Ch 4 Classic Parsing Algorithms

1, Pure bottom-up: CYK – chart parsing, 1960s
(Cocke, Younger, Kasami)

2, Pure top-down: Earley-parser, 1970s
(Earley, Stockle)

3, Recursive/iterative: Inside-outside algorithm, 1990s
(Lori, Young)

4, Heuristic: Best-first Chart Parsing, 2000s
(Chaniak, Johnson, Klein, Manning, et al)

Chart Parsing in NLP

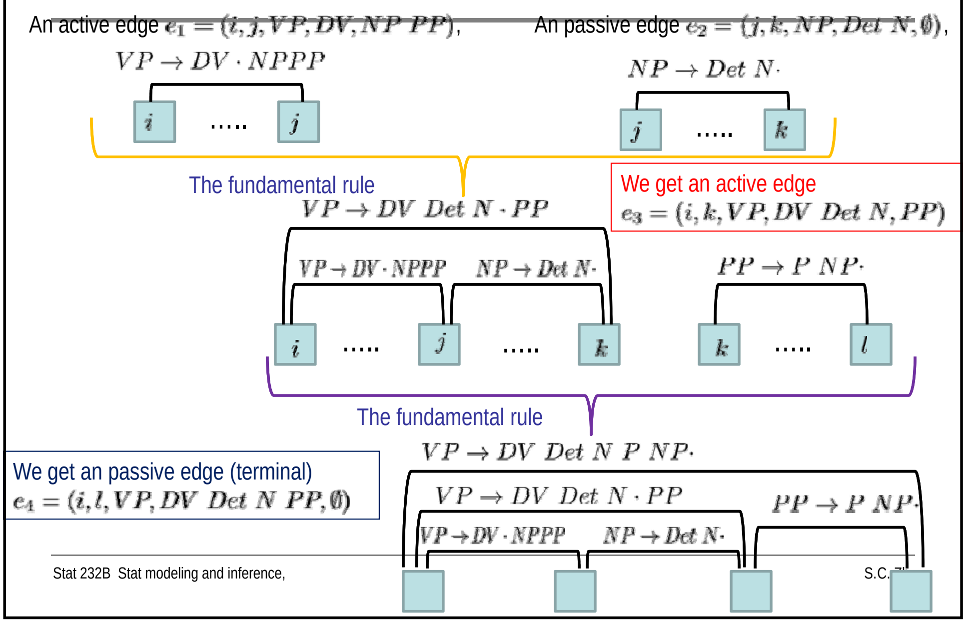
- Motivation: General search methods are not best for syntactic parsing because the same syntactic constituent may be rederived many times as a part of larger constituents due to the local ambiguities of grammar.
- Basic idea of chart parsing: Don't throw away any information. Keep a record --- a chart --- of all the structures we have found.
- Two types of chart parsing
  - Passive chart parsing: it is a bottom-up parsing
  - Active chart parsing, by introducing the agenda, say, agenda-driven chart parsing
    - Bottom-up active chart parsing
    - Top-down active chart parsing
    - The agenda is used to prioritize constituents to be processed, implemented as
      - a stack to simulate depth-first search (DFS)
      - a queue to simulate breadth-first search (BFS)
      - a priority queue to simulate best-first search (FoM)

What is a chart?

- A chart is a form of well-formed substring table [Partial parse graph],
  - It plays the role of the memo-table as in DP.
  - It keeps track of partial derivations so nothing has to be rederived

- Formally, charts are represented by directed graphs $G =< V, E >$
  - For an input sentence with n words, $V = \{0, 1, 2, \cdots, n\}$, and the i-th word is marked by two nodes in V, say, node $i-1$ and node i.
  - Each edge $e \in E$ characterizing a completed or partial constituent spanning a group of words, say, $e = (start, finish, label, found, tofound) \in E$
    - $label$ is a nonterminal node in the grammar, say, LHS of a certain rule
    - $found$ is a part of RHS of $label$ which explains words from $start$ to $finish$
    - $tofound$ is the remainder of beside the $found$ part
  - Active edge: $tofound$ is not empty
  - Inactive edge (passive edge): $tofound$ is empty

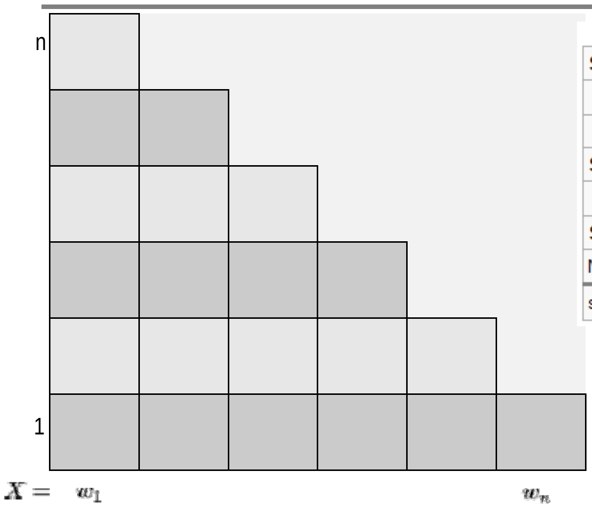
The fundamental rule for combining active and passive edges



What is an agenda?

- An agenda is a data structure that keeps track of the things we still have to do. Simply put, it is a set of edges waiting to be added to the chart.
- The agenda determines in what order edges are added to the chart
  - Stack agenda for depth-first search
  - Queue agenda for breadth-first search
  - Priority queue agenda for best-first search
- The order of elements in agenda is decided by the figures of merit (FOM) of elements, which is one of the keys to design an efficient parsing algorithm.

1, CYK algorithm: an example



CYK table

S						
	VP					
S						
	VP			PP		
S		NP			NP	
NP	V, VP	Det.	N	P	Det	N
she	eats	a	fish	with	a	fork

Pure bottom-up parser: Cocke–Younger–Kasami (CYK) algorithm

- CYK algorithm is a type of bottom-up passive chart parsing algorithm.
- The goal: to determine whether a sentence can be generated by a given context-free grammar (say, recognition) and, if so, how it can be generated (say, parse tree construction) . The context-free grammar must be in Chomsky normal form (CNF).
- The worst case running time of CYK is $O(n^3|G|)$ where n is the length of input sentence and $|G|$ is the size of grammar.
  - The drawback of all known transformations into CNF is that they can lead to an undesirable bloat in grammar size. Let g be the size of original grammar, the size blow-up in the worst case may range from g^2 to 2^{2g}, depending on the used transformation algorithm.

CYK algorithm

- Input: a sentence $X = w_1 \cdots w_n$ and the grammar G with S being the root.
  - Let $w_{ij} = w_i x_{i+1} \cdots w_{i+j-1}$ be the substring of X of length j starting with w_i. Then, we have $X = w_{1n}$.
- Output: verify whether $S \Rightarrow X$, if yes, construct all possible parse trees.
- The algorithm: for every w_{ij} and every rule $R \in G$, it determines if $R \Rightarrow w_{ij}$ and the probability if necessary.
  - Define a auxiliary 4-tuple variable for each rule $R_k \in G$:
    $v_k = (k, probability, pointerLeft, pointerRight)$
  - CYK table with the entries $V[i,j], 1 \leq i \leq n, 1 \leq j \leq n-i+1$ storing the auxiliary variables of the rules which can explain substring w_{ij}.
  - Start with substrings of length 1: $w_{i1} = w_i, 1 \leq i \leq n$, set
    $V[i,1[= \{v_k = (k, Prob(R_k|w_{i1}), NULL, NULL)|R_k \Rightarrow w_{i1}, R_k \in G\}$
  - Continue with substrings of length $j = 2, 3, \cdots, n-i+1$
    - For w_{ij}, consider all possible two-part partitions $w_{ij} = w_{im} w_{i+m\ j-m}, 1 \leq m \leq j$ set $V[i,j[= \{v_k = (k, Prob(R_k|w_{ij}), v_{k_l}, v_{k_r})|R_k \Rightarrow R_{k_l} R_{k_r}, R_{k_l} \Rightarrow w_{im}, R_{k_r} \Rightarrow w_{i+m\ j-m}, R_k, R_{k_l}, R_{k_r} \in G\}$
- The algorithm has three nested loops each of which has the range at most 1 to n. With each loop, it check all the rules. So, the worst case of running time is $O(n^3|G|)$.

2, Bottom-up passive chart parsing

- Basic algorithm flow: Scan the input sentence left-to-right and make use of CFG rules right-to-left to add more edges into the chart by using the fundamental rule.

Grammar:

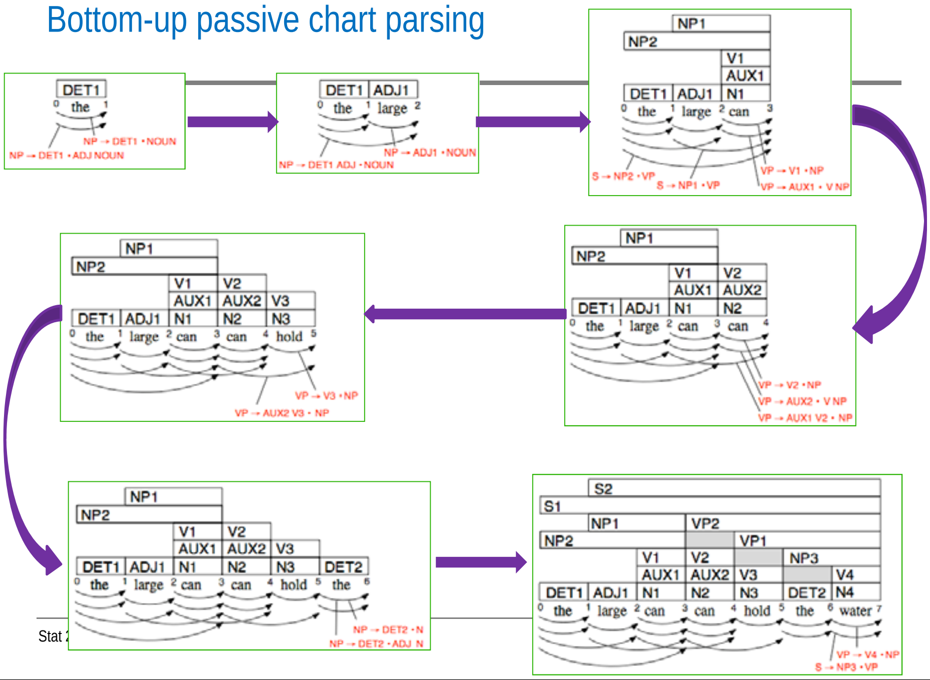
1. S → NP VP
2. NP → DET ADJ N
3. NP → DET N
4. NP → ADJ N
5. VP → AUX V NP
6. VP → V NP

Lexicon:

the... DET
large... ADJ
can... AUX, N, V
hold... N, V
water... N, V

Sentence: $_0$ *The* $_1$ *large* $_2$ *can* $_3$ *can* $_4$ *hold* $_5$ *the* $_6$ *water* $_7$

Bottom-up passive chart parsing



3, Bottom-up active chart parsing

- Algorithm flow:
  - (1) Initialize chart and agenda

    Chart = empty, Agenda = {passive edges for all possible rules for all the words}
  - (2) Repeat until agenda is empty
    - (a) Select an edge from agenda in terms of DFS, BFS or best-first search, etc.

      $$e = (start, finish, label, found, tofound)$$
    - (b) Add the selected edge e to the chart in position $(start, finish)$ if it is not on the chart
    - (c) Use the fundamental rule to combine the selected edge e with other edges from the chart, and then add all obtained edges on the agenda
    - (d) If the selected edge e is a PASSIVE one, say, $tofound = \emptyset$, then look for grammar rules which have $found$ as the first symbol on the RHS, say, $r \to found\ V_{remaining}$. For each r, build active edge e' and add it on the agenda

      $$e' = (start, start, r, \emptyset, found\ V_{remaining})$$
  - (3) succeed if there is a passive edge $e = (0, n, S, found, \emptyset)$, where S is the root node in the grammar.

