CS 3750 Machine Learning
Lecture 4

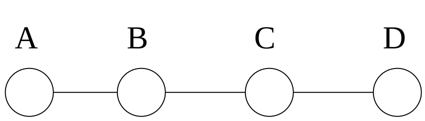
Graphical models: inference

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square

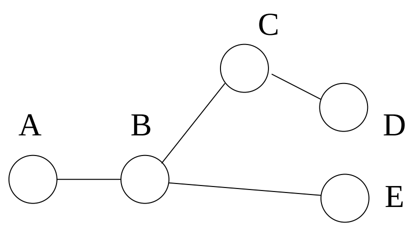
Inferences

Last lecture:
- **Exact inferences on chains and trees**
- **Factor graph representation and inference**

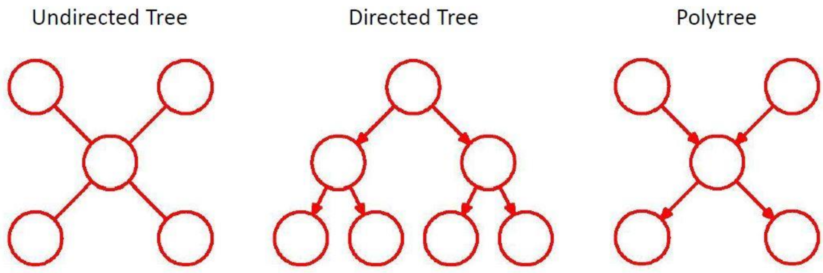
chain



tree



Trees



Factor graph

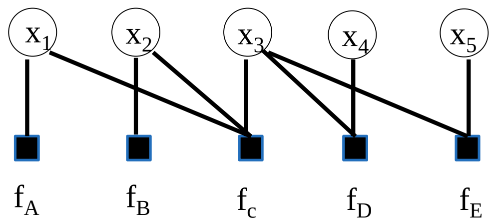
A graphical representation that lets us express a factorization of a function over a set of variables

A factor graph is bipartite graph where:

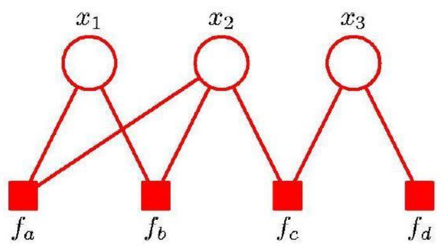
- One layer is formed by variables
- Another layer is formed by factors or functions on subsets of variables

Example: a function over variables $x_1, x_2, \ldots x_5$

$$g(x_1, x_2, \ldots x_5) = f_A(x_1)\, f_B(x_2)\, f_C(x_1, x_2, x_3)\, f_D(x_3, x_4)\, f_E(x_3, x_5)$$



Factor Graphs



$$p(\mathbf{x}) = f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3)$$

$$p(\mathbf{x}) = \prod_s f_s(\mathbf{x}_s)$$

Inferences on factor graphs

- Efficient inference algorithms for factor graphs built for trees [Frey, 1998; Kschischnang *et al.*, 2001] **:**
  - **Sum-product algorithm**
  - **Max product algorithm**

Inferences

Last lecture:

- Exact inferences on chains and trees
- Factor graph representation and inference on factor graphs

Open question:

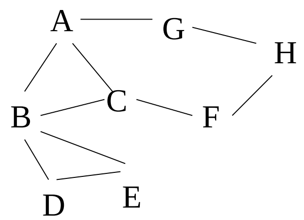
- Exact inferences on arbitrary MRFs and BBNs

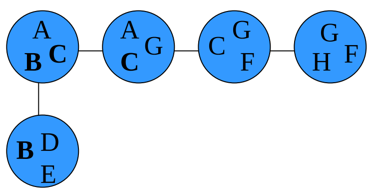
Clique tree algorithm:

- **Clique tree = Junction tree = tree decomposition of the graph**

Tree decomposition of the graph

- **A tree decomposition of a graph G:**
  - A tree *T* with a vertex set associated to every node.
  - For all edges $\{v,w\} \in G$: there is a set containing both v and w in T.
  - For every $v \in G$: the nodes in T that contain v form a connected subtree.





