & b(dh) \end{array} \right) \end{matrix} & \begin{matrix} & v_1, v_2 \\ \left( \begin{array}{cc} ac + (ad)e & bc + (bd)e \\ (ad)f & (bd)f \\ (ad)g & (bd)g \\ (ad)h & (bd)h \end{array} \right) \end{matrix}
 \end{array}$$

- Makes good use of associativity  
 $(ad), (bd)$
- doesn't make good use of distributivity  
 $a(c + de), b(c + de)$
- Makes good use of associativity  
 $(ad), (bd)$
- doesn't make good use of distributivity  
 $a(c + de), b(c + de)$

## More Elemental: Edge Elimination (Naumann 1999)

Front-eliminate edge  $(v_i, v_j)$ :

- Create new edges  $(v_i, v_k)$  for all  $v_k \in S_{v_j}$ ; label with  $c_{ki} = c_{ji} * c_{kj}$
- If edge  $(v_i, v_k)$  already exists, set  $c_{ki} = c_{ki} + c_{ji} * c_{kj}$  (absorption)

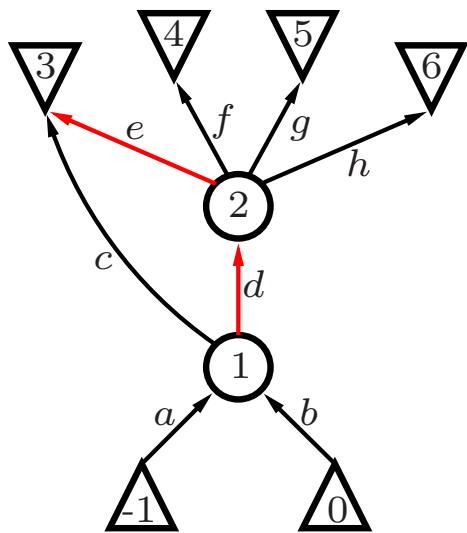
Back-eliminate edge  $(v_j, v_k)$ :

- Create new edges  $(v_i, v_k)$  for all  $v_i \in P_{v_j}$ ; label with  $c_{ki} = c_{ji} * c_{kj}$
- If edge  $(v_i, v_k)$  already exists, set  $c_{ki} = c_{ki} + c_{ji} * c_{kj}$  (absorption)

Motivation: “Lion Graph”:

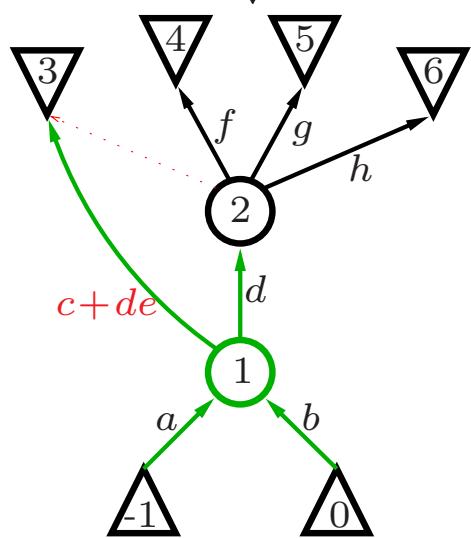
Even larger search space:  $O(|V|^2!)$  possible edge elimination sequences

## Lion Example



$$J = \begin{pmatrix} ac + ade & bc + bde \\ adf & bdf \\ adg & bdg \\ adh & bdh \end{pmatrix}$$

↓ back-elim.  $e = (2, 3)$ .  $P_2 = \{1\}, (1, 2) = d$ .



$$J = \begin{pmatrix} a(c + de) & b(c + de) \\ (ad)f & (bd)f \\ (ad)g & (bd)g \\ (ad)h & (bd)h \end{pmatrix}$$

## Most Elemental??: Face Elimination (Naumann 2004)

Eliminations in *dual graph*  $\tilde{G}$

Motivation: “bat graph”

- Really *directed line graph* of  $G$ , (with some added extras)
- If edge  $(v_i, v_k)$  already exists, set  $c_{ki} = c_{ki} + c_{ji} * c_{kj}$  (absorption)
- Even larger search space:  $O(|V|^2!)$  possible edge elimination sequences