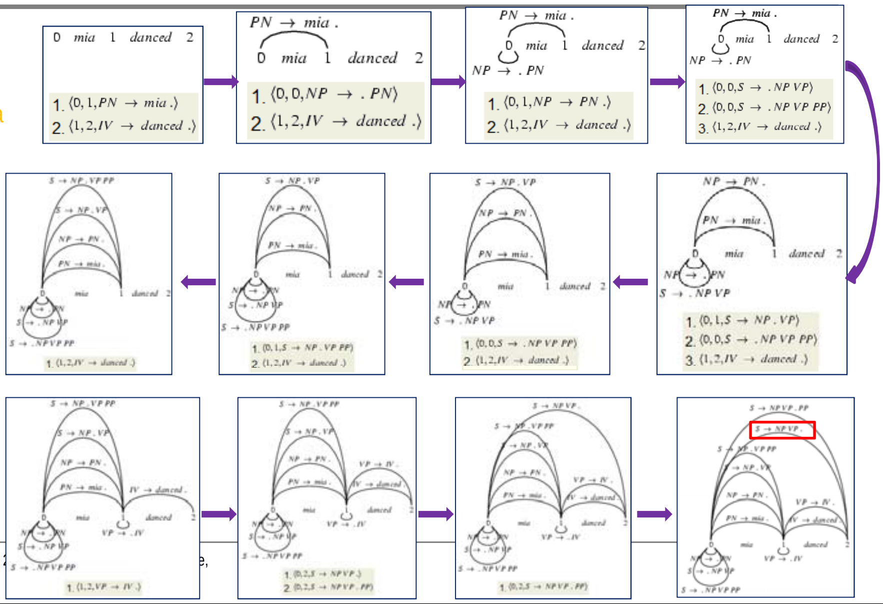
An example of bottom-up active chart parsing

$S \to NP\ VP$
$S \to NP\ VP\ PP$
$NP \to PN$
$VP \to IV$
$PP \to P\ NP$
$PN \to mia$
$IV \to danced$

Mia danced

Chart

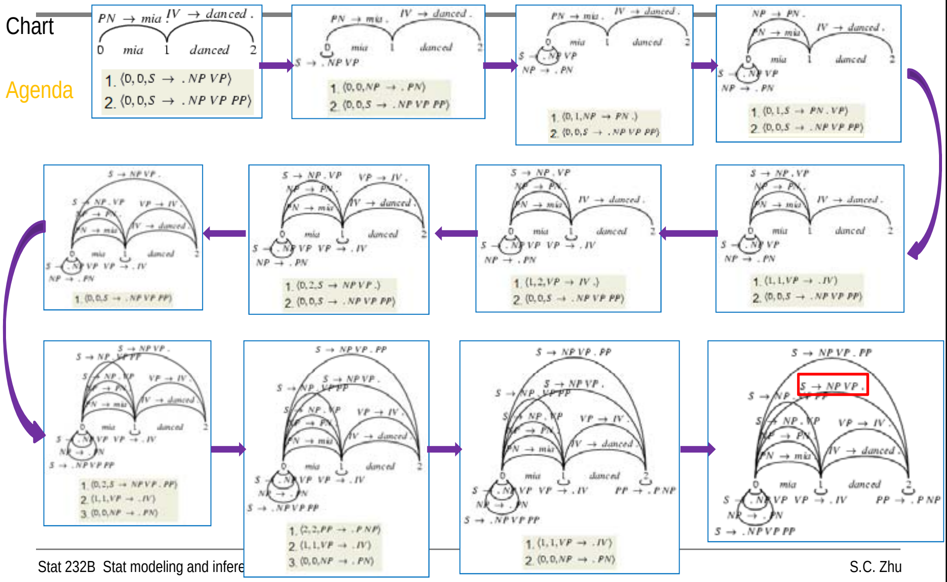
Agenda



An example of top-down active chart parsing

$S \rightarrow NP\ VP$
$S \rightarrow NP\ VP\ PP$
$NP \rightarrow PN$
$VP \rightarrow IV$
$PP \rightarrow P\ NP$
$PN \rightarrow mia$
$IV \rightarrow danced$

Mia danced



4, Top-down active chart parsing (Earley parser)

- Top-down vs. Bottom-up active chart parsing
  - Bottom-up chart parsing checks the input sentence and builds each constituent exactly once. Avoid duplication of effort!
  - But bottom-up chart parsing may build constituents that cannot be used legally, as the example shown in previous slide.
  - By working bottom up, the algorithm reads the rules right-to-left, and starts with the information in passive edges.

  - Top-down chart parsing is highly predictive. Only grammar rules that can be legally applied will be put on the chart.
  - By working top-down, the algorithm reads the rules left-to-right and starts with the information in active edges.

Top-down active chart parsing (Earley parser)

- Algorithm flow:
  - (1) Initialize chart and agenda

    Chart={passive edges for all possible rules for all words}, Agenda={root rules}
  - (2) Repeat until agenda is empty
    - (a) Select an edge from agenda in terms of DFS, BFS or best-first search, etc.

      $$e = (start, finish, label, found, tofound)$$
    - (b) Add the selected edge e to the chart in position $(start, finish)$ if it is not on the chart
    - (c) Use the fundamental rule to combine the selected edge e with other edges from the chart, and then add all obtained edges on the agenda
    - (d) If the selected edge e is a ACTIVE one, say, $tofound \neq \emptyset$, then look for grammar rules which have the forms $r = tofound \to V_1 \cdots V_m$ For each r, build active edge e' and add it on the agenda

      $$e' = (finish, finish, r, \emptyset, V_1 \cdots V_m)$$
  - (3) succeed if there is a passive edge $e = (0, n, S, found, \emptyset)$, where S is the root node in the grammar.

Example of Early Parser

S->A B C	• a1b1c1
S->D E F	• a1b1c2
A->a1 A->a2	• a1b2c1
B->b1 B->b2	• a1b2c2
C->c1 C->c2	• a2b1c1
D->a1 D->a2	• a2b1c2
E->b1 E->b2	• a2b2c1
F->c1 F->c2	• a2b2c2

Earley-Stolcke parsing algorithm

State set 0	State set 1	State set 2	State set 3
$_0$->.S	Scanned	Scanned	Scanned
	$_0$A->a1.	$_1$B->b1.	$_2$C->c1.
Predicted	$_0$D->a1.	$_1$E->b1.	$_2$F->c1.
$_0$S->.A B C	Completed	Completed	Completed
$_0$S->.D E F	$_0$S->A. B C	$_0$S->A B. C	$_0$S->A B C.
	$_0$S->D. E F	$_0$S->D E. F	$_0$S->D E F.
$_0$A->.a1	Predicted	Predicted	
$_0$A->.a2	$_1$B->.b1	$_2$C->.c1	$_0$->S.
$_0$D->.a1	$_1$B->.b2	$_2$C->.c2	
$_0$D->.a2	$_1$E->.b1	$_2$F->.c1	
	$_1$E->.b2	$_2$F->.c2	

The input is a1 b1 c1

Modifications of the Earley-Stolcke parsing algorithm

E1->A B C
A->a1
A->a2
B->b1
B->b2
C->c1
C->c2

E2->D
D->d

The input is a1 d b2 c1

Modifications of the Earley-Stolcke parsing algorithm

State set 0	State set 1	State set 2	State set 3
${}_0$->.E1 Predicted ${}_0$E1->.A B C ${}_0$A->.a1 ${}_0$A->.a2			
${}_0$->.E2 Predicted ${}_0$E2->.D ${}_0$D->.d			

Modifications of the Earley-Stolcke parsing algorithm

State set 0	State set 1	State set 2	State set 3
${}_0$->.E1 Predicted ${}_0$E1->.A B C ${}_0$A->.a1 ${}_0$A->.a2	Scanned ${}_0$A->a1. Completed ${}_0$E1->A. B C Predicted ${}_1$B->.b1 ${}_1$B->.b2		
${}_0$->.E2 Predicted ${}_0$E2->.D ${}_0$D->.d	 Shifted ${}_1$D->.d		

Modifications of the Earley-Stolcke parsing algorithm

State set 0	State set 1	State set 2	State set 3
${}_0$->.E1 Predicted ${}_0$E1->.A B C ${}_0$A->.a1 ${}_0$A->.a2	Scanned ${}_0$A->a1. Completed ${}_0$E1->A. B C Predicted ${}_1$B->.b1 ${}_1$B->.b2	Shifted ${}_2$B->.b1 ${}_2$B->.b2	
${}_0$->.E2 Predicted ${}_0$E2->.D ${}_0$D->.d	Shifted ${}_1$D->.d	Scanned ${}_1$D->d. Completed ${}_0$E2->D. ${}_0$->E2.	

Modifications of the Earley-Stolcke parsing algorithm

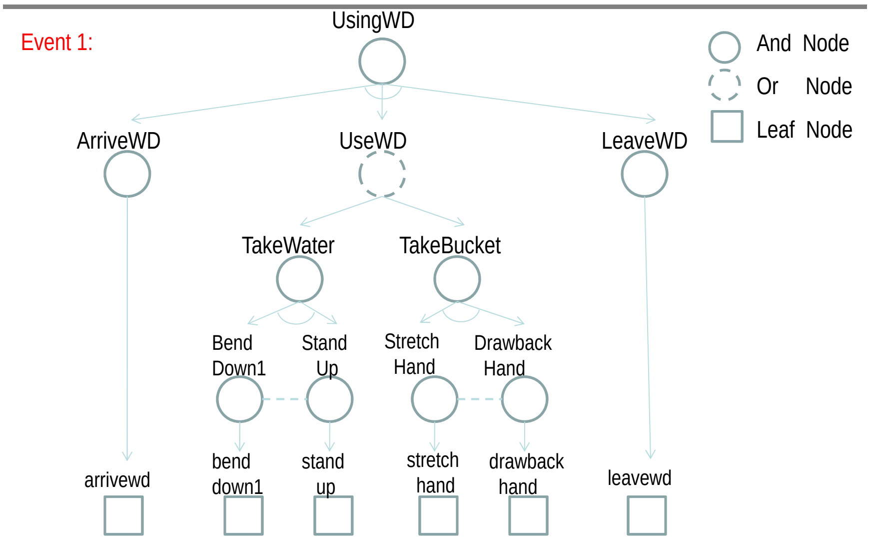
State set 0	State set 1	State set 2	State set 3
${}_0$->.E1 Predicted ${}_0$E1->.A B C ${}_0$A->.a1 ${}_0$A->.a2	Scanned ${}_0$A->a1. Completed ${}_0$E1->A. B C Predicted ${}_1$B->.b1 ${}_1$B->.b2	Shifted ${}_2$B->.b1 ${}_2$B->.b2	Scanned ${}_2$B->b2. Completed ${}_0$E1->A B. C Predicted ${}_3$C->.c1 ${}_3$C->.c2
${}_0$->.E2 Predicted ${}_0$E2->.D ${}_0$D->.d	Shifted ${}_1$D->.d	Scanned ${}_1$D->d. Completed ${}_0$E2->D. ${}_0$->E2.	

5, Grammars for Events

- UsingWD→ArriveWD UseWD LeaveWD
- ArriveWD→arrivewd
- UseWD→TakeWater
- UseWD→TakeBucket
- TakeWater→benddown1 standup
- TakeBucket→stretchhand drawbackhand
- LeaveWD→leavewd
- BendDown→Pickup
- BendDown→TieString
- Pickup→benddown2 standup
- TieString→benddown2 standup

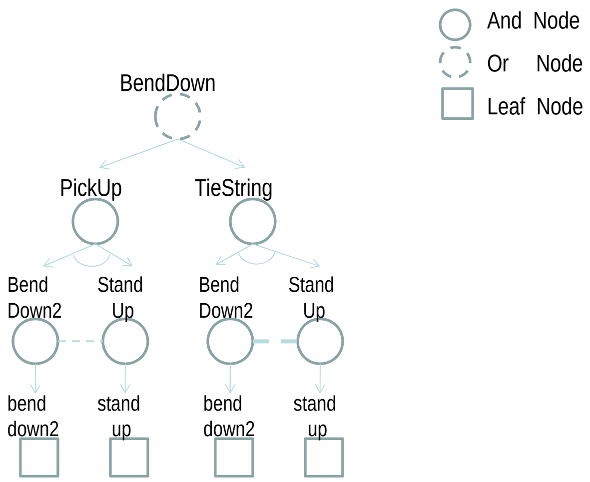
M. Pei, Y. Jia, and S.-C. Zhu, "Parsing Video Events with Goal inference and Intent Prediction," *Int'l Conf. on Computer Vision* (ICCV), 2011.

AOGs representation for Events

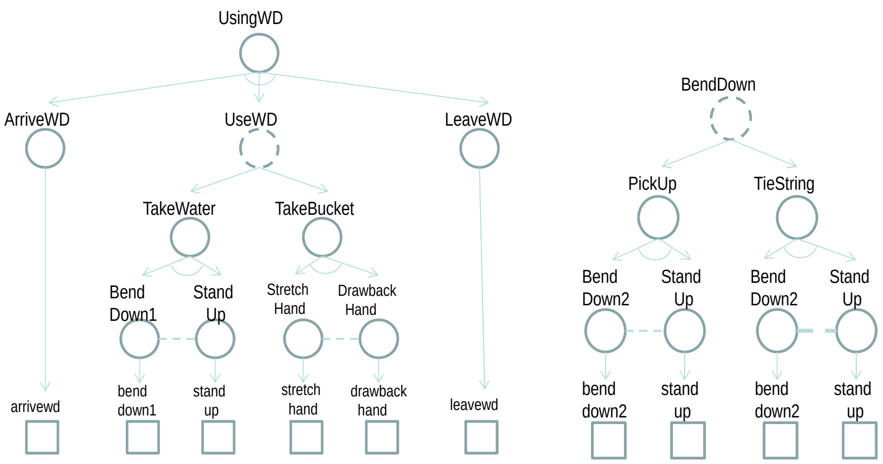


AOGs for Events

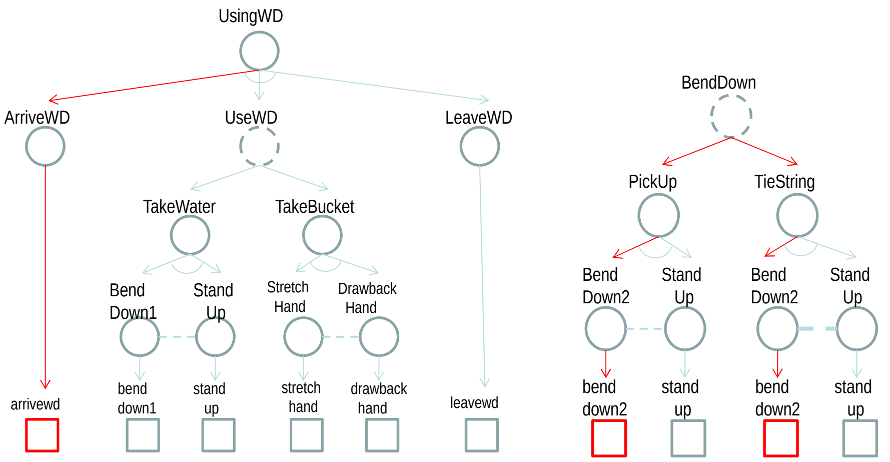
Event 2:



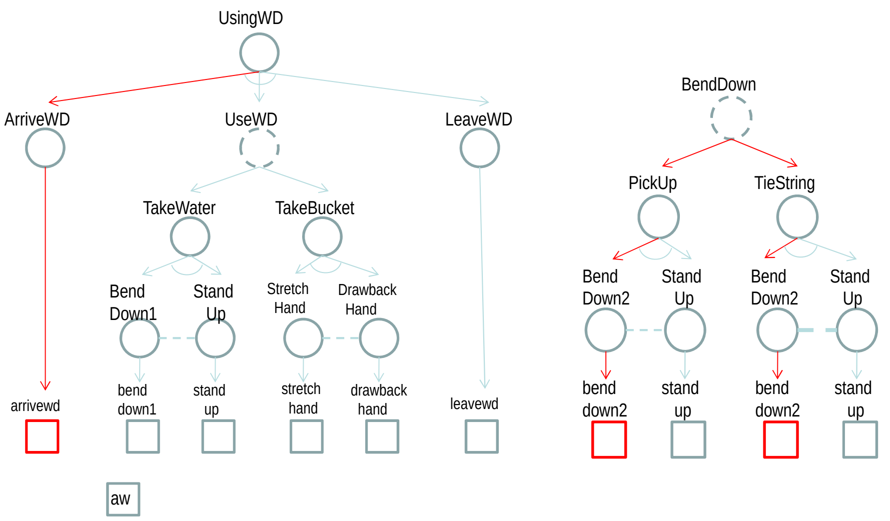
The parsing process



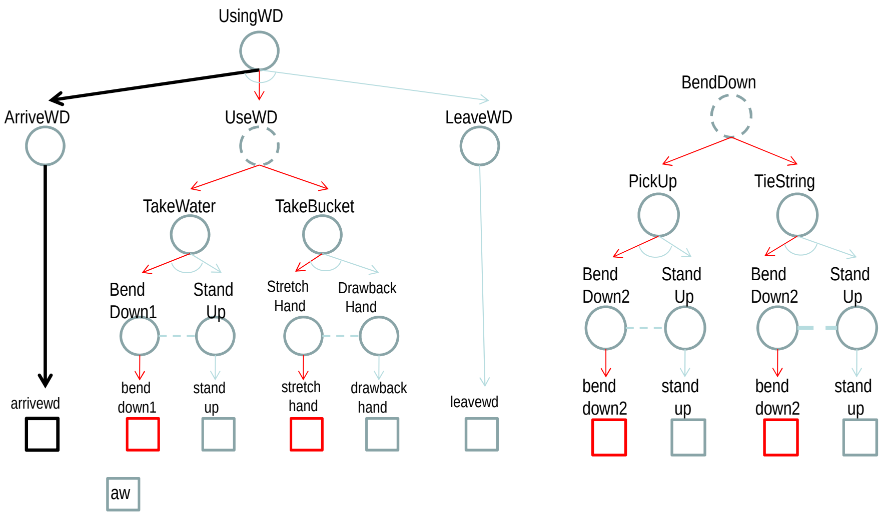
The parsing process



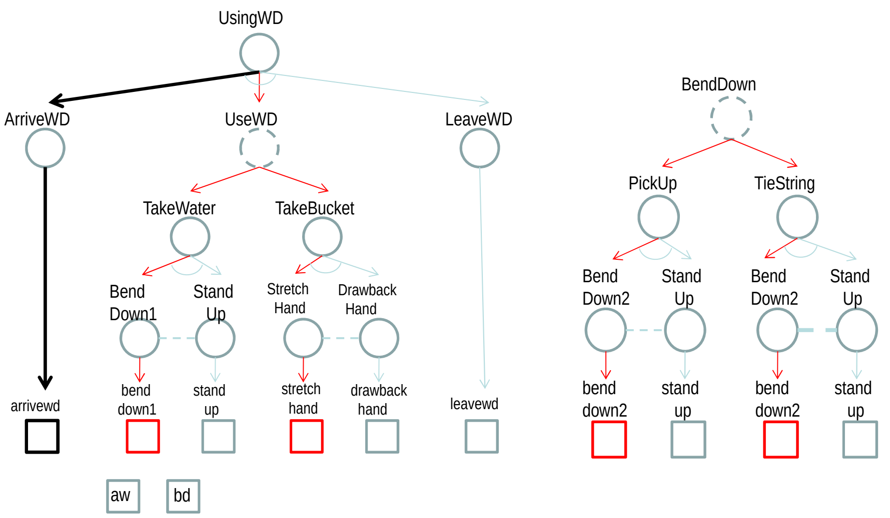
The parsing process



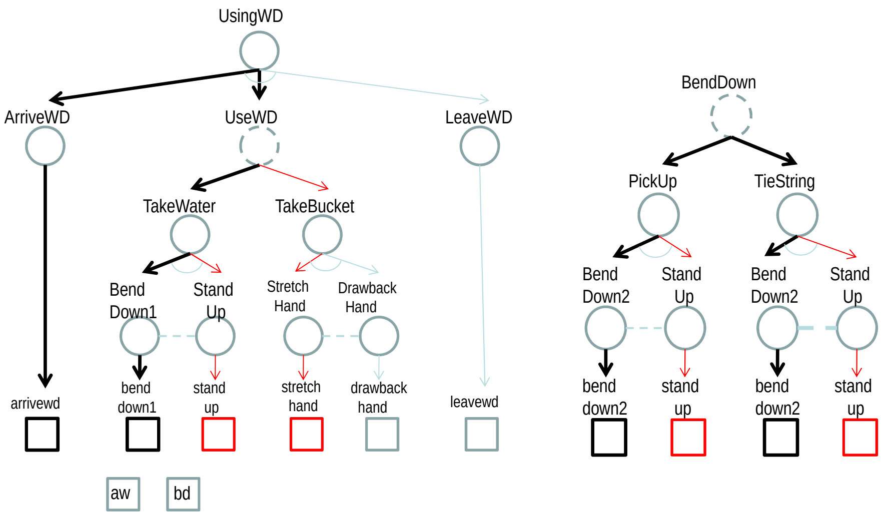
The parsing process



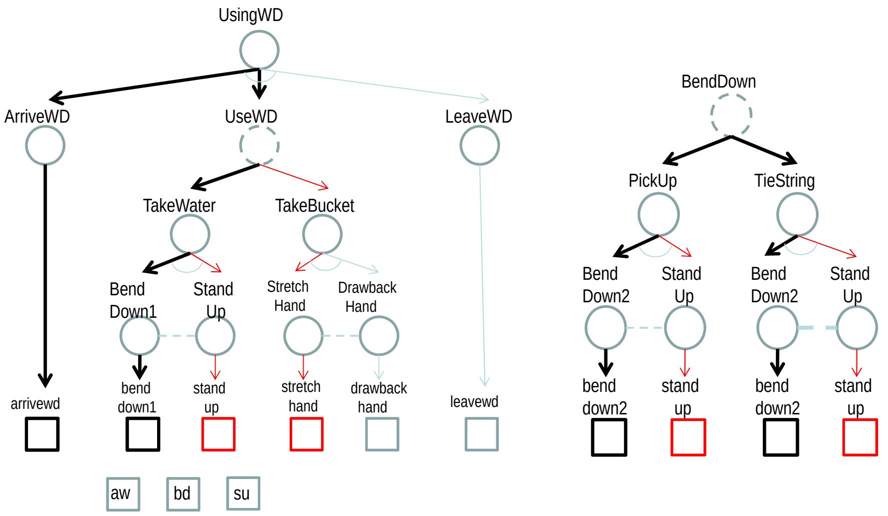
The parsing process



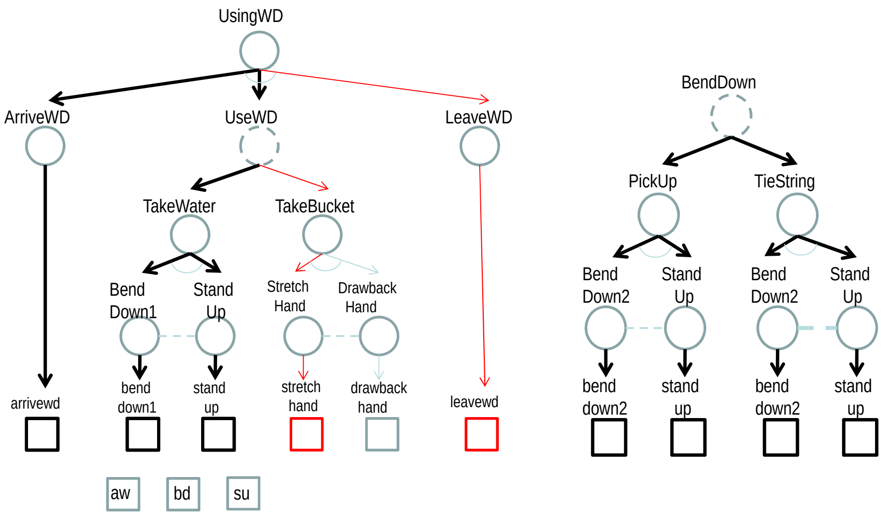
The parsing process



The parsing process

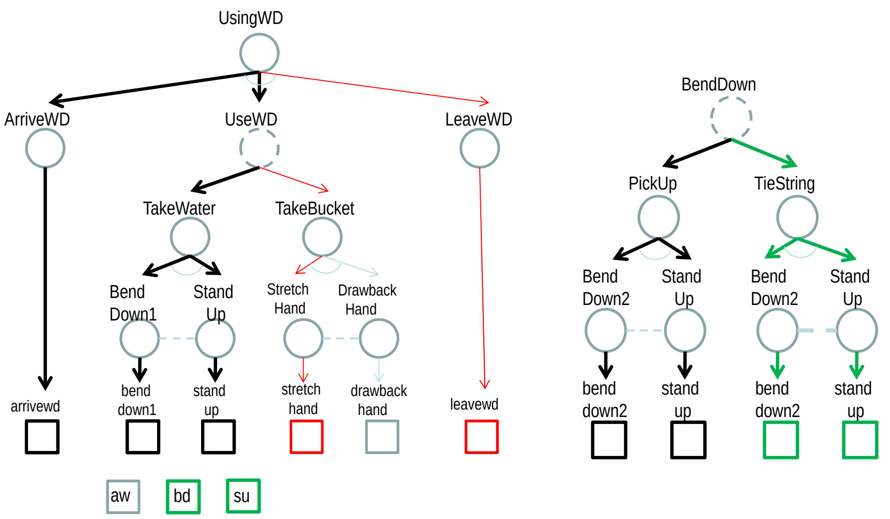


The parsing process



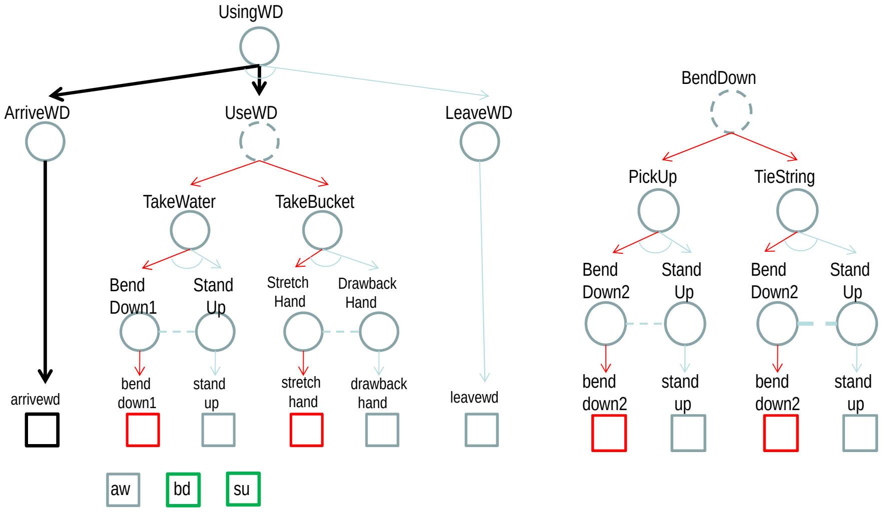
Stat 232B Stat modeling and inference, S.C. Zhu

The parsing process



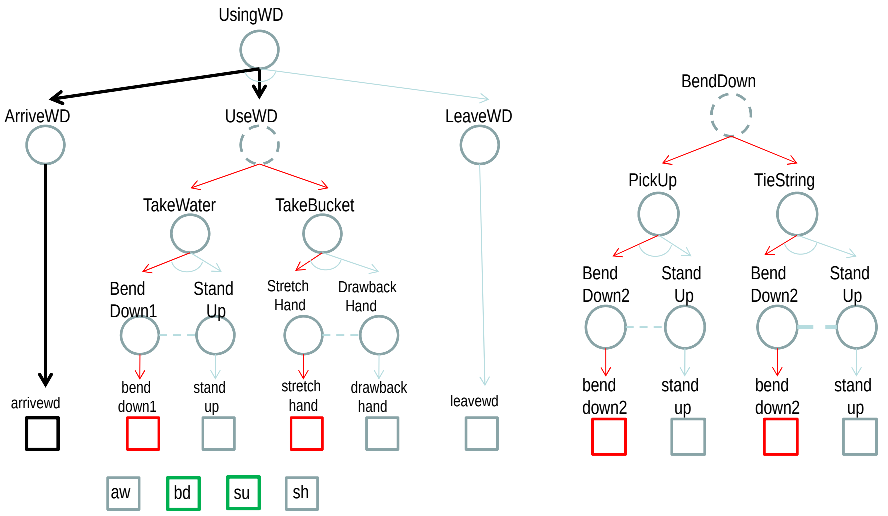
Stat 232B Stat modeling and inference, S.C. Zhu