Conversion of the BBN and MRF to a Clique tree

MRF conversion:

Option 1:

- Via triangulation to form a chordal graph
- Cliques in the chordal graph define the clique tree

Option 2:

- from the induced graph built by running the variable elimination procedure
- Cliques are factors generated during the procedure

BBN conversion:

- Convert the BBN to an MRF – a moral graph
- Apply MRF conversion

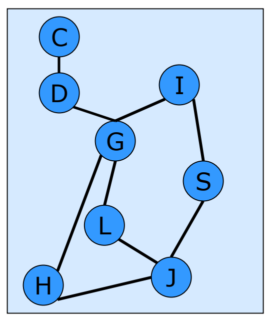
Clique tree algorithms

- We have precompiled a clique tree.
- So how to take advantage of the clique tree to perform inferences?

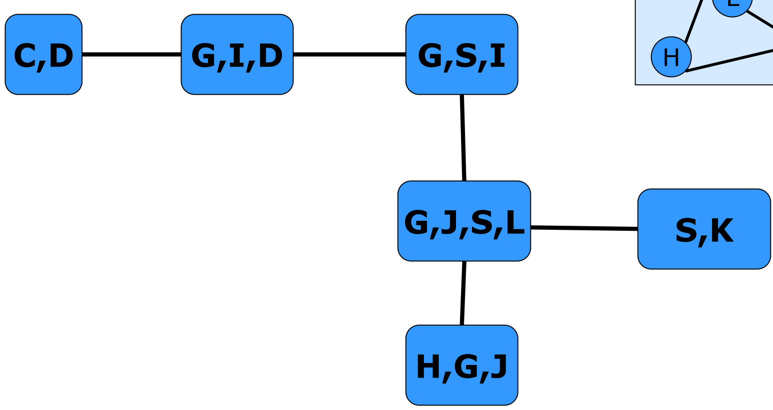
VE on the Clique tree

- Variable Elimination on the clique tree
  - works on *factors*
- Makes factor a data structure
  - Sends and receives messages

Clique trees



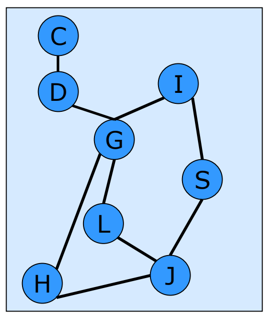
- **Example clique tree**



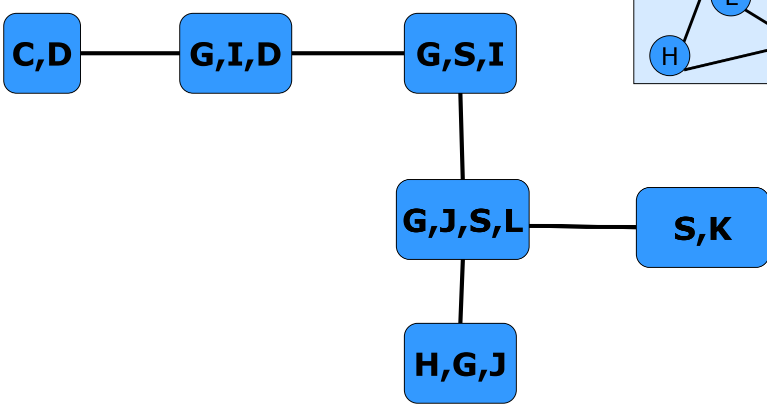
Clique tree properties

- **Running intersection property**
  – if C_i and C_j both contain a vertex X, then all cliques on the unique path between them also contain X
- **Sepset** $S_{ij} = C_i \cap C_j$
  – **separation** set: Variables **X** on one side of sepset are separated from the variables **Y** on the other side in the factor graph given variables in **S**