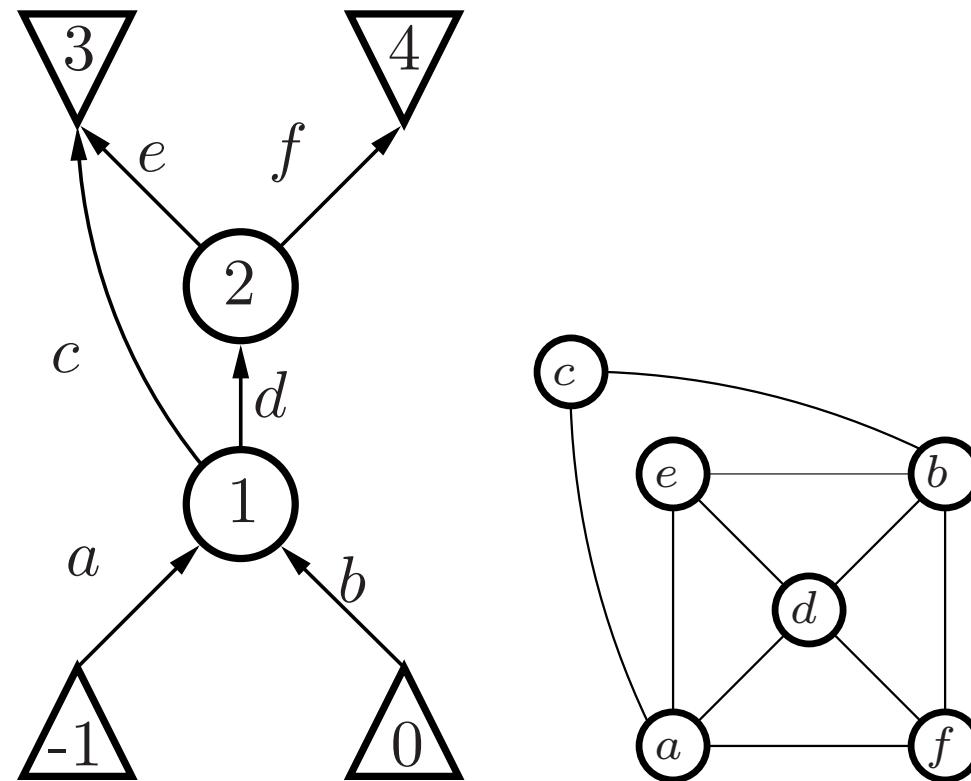
## Path Comparability Graph

Define (strict) partial order  $P = \langle E, <_p \rangle$  on the edge set  $E$  of  $\mathbf{G}$ :

$$a <_p b \Leftrightarrow a \text{ comes before } b \text{ in a path in } \mathbf{G}$$

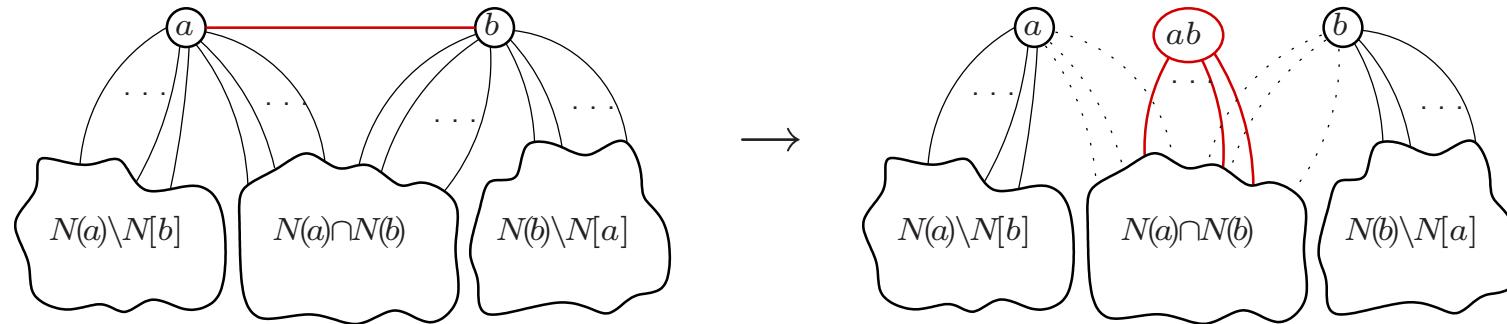
Note: Dual graph  $\tilde{\mathbf{G}}$  is the graph of the covering relation  $\prec_p$