The parsing process



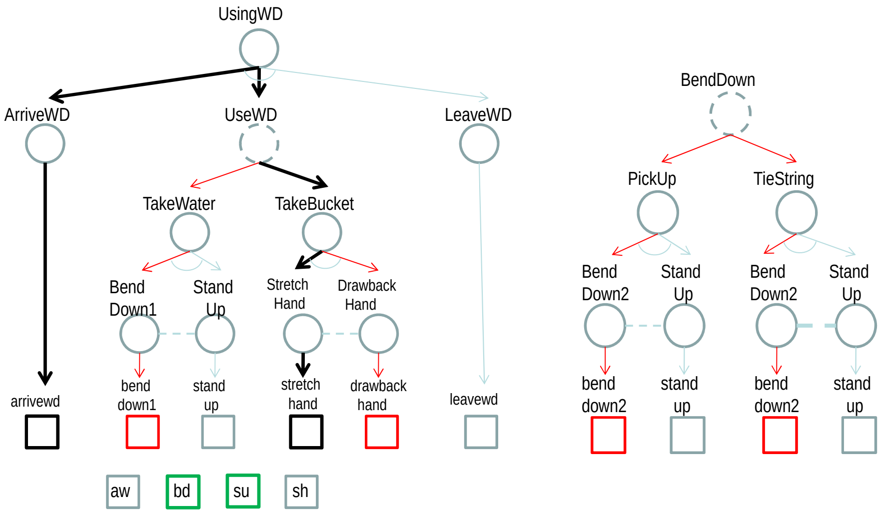
Stat 232B Stat modeling and inference, S.C. Zhu

The parsing process



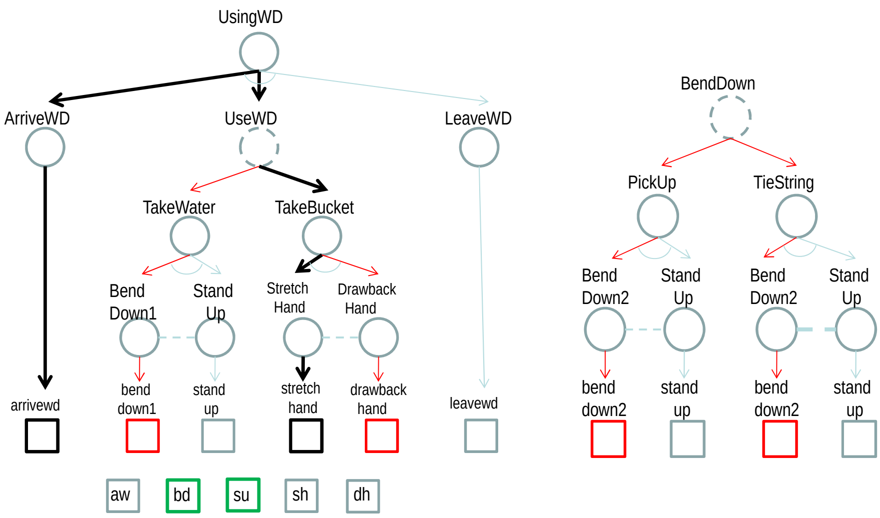
Stat 232B Stat modeling and inference, S.C. Zhu

The parsing process



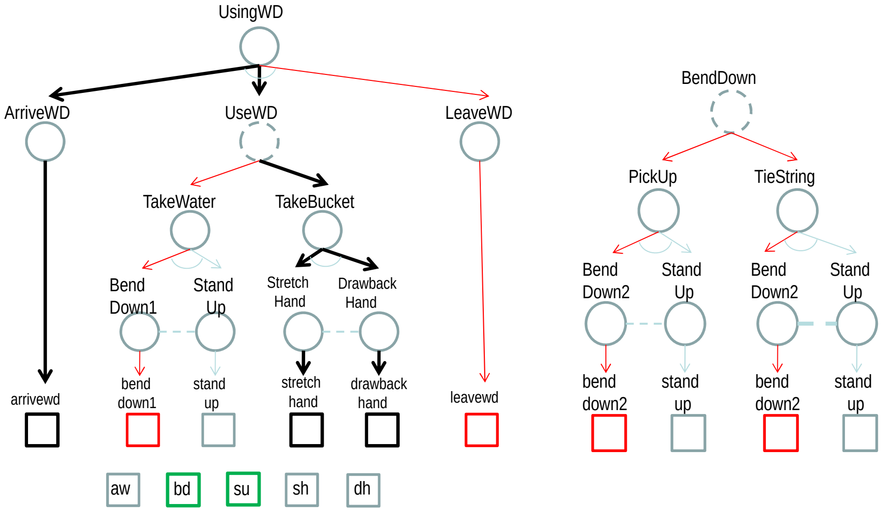
Stat 232B Stat modeling and inference, S.C. Zhu

The parsing process

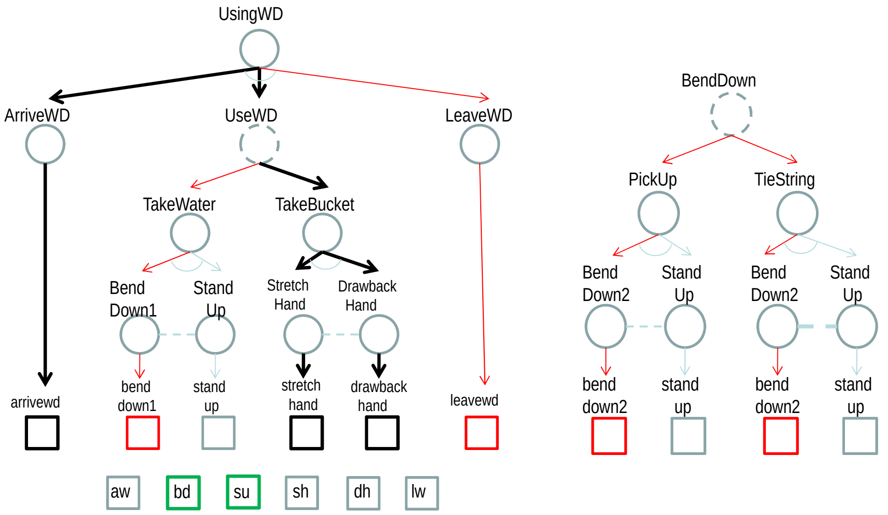


Stat 232B Stat modeling and inference, S.C. Zhu

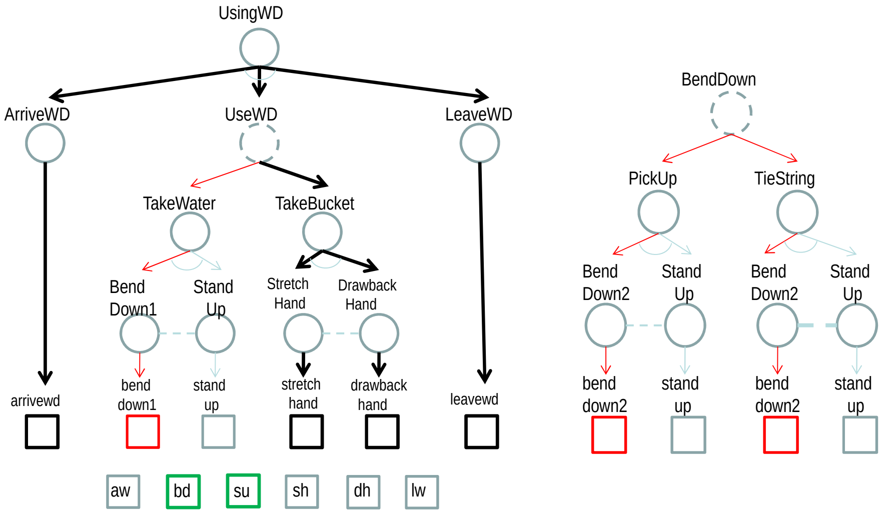
The parsing process



The parsing process

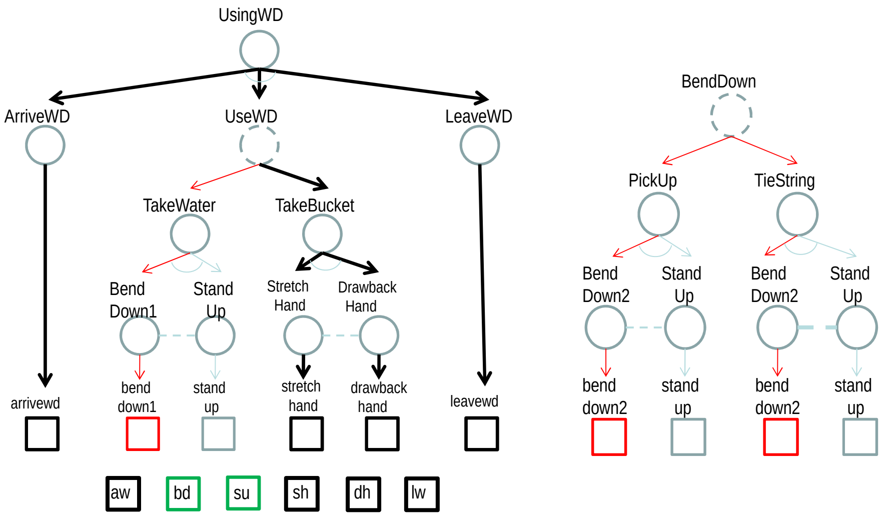


The parsing process



Stat 232B Stat modeling and inference, S.C. Zhu

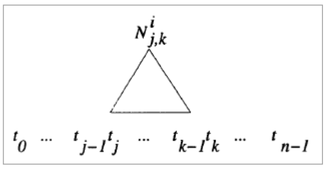
The parsing process



Stat 232B Stat modeling and inference, S.C. Zhu

6, Best-first chart parsing

- The agenda uses a priority queue to keep track of partial derivations.
- The order of constituents in agenda is calculated based on some figures of merit of constituents which measure the likelihood that the constituents will appear in a correct parse, rather than simply popping constituents off of a stack (which simulates DFS) or a queue (which simulates BFS).



Constituent $N^i_{j,k}$ in a sentence $t_{0,n}$

Ideally, the objective is to pick the constituent that maximizes the conditional probability: $p(N^i_{j,k}|t_{0,n})$.

The key to best-first chart parsing is how to estimate it.

Ideal figures of merit

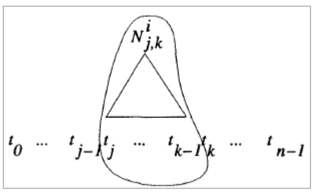
$$p(N^i_{j,k}|t_{0,n}) = \frac{p(N^i_{j,k},t_{0,n})}{p(t_{0,n})} = \frac{p(N^i_{j,k},t_{0,j},t_{j,k},t_{k,n})}{p(t_{0,n})}$$

$$= \frac{p(t_{0,j},N^i_{j,k},t_{k,n})p(t_{j,k}|t_{0,j},N^i_{j,k},t_{k,n})}{p(t_{0,n})}$$

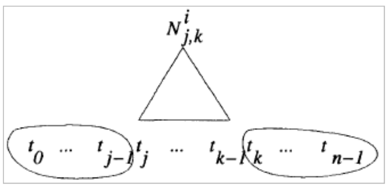
$$\approx \frac{p(t_{0,j},N^i_{j,k},t_{k,n})p(t_{j,k}|N^i_{j,k})}{p(t_{0,n})} = \frac{p^{out}(N^i_{j,k})p^{in}(N^i_{j,k})}{p(t_{0,n})}$$

where the inside probability $p^{in}(N^i_{j,k}) = p(t_{j,k}|N^i_{j,k})$

and the outside probability $p^{out}(N^i_{j,k}) = p(t_{0,j}, N^i_{j,k}, t_{k,n})$



Inside probability includes only words within the constituent



Outside probability includes the entire context of the constituent

Simple designs for the figures of merit

(1) Straight inside probability: $p^{in}(N^i_{j,k}) = p(t_{j,k}|N^i)$, which prefers shorter constituents to longer ones, resulting in a "thrashing" effect due to omit $p^{out}(N^i_{j,k})$ and $p(t_{0,n})$.

(2) Normalized per-word inside probability: $\sqrt[k-j]{p^{in}(N^i_{j,k})}$

(3) Trigram estimate:

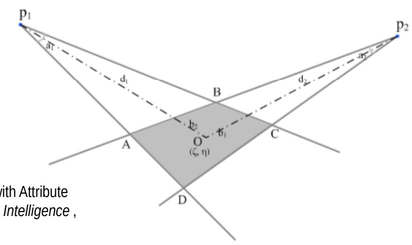
$$\begin{aligned} p(N^i_{j,k}|t_{0,n}) &= \frac{p(N^i_{j,k},t_{0,n})}{p(t_{0,n})} \\ &= \frac{p(t_{0,j},t_{k,n})p(N^i_{j,k}|t_{0,j},t_{k,n})p(t_{j,k}|N^i_{j,k},t_{0,j},t_{k,n})}{p(t_{0,j},t_{k,n})p(t_{j,k}|t_{0,j},t_{k,n})} \\ &\approx \frac{p(N^i_{j,k}|t_{0,j},t_{k,n})p^{in}(N^i_{j,k})}{p(t_{j,k}|t_{0,j},t_{k,n})} \end{aligned}$$

Assume $p(N^i_{j,k}|t_{0,j},t_{k,n}) \approx p(N^i_{j,k}) = p(N^i)$ and use a trigram model for $p(t_{j,k}|t_{0,j},t_{k,n}) \approx p(t_{j,k}|t_{j-2},t_{j-1}) = \prod_{a=j}^{k-1} p(t_a|t_{a-2},t_{a-1})$we have,

$$= \frac{p(N^i)p^{in}(N^i_{j,k})}{p(t_{j,k}|t_{j-2},t_{j-1})}$$

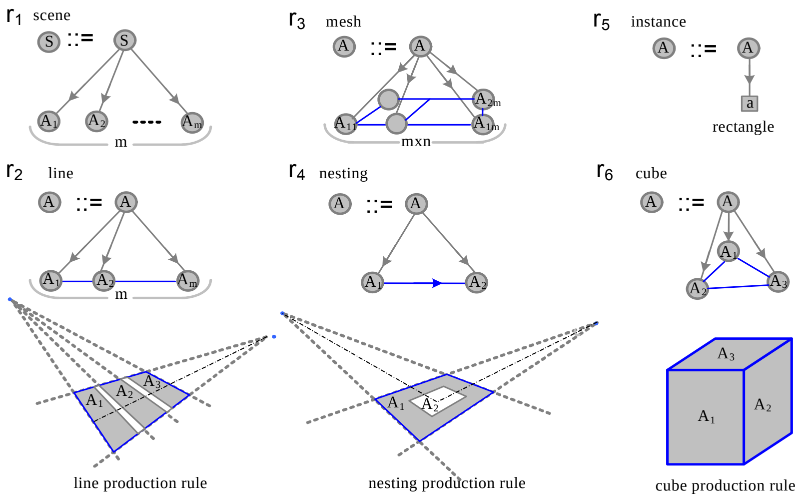
7, FoM example: scene parsing

One terminal sub-template
--- a planar rectangle in 3-space



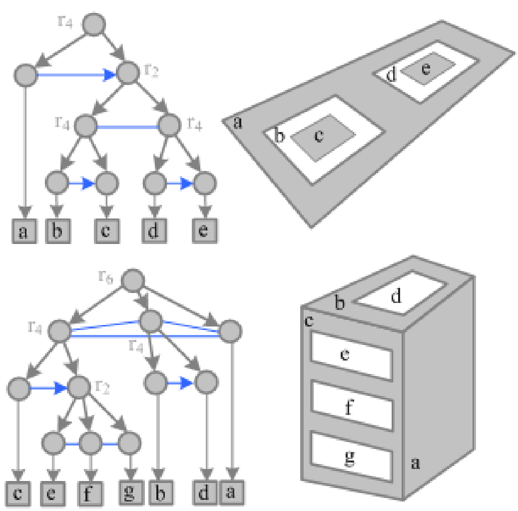
F. Han and S.C.Zhu, "Bottom-up/Top-down Image Parsing with Attribute Graph Grammar," *IEEE Trans. on Pattern Analysis and Machine Intelligence* , vol.31, no.1, pp 59-73, Jan. 2009.