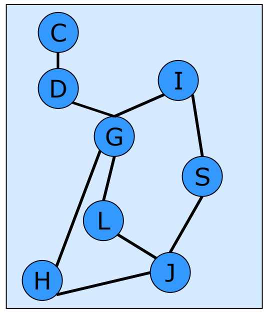
Clique trees



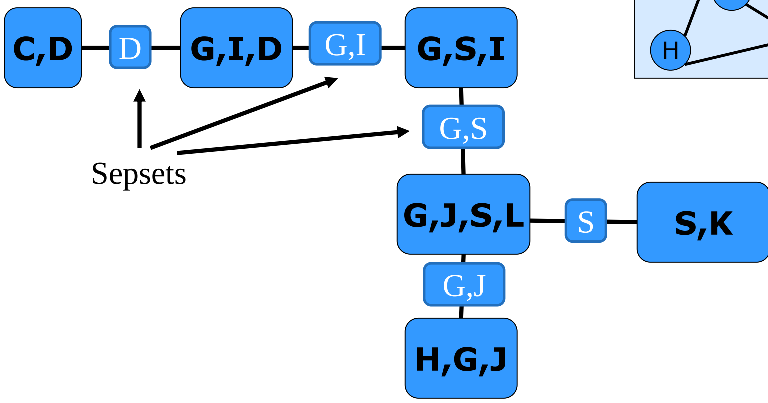
- **Running intersection:**
  E.g. Cliques involving G form a connected subtree.



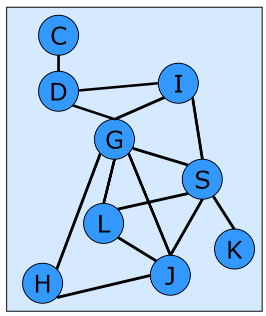
Clique trees



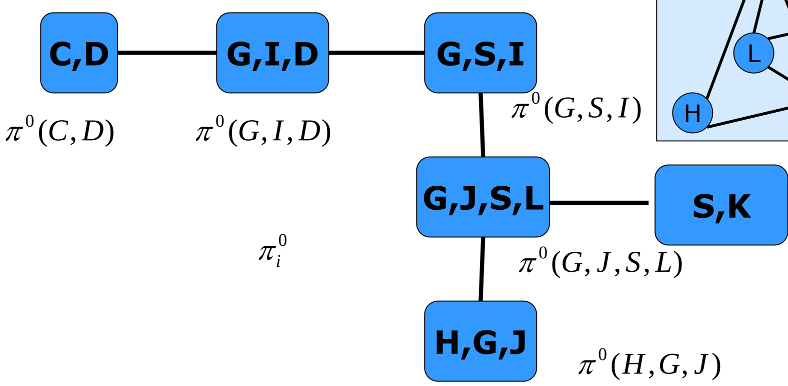
- **Sepsets:** $S_{ij} = C_i \cap C_j$
- Variables **X** on one side of a sepset are separated from the variables **Y** on the other side given variables in **S**



Clique trees



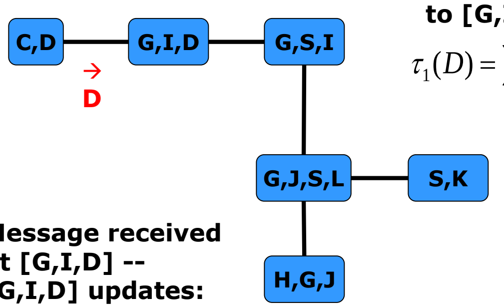
Initial potentials π_i^0 **:**
Assign factors to cliques and multiply them.



$$p(C,D,G,I,S,J,L,K,H)$$
$$= \pi^0(C,D)\pi^0(G,I,D)\pi^0(G,S,I)\pi^0(G,J,S,L)\pi^0(S,K)\pi^0(H,G,J)$$

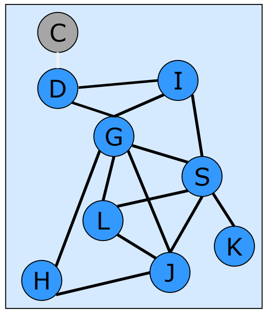
Message Passing VE

- **Query for P(J)**
  - **Eliminate C:**



Message sent from [C,D] to [G,I,D]

$$\tau_1(D) = \sum_C \pi_1^0[C,D]$$

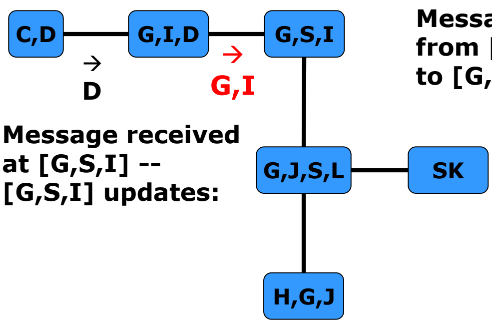


Message received at [G,I,D] -- [G,I,D] updates:

$$\pi_2[G,I,D] = \tau_1(D) \times \pi_2^0[G,I,D]$$

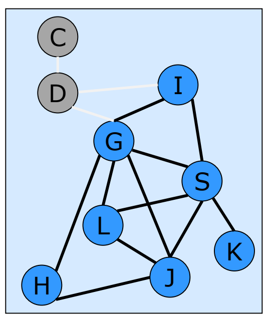
Message Passing VE

- Query for P(J)
  - Eliminate D: $\tau_2(G,I) = \sum_D \pi_2[G,I,D]$



Message sent from [G,I,D] to [G,S,I]

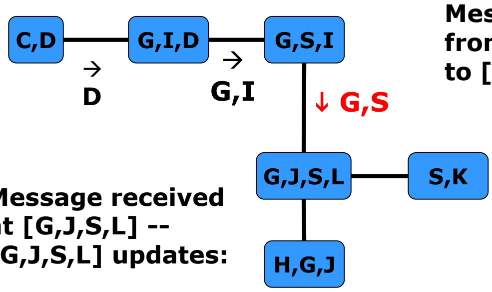
Message received at [G,S,I] -- [G,S,I] updates:



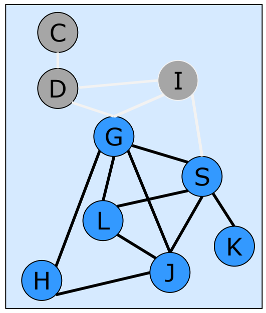
$$\pi_3[G,S,I] = \tau_2(G,I) \times \pi_3^0[G,S,I]$$

Message Passing VE

- Query for P(J)
  - Eliminate I: $\tau_3(G,S) = \sum_I \pi_3[G,S,I]$



Message sent from [G,S,I] to [G,J,S,L]



Message received at [G,J,S,L] -- [G,J,S,L] updates:

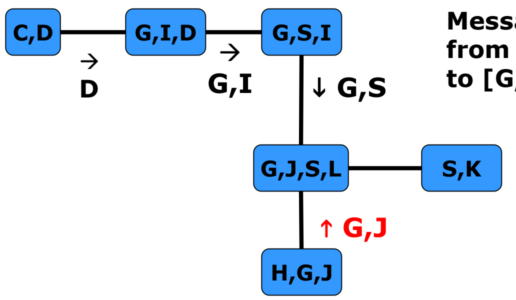
$$\pi_4[G,J,S,L] = \tau_3(G,S) \times \pi_4^0[G,J,S,L]$$

!

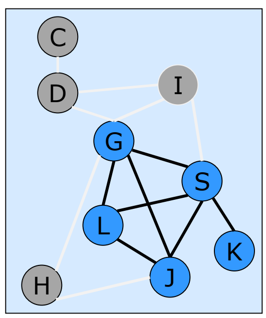
[G,J,S,L] is not **ready**!

Message Passing VE

- Query for P(J)
  - Eliminate H: $\tau_4(G,J) = \sum_H \pi_5[H,G,J]$



Message sent from [H,G,J] to [G,J,S,L]

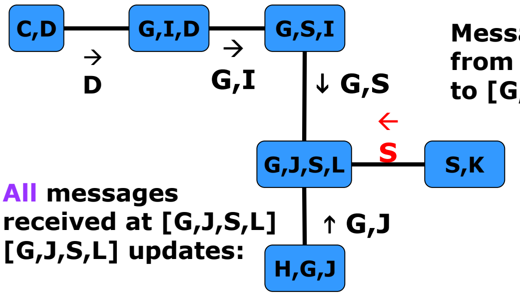


$$\pi_4[G,J,S,L] = \tau_3(G,S) \times \tau_4(G,J) \times \pi_4^0[G,J,S,L]$$

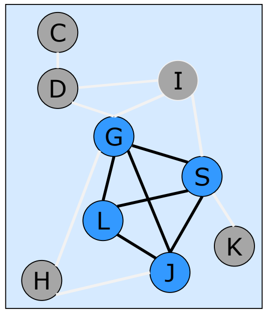
And …

Message Passing VE

- Query for P(J)
  - Eliminate K: $\tau_6(S) = \sum_K \pi^0[S,K]$



Message sent from [S,K] to [G,J,S,L]