## Path Comparability Graph



# Path Comparability Graph

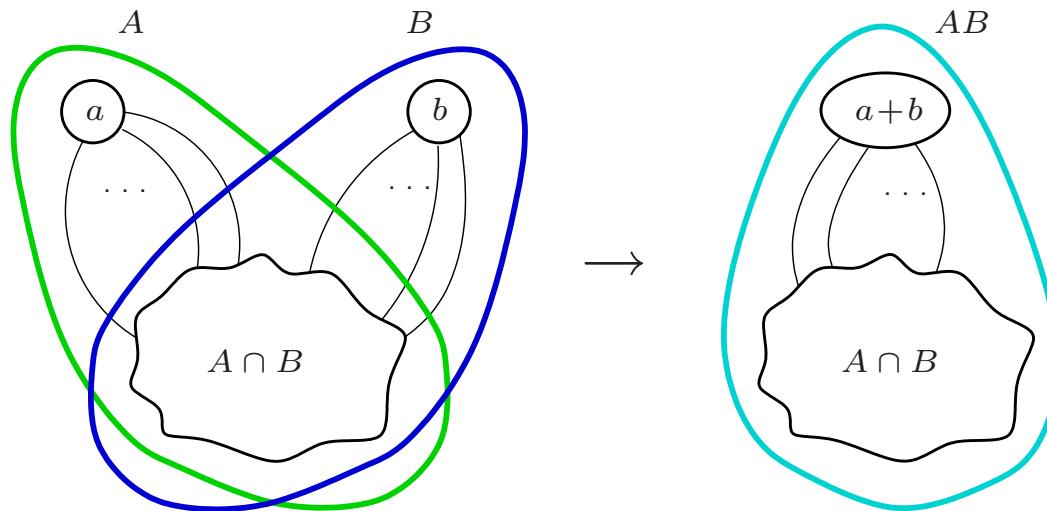
Edge Contractions: contract edge  $\{a, b\}$  to create new vertex  $ab$



- Correspond to multiplications in generated derivative code:  
$$dv_{ab} = dv_a * dv_b$$

## Path Comparability Graph

Clique Merges: merge two vertices  $a, b$  to create new vertex  $a + b$

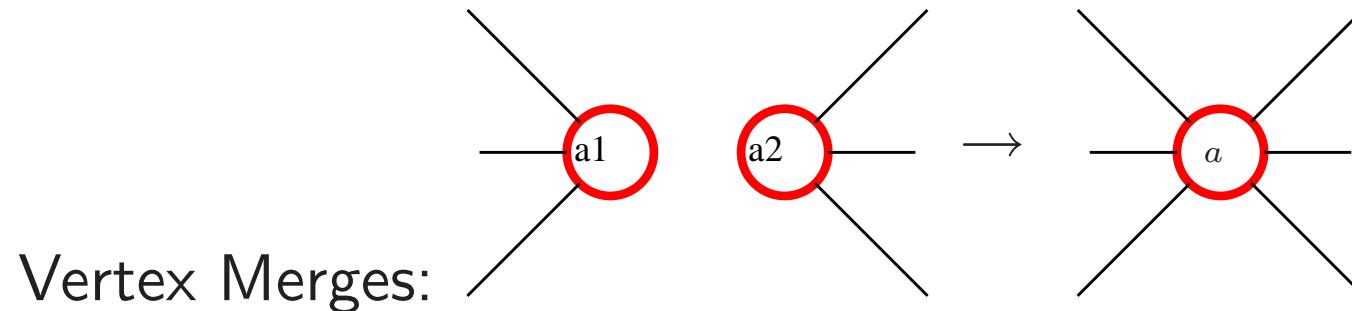


- Correspond to additions in generated derivative code:  $dv_{a+b} = dv_a + dv_b$

## Algebraic Dependences Between Local Partials

- Recent revelation
- Key in NP-completeness proof of OJA

## Exploiting Algebraic Dependencies in $G^p$



- Merge two vertices  $a_1, a_2$  to create new vertex  $a$  where  $N(a) = N(a_1) \cup N(a_2)$
- undefined if  $\{a, b\} \in E^p$

## Open Problems

- Complexity of vertex/edge/face elimination and edge contraction
- No free refill conjecture
- Complexity of structural Jacobian Accumulation
- Optimality of face elimination
- taping via inlinable calls
- Issues with path comparability graph:
  - Algorithmic Complexity

- Generality (comm and assoc and dist?)
- Issues with vertex merges (cliques no longer maximal, etc.)