Six grammar rules which can be used recursively



Stat 232B Stat modeling and inference, S.C. Zhu

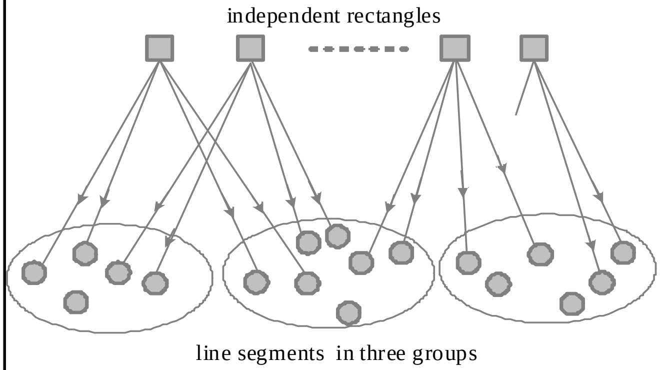
Two configuration examples

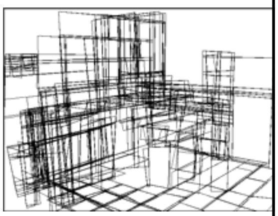


Stat 232B Stat modeling and inference, S.C. Zhu

Bottom-up detection (proposal) of rectangles

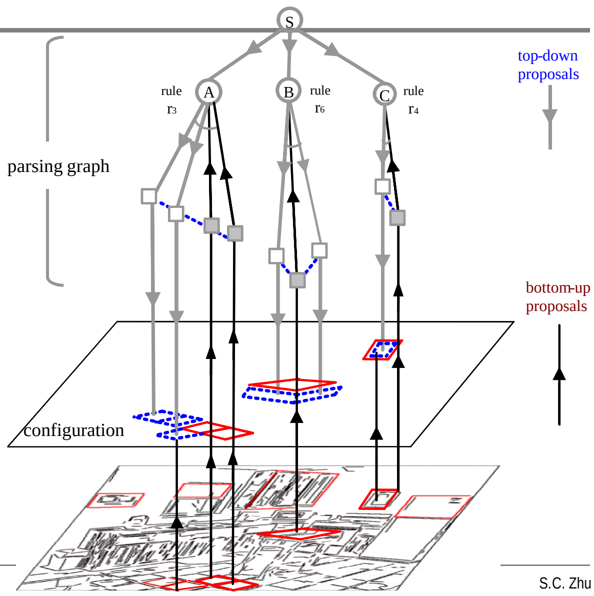
Each rectangle consists of two pairs of line segments that share a vanish point.





Top-down / bottom-up inference

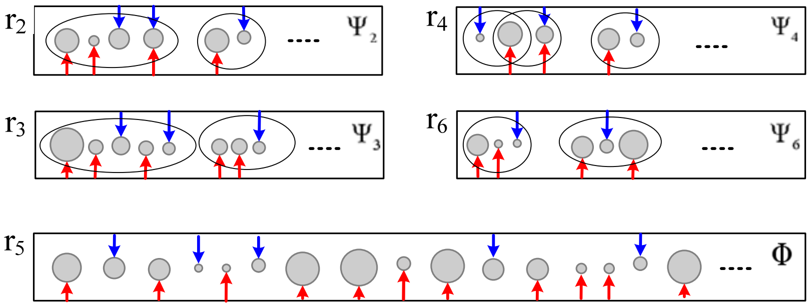
Red ones are in chart,
blue ones are in agenda



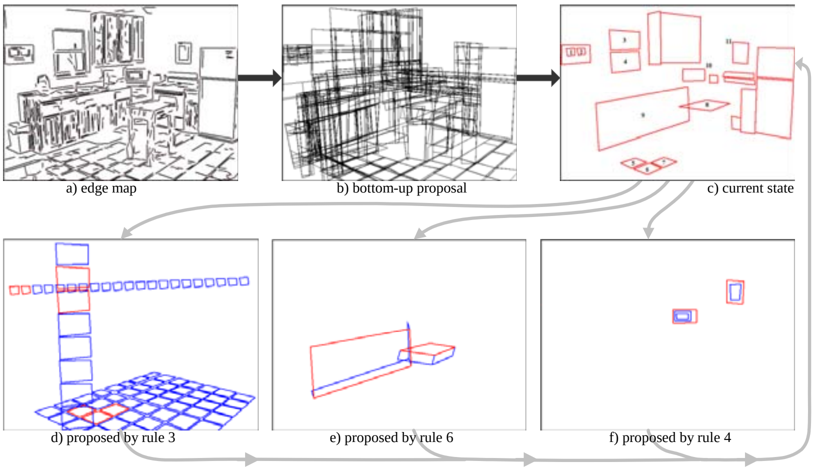
Each grammar rule is an assembly line and maintains an Open-list and Closed-list of particles

A particle is a production rule partially matched, its probability measures an approximated posterior probability ratio.

Open-list or agenda

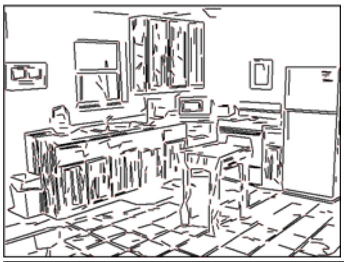


Example of top-down / bottom-up inference

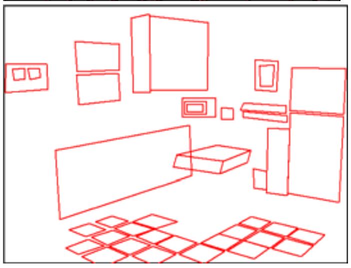


Results

(Han and Zhu, 05)

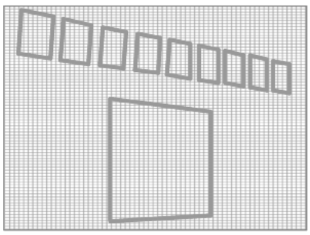


Edge map



Rectangles inferred

Likelihood model based on primal sketch



$$\Lambda = \Lambda_{\mathrm{sk}} \cup \Lambda_{\mathrm{nsk}},$$

$$\Lambda_{\mathrm{sk}} = \cup_{k=1}^{N} \Lambda_{\mathrm{sk,k}}$$

$$\Lambda_{\mathrm{nsk}} = \cup_{m=1}^{M} \Lambda_{\mathrm{nsk,m}},$$

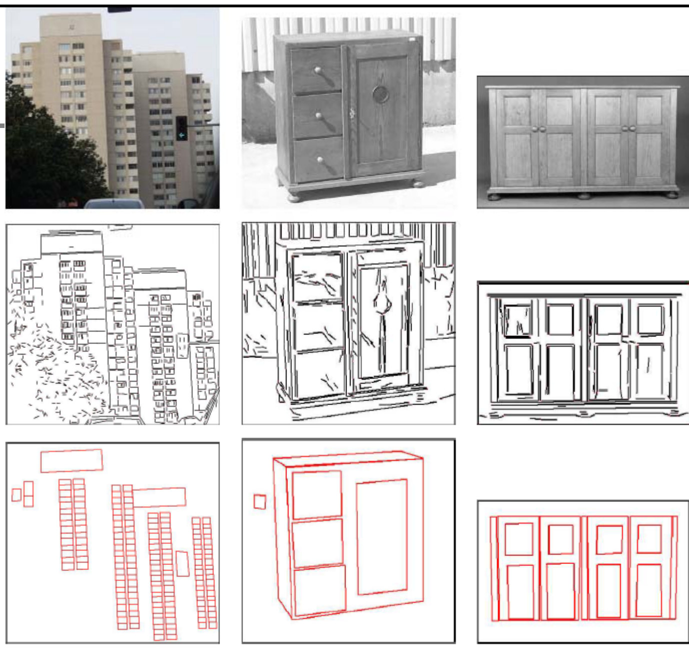
$$\Lambda_{\mathrm{nsk,m_1}} \cap \Lambda_{\mathrm{nsk,m_2}} = \emptyset, m_1 \neq m_2$$

$$p(\mathbf{I}_{\mathrm{sk,k}}|C) \propto \exp\{-\sum_{(x,y)\in\Lambda_{\mathrm{sk,k}}} \frac{(\mathbf{I}(x,y) - B_k(x,y))^2}{2\sigma^2}\}$$

$$p(\mathbf{I}|C(\mathbf{G})) = \frac{1}{Z}\exp\{-\sum_{k=1}^{N}\sum_{(x,y)\in\Lambda_{\mathrm{sk,k}}} \frac{(\mathbf{I}(x,y) - B_k(x,y))^2}{2\sigma^2} - \sum_{m=1}^{M}\sum_{i=1}^{n}\langle\beta_{mi}, h_i(\mathbf{I}_{\Lambda_{\mathrm{nsk,m}}})\rangle\}$$

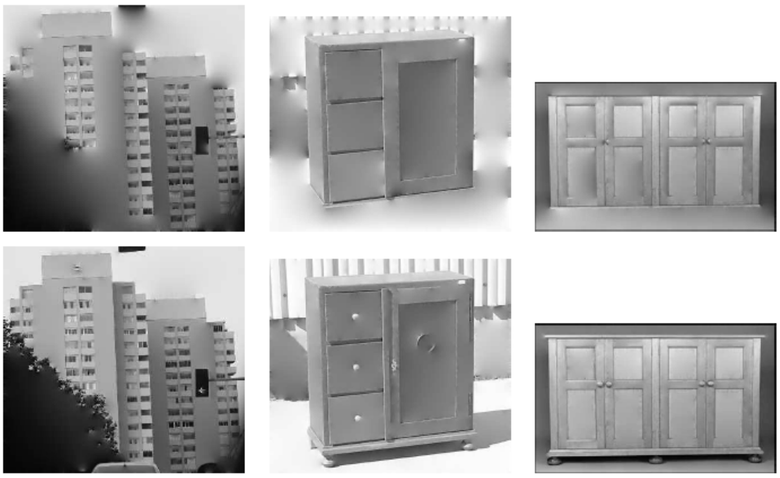
Synthesis based on the parsing model

Stat 232B Stat modeling and inference, S.C. Zhu

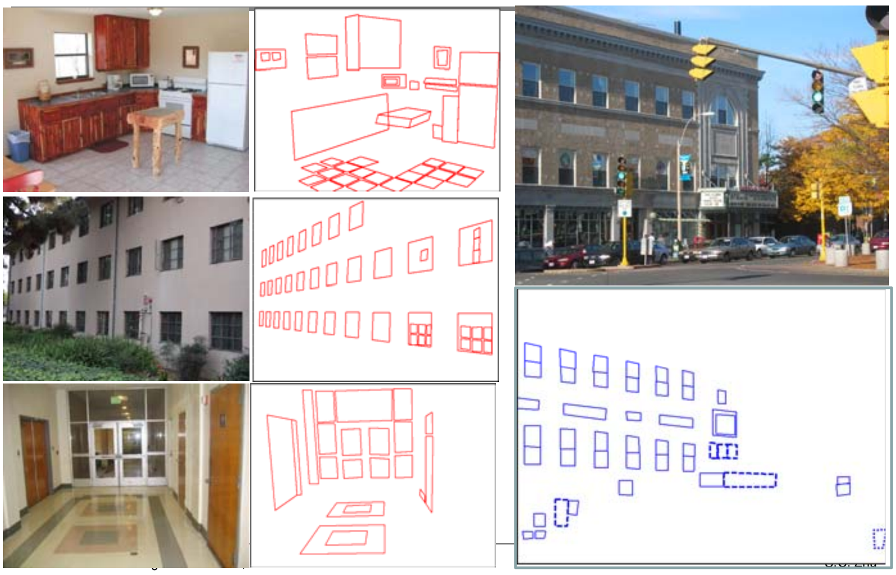


Stat 232B Stat modeling and inference, S.C. Zhu

Synthesis based on the parsing model



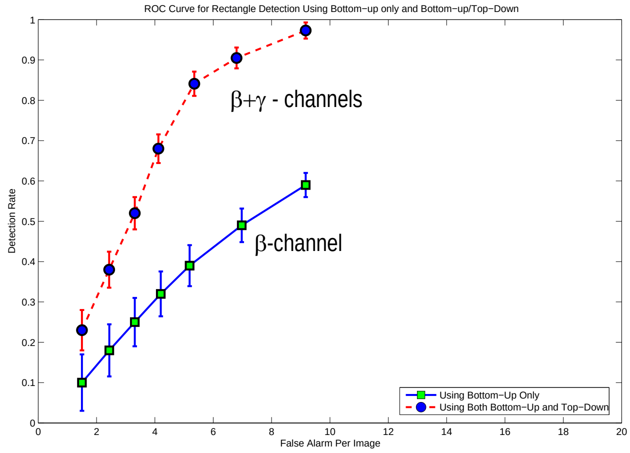
Parsing rectangular scenes by grammar



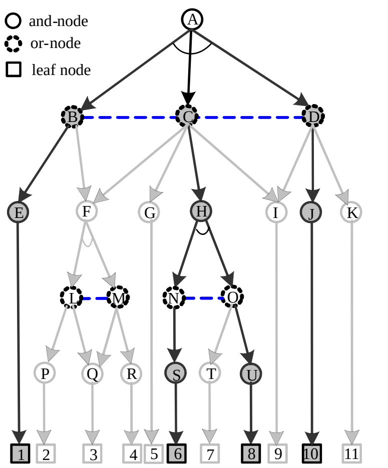
How much does top-down improve bottom-up?