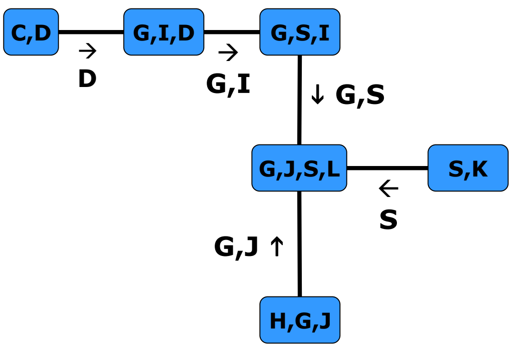
All messages received at [G,J,S,L]
[G,J,S,L] updates:

$$\pi_4[G,J,S,L] = \tau_3(G,S) \times \tau_4(G,J) \times \tau_6(S) \times \pi_4^0[G,J,S,L]$$

And calculate P(J) from it by summing out G,S,L

Message Passing VE

- [G,J,S,L] clique potential
- … is used to finish the inference

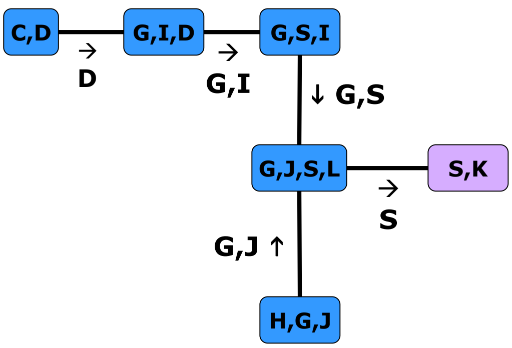


Message passing VE

- Often, **many marginals are desired**
  - Inefficient to re-run each inference from scratch
  - One distinct message per edge & direction
- **Methods :**
  - Compute (**unnormalized**) marginals for any vertex (clique) of the tree
  - Results in a ***calibrated* clique tree** $\sum_{C_i - S_{ij}} \pi_i = \sum_{C_j - S_{ij}} \pi_j$
- **Recap:** three kinds of factor objects
  - Initial potentials, final potentials and messages

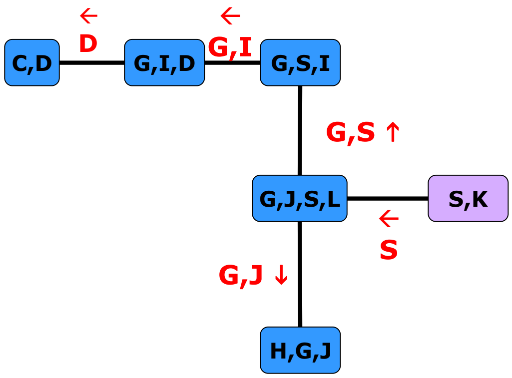
Two-pass message passing VE

- Chose the root clique, e.g. [S,K]
- Propagate messages to the root



Two-pass message passing VE

- Send messages back from the root



Message Passing: BP

- **Belief propagation**
  - A different algorithm but equivalent to variable elimination in terms of the results
  - Asynchronous implementation

Message Passing: BP

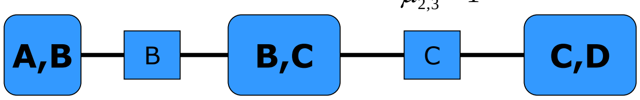
- **Each node:** multiply all the messages and divide by the one that is coming from node we are sending **the message to**
  - Clearly the same as VE

$$\delta_{i \to j} = \frac{\sum_{C_i - S_{ij}} \pi_i}{\delta_{j \to i}} = \frac{\sum_{C_i - S_{ij}} \pi_i^{\,0} \prod_{k \in N(i)} \delta_{k \to i}}{\delta_{j \to i}} = \sum_{C_i - S_{ij}} \pi_i^{\,0} \prod_{k \in N(i) \setminus j} \delta_{k \to i}$$

  - Initialize the messages on the edges to 1

Message Passing: BP

$\mu_{2,3} = 1$



$\pi_1^0(A,B)$ $\pi_2^0(B,C)$ → $\pi_3^0(C,D)$

Store the last message on the edge and divide each passing message by the last stored.