In the rectangle experiments:

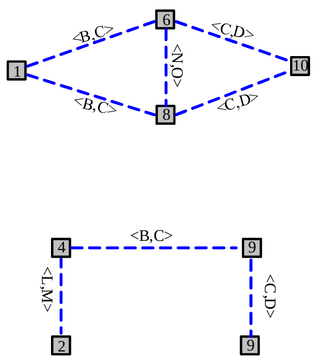
Han and Zhu, 2005-07



8, Chart-agenda parsing with And-Or graphs



some graph configurations generated by the AndOr graph



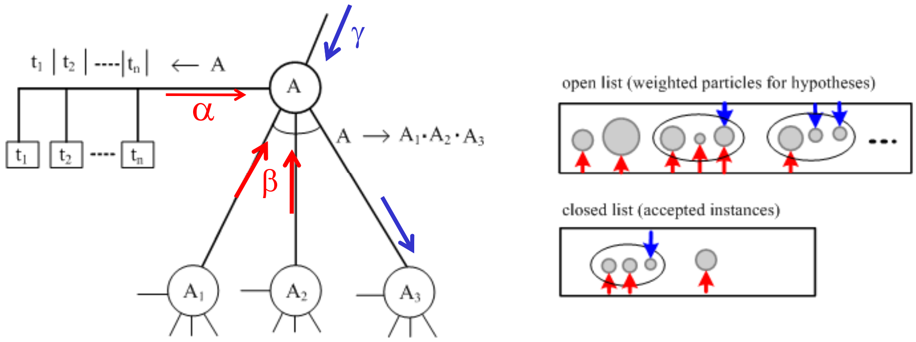
α, β and γ computing processes in AoG

The And-Or graph is a recursive structure. So, consider a node A.

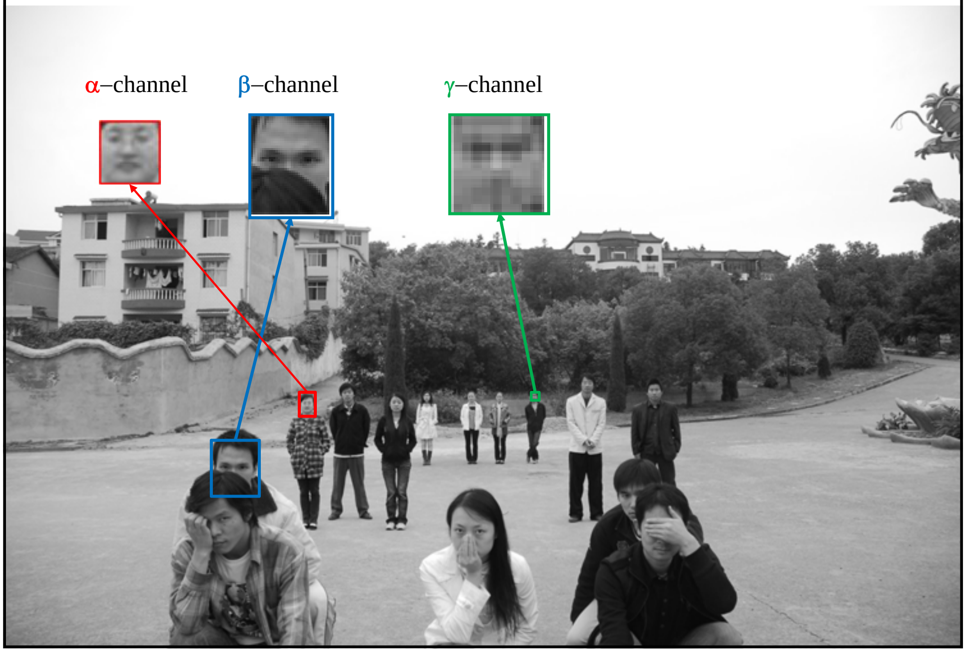
1, any node A terminate to leaf nodes at a coarse scale (ground).

2, any node A is connected to the root.

Starting the $\alpha/\beta/\gamma$ channels when they are applicable ---an optimal scheduling problem



An example: human faces are computed in 3 channels



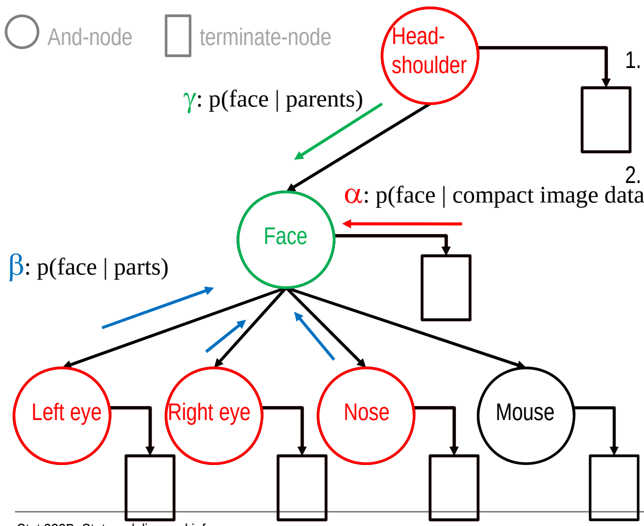
Human faces in real scenarios

α-channel

β-channel

γ-channel

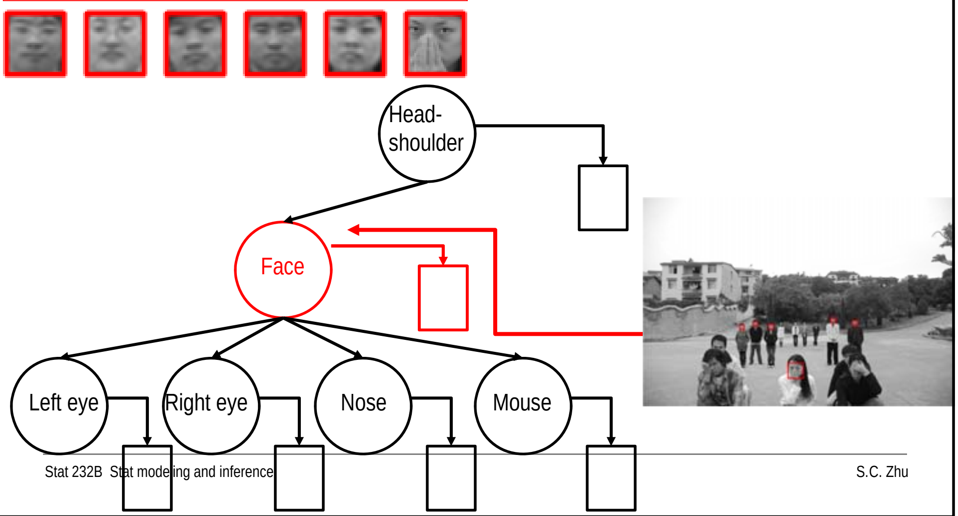
Hierarchical modeling and α, β and γ computing



1. Each node has its own α, β and γ computing processes.
2. How much does each channel contribute?

α processes for the face node

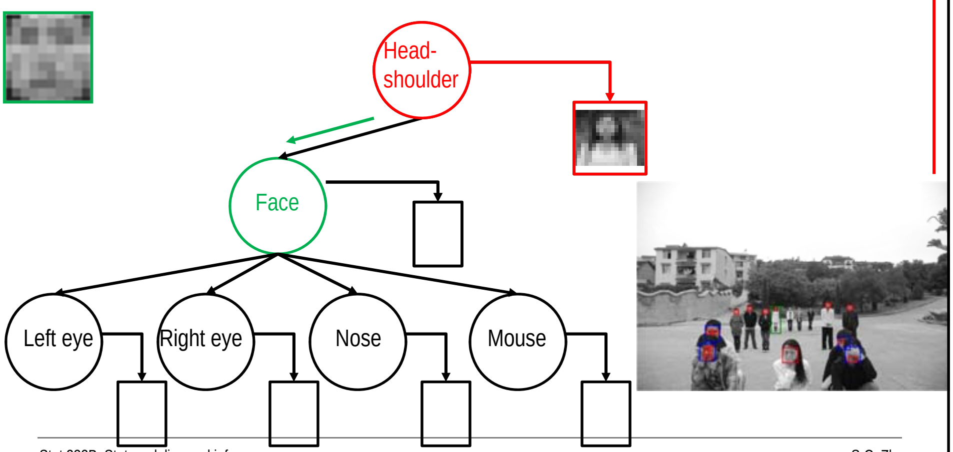
α-channels: p(face | compact image data)



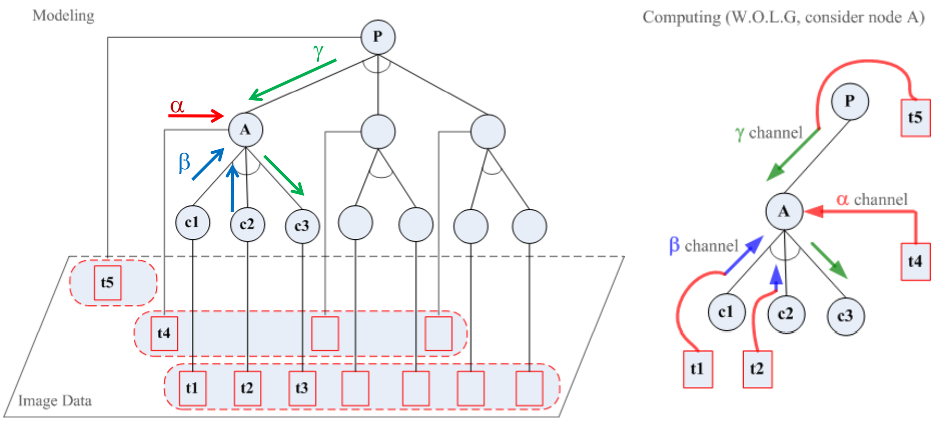
γ processes for the face node(when it's α and β is off)

γ-channels: p(face | parents), predicting

α-channels of some parents are on



In general: recursive α, β and γ channels



Deriving the log-probabilities

$$pg^* = \arg \max_{pg \in \Omega_{pg}} p(pg)p(I_{\Lambda_{pg}}|pg)$$

$$p(I_{\Lambda_{pg}}|pg) = q(I_{\Lambda_k})\frac{p(I_{\Lambda_{pg}}|pg)}{q(I_{\Lambda_{pg}})}$$

$$\begin{aligned}
pg^* = \arg\max_{pg}\{&\log p(A|O) + \sum_{i=1}^{2} \log p(X(C_i)|X(A)) \\
&+ \underbrace{\log \frac{p(I_{\Lambda_{t_A}}|t_A)}{q(I_{\Lambda_{t_A}})}}_{\alpha(A)\ \text{process}} \\
&+ \underbrace{[\underbrace{\sum_{i=1}^{2} \log \frac{p(I_{\Lambda_{t_{C_i}}}|t_{C_i})}{q(I_{\Lambda_{t_{C_i}}})}}_{\alpha(t_{C_i})\ \text{process}} + \underbrace{\log p(X(C_1), X(C_2))]}_{\text{binding model}}}_{\beta(A)\ \text{process}} \\
&+ \underbrace{[\underbrace{\log \frac{p(I_{\Lambda_{t_P}}|t_P)}{q(I_{\Lambda_{t_P}})}}_{\alpha(P)\ \text{process}} + \underbrace{\log p(X(A)|X(P))]\}}_{\text{prediction model}}}_{\gamma(A)\ \text{process}} \quad (
\end{aligned}$$

T.F. Wu and S.C. Zhu, "A Numeric Study of the Bottom-up and Top-down Inference Processes in And-Or Graphs," *Int'l Journal of Computer Vision*, vol.93, no 2, p226-252, June 2011.

Deriving the log-probabilities

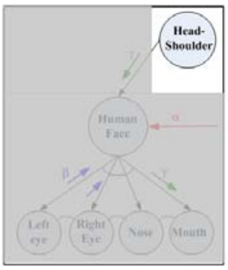
Log probabilities becomes the weights for each node:

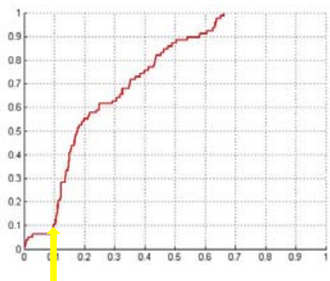
$$w_A^{\alpha} = \log \frac{p(I_{\Lambda_{t_A}} | t_A)}{q(I_{\Lambda_{t_A}})}$$

$$w_A^{\beta(\mathbf{c})} = \sum_{i=1}^{2} \log \frac{p(I_{\Lambda_{t_{C_i}}} | t_{C_i})}{q(I_{\Lambda_{t_{C_i}}})} + \log p(X(C_1), X(C_2))$$

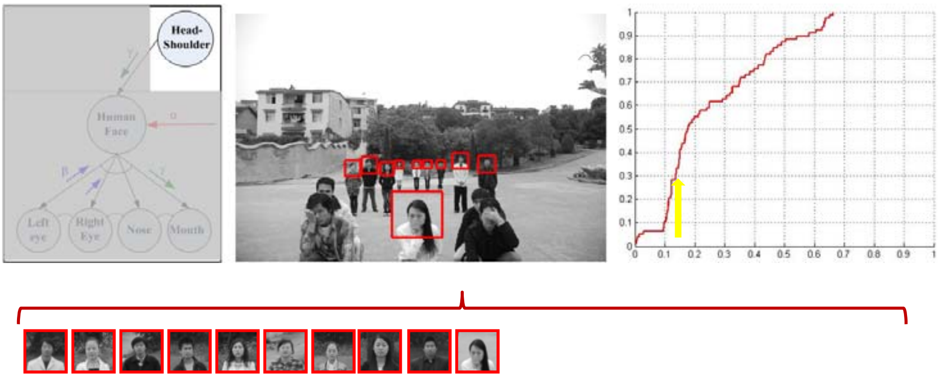
$$w_A^{\gamma(P)} = \log \frac{p(I_{\Lambda_{t_P}} | t_P)}{q(I_{\Lambda_{t_P}})} + \log p(X(A)|X(P))$$

α–channel: head-shoulder



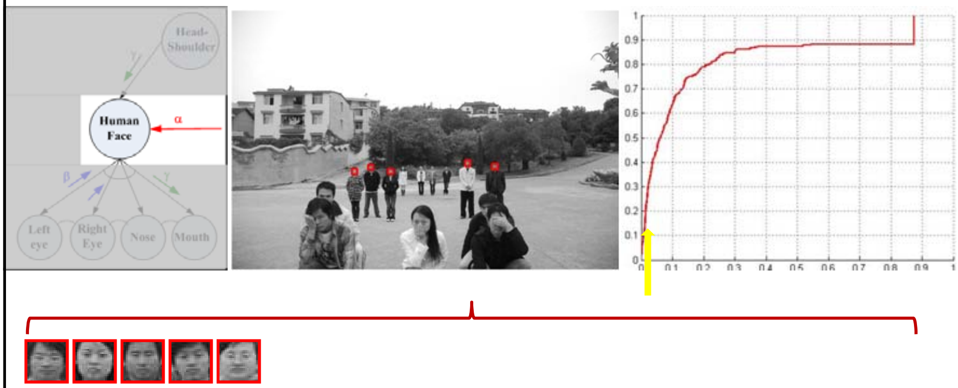


α–channel: head-shoulder

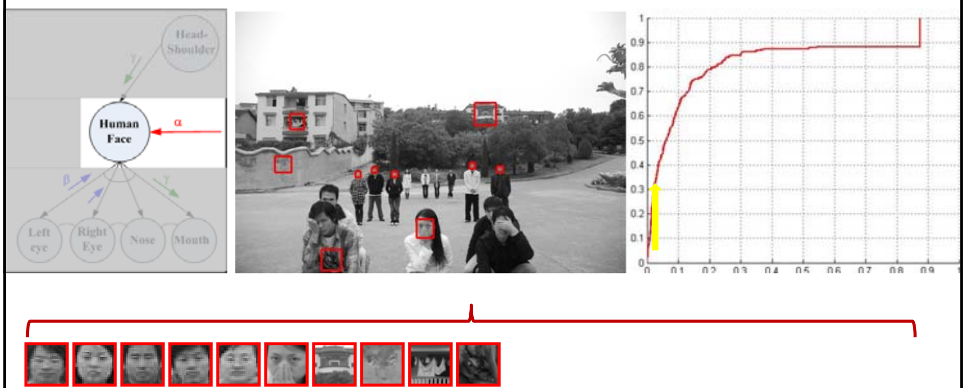


α–channel: head-shoulder

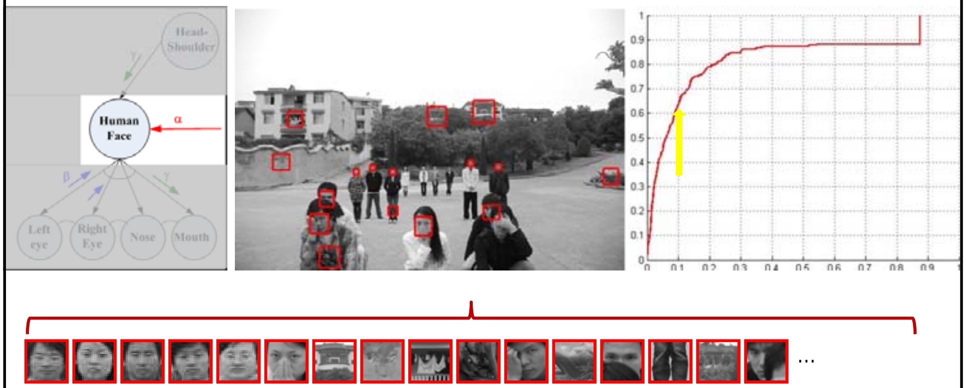
α–channel: face



α–channel: face

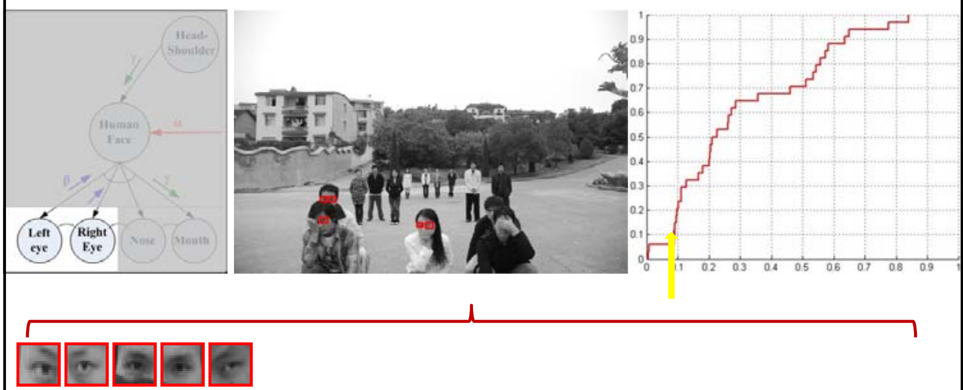


α–channel: face



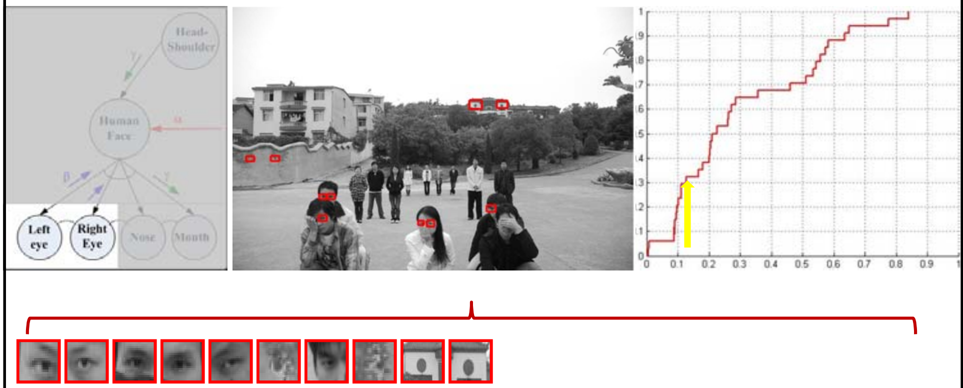
Stat 232B Stat modeling and inference, S.C. Zhu

α–channel: eye



Stat 232B Stat modeling and inference, S.C. Zhu

α–channel: eye

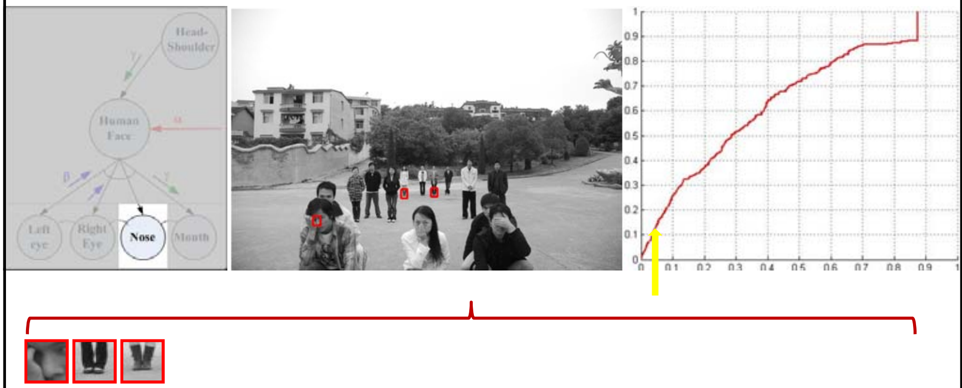


Stat 232B Stat modeling and inference, S.C. Zhu

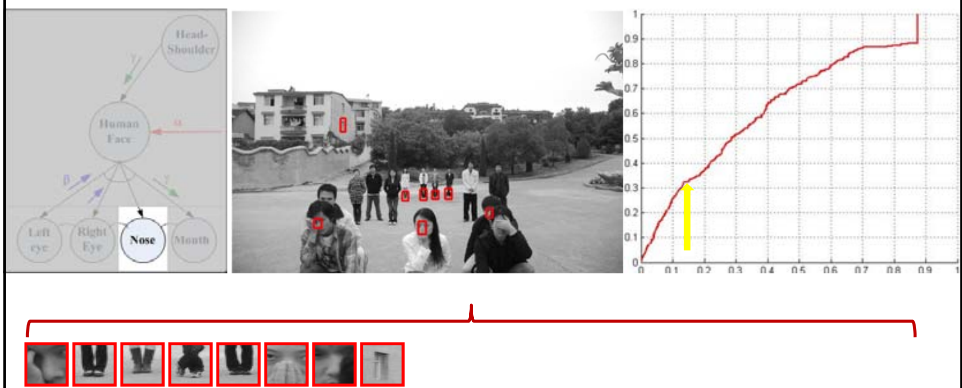
α–channel: eye

Stat 232B Stat modeling and inference, S.C. Zhu

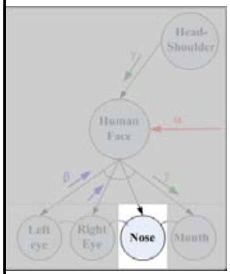
α–channel: nose

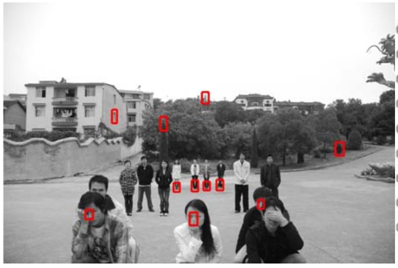


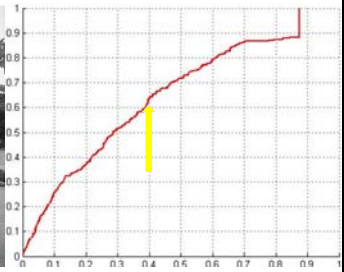
α–channel: nose

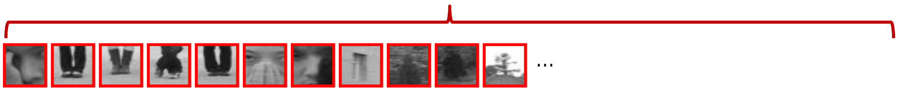


α–channel: nose



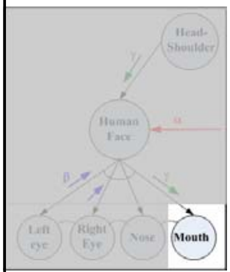


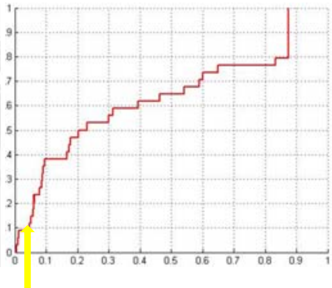


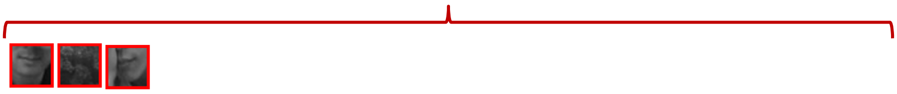


...