$$\delta_{2\to 3} = \left(\sum_B \pi_2^0(B,C)\right)$$

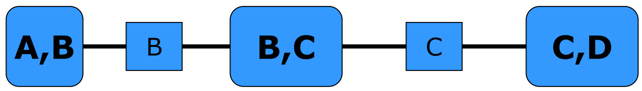
$$\pi_3(C,D) = \pi_3^0(C,D)\frac{\delta_{2\to 3}}{\mu_{2,3}} = \pi_3^0(C,D)\sum_B \pi_2^0(B,C)$$

$$\mu_{2,3} = \delta_{2\to 3} = \left(\sum_B \pi_2^0(B,C)\right)$$

New message stored

Message Passing: BP

$$\mu_{2,3} = \left(\sum_B \pi_2^0(B,C) \right)$$



$\pi_1^0(A,B)$ $\qquad \pi_2^0(B,C)$ $\qquad \longleftarrow \qquad \pi_3(C,D)$

Store the last message on the edge and divide each passing message by the last stored.

$$\pi_3(C,D) = \pi_3^0(C,D)\sum_B \pi_2^0(B,C) = \pi_3^0(C,D)\mu_{2,3}$$

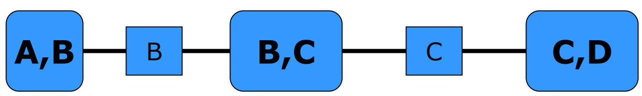
$$\delta_{3->2} = \left(\sum_D \pi_3(C,D) \right)$$

$$\pi_2(B,C) = \pi_2^0(B,C)\frac{\delta_{3->2}}{\mu_{2,3}(C)} = \frac{\pi_2^0(B,C)}{\mu_{2,3}(C)} \times \sum_D \pi_3^0(C,D) \times \mu_{2,3}(C) = \pi_2^0(B,C) \times \sum_D \pi_3^0(C,D)$$

$$\mu_{2,3} = \delta_{3->2} = \left(\sum_D \pi_3(C,D) \right) = \sum_D \pi_3^0(C,D)\sum_B \pi_2^0(B,C)$$ New message

Message Passing: BP

$$\mu_{2,3} = \sum_D \pi_3^0(C,D)\sum_B \pi_2^0(B,C)$$



$\pi_1^0(A,B)$ $\qquad \pi_2(B,C)$ $\qquad \longleftarrow \qquad \pi_3(C,D)$

Store the last message on the edge and divide each passing message by the last stored.

$$\pi_3(C,D) = \pi_3^0(C,D)\sum_B \pi_2^0(B,C)$$

$$\delta_{3->2} = \left(\sum_D \pi_3(C,D) \right)$$

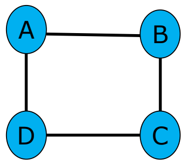
$$\pi_2(B,C) = \pi_2^0(B,C) \times \sum_D \pi_3^0(C,D)$$

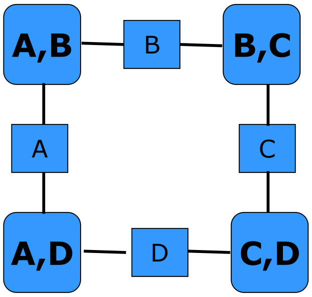
The same as before

$$\pi_2(B,C) = \pi_2(B,C)\frac{\delta_{3->2}}{\mu_{2,3}(C)} = \pi_2(B,C) \times \frac{\sum_D \pi_3^0(C,D) \times \sum_B \pi_2^0(B,C)}{\sum_D \pi_3^0(C,D) \times \sum_B \pi_2^0(B,C)} = \pi_2(B,C)$$

Loopy belief propagation

- The asynchronous BP algorithm works on clique trees
- What if we run the belief propagation algorithm on a non-tree structure?





- Sometimes converges
- If it converges it leads to an approximate solution
- **Advantage:** tractable for large graphs

Loopy belief propagation

- If the BP algorithm converges, it converges to an optimum of the Bethe free energy

See papers:

- Yedidia J.S., Freeman W.T. and Weiss Y. Generalized Belief Propagation, 2000
- Yedidia J.S., Freeman W.T. and Weiss Y. Understanding Belief Propagation and Its Generalizations, 2001