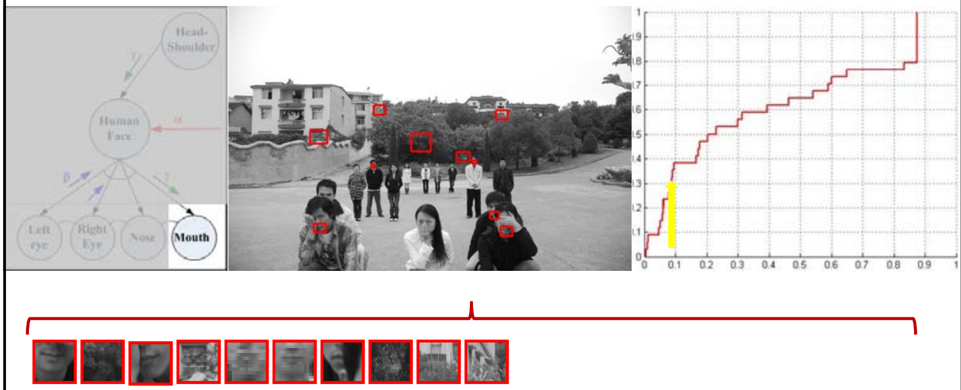
α–channel: mouth







α–channel: mouth



α–channel: mouth

…

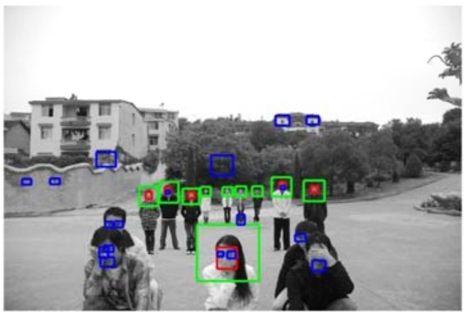
All α channels

keep things and throw away stuffs by
integrating α, β and γ channels

infe

S.C. Zhu

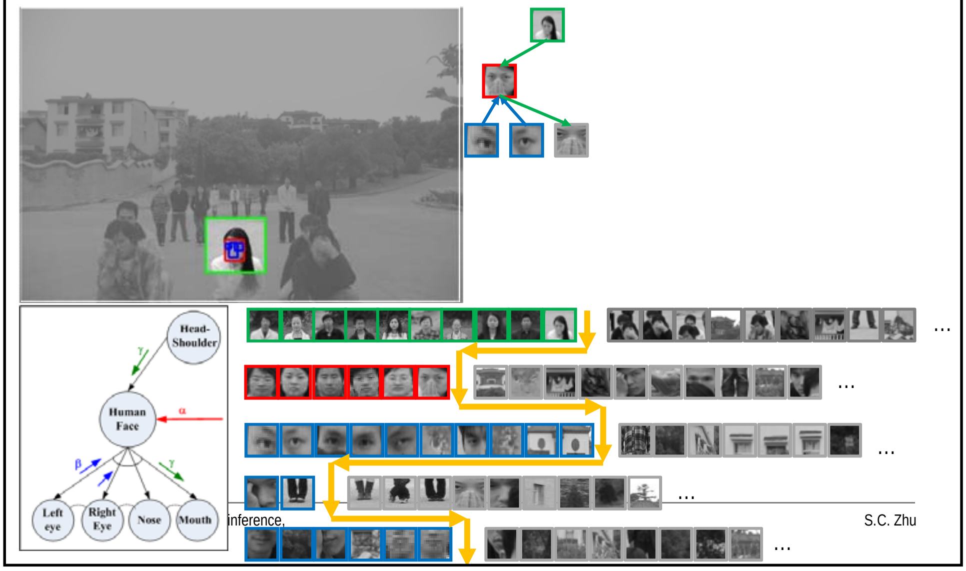
Integrating α, β and γ channels



inference.

S.C. Zhu

Integrating α, β and γ channels



inference,

S.C. Zhu

Integrating α, β and γ channels

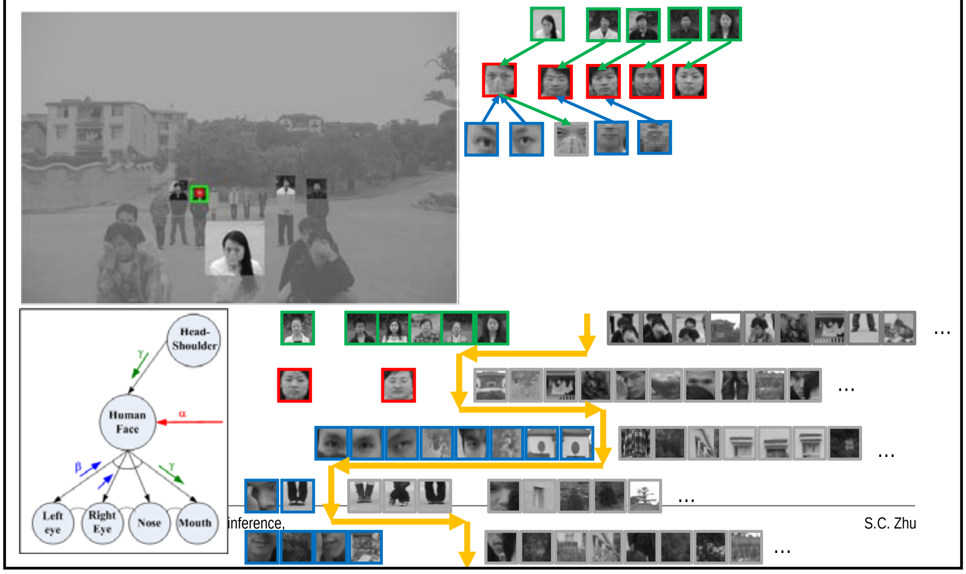
inference,

S.C. Zhu

Integrating α, β and γ channels

Integrating α, β and γ channels

Integrating α, β and γ channels



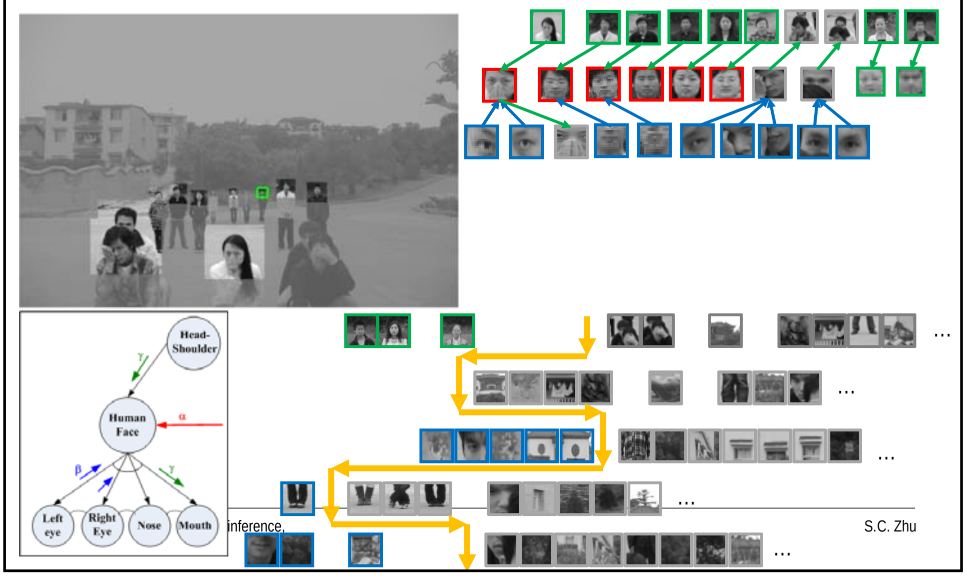
Integrating α, β and γ channels

Integrating α, β and γ channels

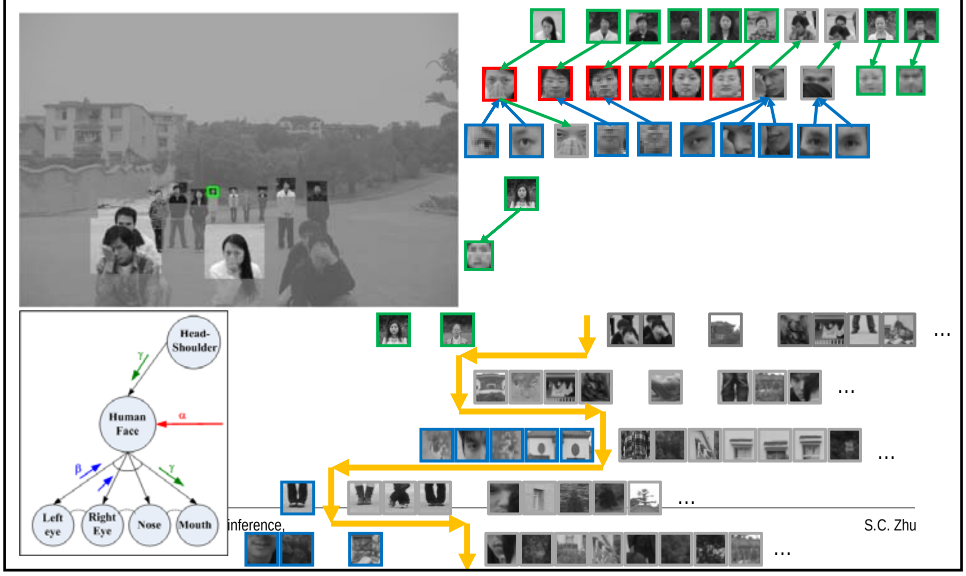
Integrating α, β and γ channels

Integrating α, β and γ channels

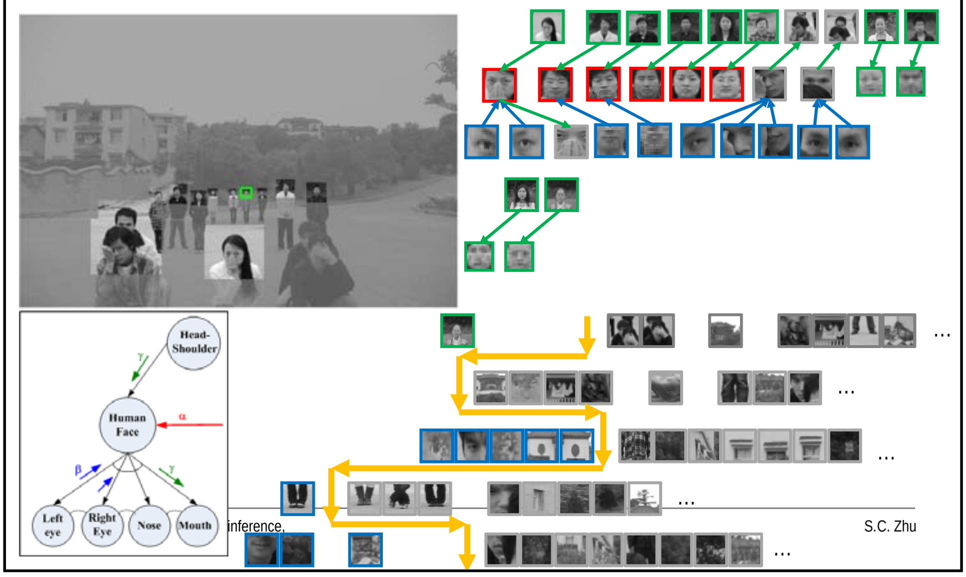
Integrating α, β and γ channels



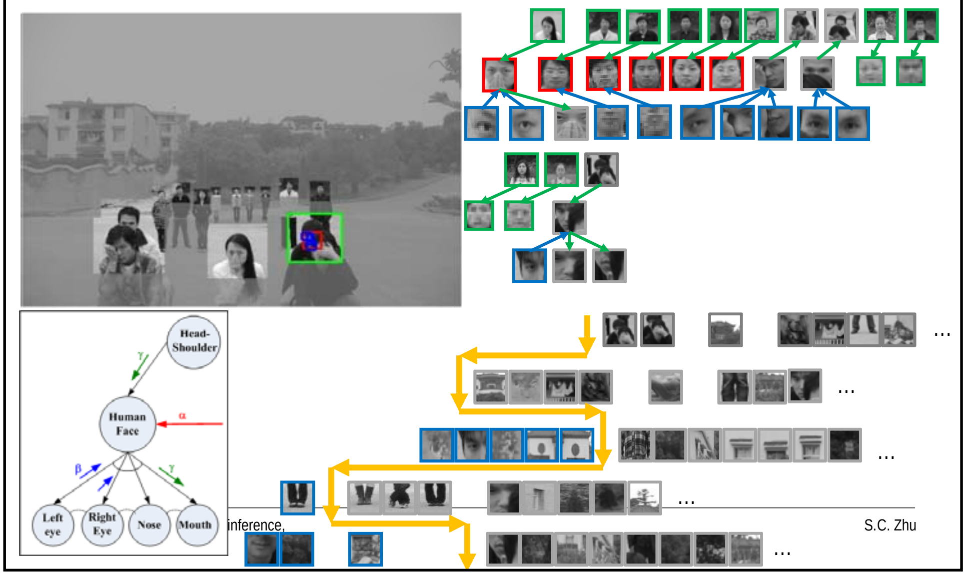
Integrating α, β and γ channels



Integrating α, β and γ channels

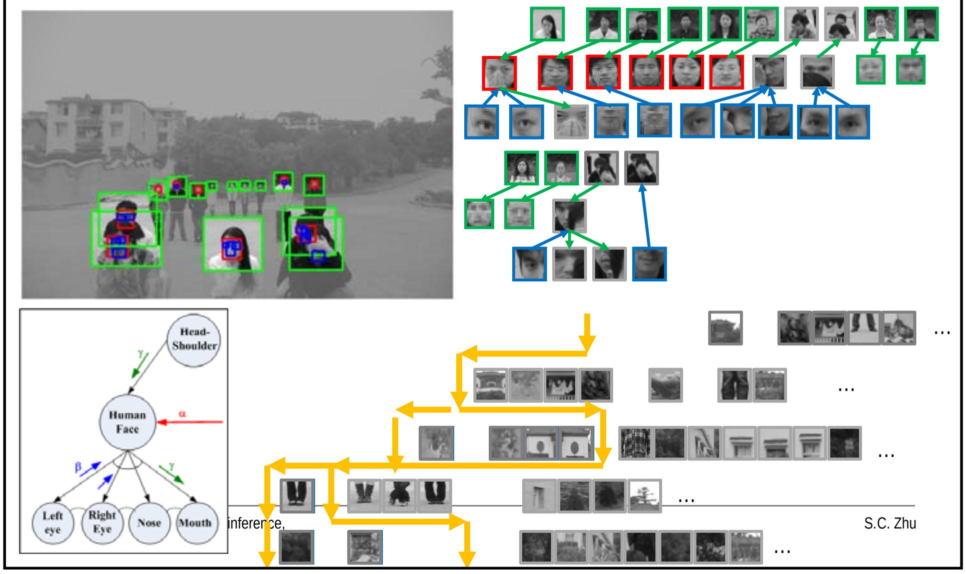


Integrating α, β and γ channels



Integrating α, β and γ channels

Integrating α, β and γ channels



Performance improvement

red for α, blue for $\alpha+\beta$, green for $\alpha+\gamma$, cyan for $\alpha+\beta+\gamma$ channels